

**MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN
TASHKENT STATE TECHNICAL UNIVERSITY**

Registered

№ _____
« _____ » _____ 2022

«APPROVED»

Vice Rector for Academic Affairs

_____ **O.O. Zaripov**
« _____ » _____ 2022



Department Information technologies

Educational and methodological complex

on subject

"Information technologies in technical systems"

Field of knowledge: 700 000 – Engineering, manufacturing and construction industries.
1 000 000 – Services.

Field of education: 710 000 – Engineering.
720 000 – Industrial and technological sphere.

Direction of education: For all directions in the field of education.

Tashkent – 2022

The educational and methodological complex is compiled on the basis of the discipline program approved by order from № _____ « _____ » _____ 2022г.

Authors: - *Sagatov M.V.* – professor of the Tashkent State Technical University, Doctor of Technical Sciences, Head of the Department of Information Technologies;

Tashmatova Sh.S. - Senior Lecturer at the Department of Information Technologies, Tashkent State Technical University.

Kurbanova K.E. - Senior Lecturer at the Department of Information Technologies, Tashkent State Technical University.

Khamdamova S.M. - Senior Lecturer at the Department of Information Technologies, Tashkent State Technical University.

The educational and methodological complex was discussed at a meeting of the Department of Information Technology of the Faculty of Electronics and Automation and submitted for consideration by the educational and methodological council of the faculty (protocol No. ____ dated " ____ " _____ 2022_).

Head of the department

_____ Sagatov M.V.

Secretary

_____ Akbarova Sh.A.

The educational and methodological complex was reviewed and approved at a meeting of the scientific and methodological council of the university (minutes No. ____ dated " ____ " _____ 2022).

Secretary of the Scientific and Methodological Council _____ **N. Mambetov.**

Page content

1.	The content of the lectures (modules may be included in accordance with the working program of the course).....	4
2.	Practical lessons, main text, tasks, variants, examples, instructions.....	128
3.	Laboratory work, main text, tasks, variants, equipment, links to foreign literature.....	194
4.	Instructions and recommendations for topics for performing independent work.....	259
5.	Glossary.....	262
6.	Applications	
	Syllabus	276
	Working curriculum.....	284
	Tests.....	295
7.	Used literature.....	311

MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY NAMED AFTER ISLAM
KARIMOV



Lecture notes

on subject

"Information technologies in technical systems"

Tashkent –2022

The main theoretical part (lectures)

1-module.

Introduction to the subject "Information technologies in technical systems".

Topic 1. Introduction to the subject "Information technologies in technical systems". The subject of the course, its goals and objectives. Main functions and tasks of ICT in technical systems. Principles for the implementation of ICT in technical areas, modernization of digital infrastructure in order to develop the digital economy.

2-module.

Theoretical foundations of computer-aided design systems in technical systems

Topic 2. Theoretical foundations of computer-aided design systems. Modern computer-aided design systems and their use in technical systems (CAD/CAE/CAM-systems). CAD in the field of mechanics. Classification of CAD by intended purpose, used in computer-aided design.

Topic 3 Modeling. Computer modelling. Modeling. The main types of modeling, their scope. Computer modelling. Classification of models. The principle of computer simulation.

Topic 4 Mathematical modeling. Math modeling. The use of mathematical packages for the study and analysis of mathematical models, mathematical packages (3D Max, Solid Works, Matlab and MathCAD).

Topic 5. Graphic modeling. Processing of numerical and graphic information in engineering problems. Implementation of static and dynamic mathematical models in Matlab and MathCAD systems.

Topic 6. Fundamentals of simulation modeling. Classification of simulation software. Varieties of simulation modeling. Simulation modeling using the Simulink package

Topic 7. Programming in MATLAB. Modules and their functions. M-script files and functions. Function file structure. **MATLAB** control structures.

3-module.

Issues of information security in computer networks

Topic 8 Information security systems. Mathematical foundations of cryptology. Cryptographic methods of information protection. Issues of information security in computer networks.

Topic 9. Cyberspace and the basics of cyber security. Computer technology objects used in cybersecurity

4-module.

Modern programming technologies.

Topic 10. Modern programming technologies. Programming languages. Systems of object-oriented programming. Programming languages and systems, features of their use and classification. Basic modules of programming languages.

Topic11. Object programming systems. Basic constructions of languages and features of programming in the system. Classes, methods and properties. Properties and their application to objects. Constructors and destructors. The object and its components. Components of programs.

Topic12. Logic programming technology. The logical structure of the program. Conditional, unconditional, and select statements

Topic13. Components used in visual programming. Loop operators. Their different forms (parametric, conditional check before and after). Complex algorithms.

Topic14 Functions and modules. Standard and user-defined functions in programming languages. Local, static, dynamic variables.

Topic 15. Application in graphical and multimedia programming systems. Capabilities of the graphic module and their use. Object animation, animation options

Lecture 1

Introduction to the subject "Information technologies in technical systems". The subject of the course, its goals and objectives. Main functions and tasks of ICT in technical systems. Principles for the implementation of ICT in technical areas, modernization of digital infrastructure in order to develop the digital economy.

Plan:

1. Main tasks and directions of ICT development in modern society.
2. The main directions of the state policy of the Republic of Uzbekistan in the field of ICT development.
3. The concept of computer systems. Stages of development and types of computer systems.
4. Development of a perfect computer system.
5. Technical, software and linguistic support of computer systems and their technologies

Key terms: communication, information, technology, hardware configuration, interface, telecommunications, infrastructure, online, automation technology system, design source, technology integration, application software, graphical interface, multiprogramming, translator.

The main tasks and directions of development of ICT in modern society.

The beginning of the 21st century is characterized by informatization, globalization and technologization of all spheres of society, which determine further global trends in the development of science, education, production and management. The development of the economy of any country, the expansion of the activities of companies in various industries, the need to improve systems for storing, processing, transmitting and protecting data, the desire to increase labor productivity are factors that determine the state's needs for informatization of various spheres of social reality and the introduction of information and communication technologies.

It is generally recognized that there is no alternative to informatization based on computer technologies and means of communication in solving issues of the further development of technical civilization. In the information society, where information is an important and expensive resource, the level of a country's development is already assessed by the level of its informatization. It can be stated that today information is perceived as a product with a significant driving force potential, and information technology as a means of activating and effectively using the information resources of society, which are the most important factor in its development. Therefore, the task of qualitative improvement and intensive development of information and communication technologies, their transition from routine to innovation becomes strategically important, both at the national and global levels.

In modern conditions, the ability of each society and its institutions to collect, process, analyze, systematize and accumulate information using modern information and communication technologies is becoming a key prerequisite for social and technological progress. Along with this, as a result of the active use of information technologies and telecommunications, the concept of "information and communication technologies" (ICT) is used.

Information technology in technical systems is a science that studies the theory and methods of information processing in digital computers, the design of computer hardware and software, and the use of computer tools.

Information and communication technologies should be understood as an integral technical system, which is a combination of information technologies, telecommunications and Internet companies, and allows for a system-organized sequence of operations for the creation, processing, storage, distribution, display and use of an information product.

ICT means are understood as software, hardware and technical means and devices that operate on the basis of microprocessor computer technology, as well as modern means and systems for broadcasting information, information exchange, providing operations for the collection, accumulation, processing, storage, production, transmission, the use of information, the ability to access information resources of computer networks, including global ones. To clarify the content and essence of information and communication technologies, consider their structure. Scientists identify several classifications of segments of the information and communication technology industry, but there is no single approach.

Analyzing the definition of information and communication technologies, we consider it appropriate to consider its three main segments:

1. Information technology industry;
2. Telecommunications industry;
3. Internet companies.

Information technology industry. According to the definition adopted by UNESCO, information technology is a complex of interrelated scientific, technological, engineering disciplines that study methods for the effective organization of the work of people involved in the processing and storage of information, methods of interaction between people and computers and production equipment, their practical applications, as well as social, economic and cultural aspects of this problem. The information technology industry has the most pronounced structure and is a combination of intellectual and intermediary electronic services. It includes the field of hardware, the field of software, the field of IT services, the implementation of equipment and software. Let's consider each of the areas in more detail.

The sphere of hardware (hardware-solid products) is a set of technical means that ensure the functioning of an information system. It includes personal computers, servers, peripherals, components and storage systems.

The sphere of software is a set of software tools that, in combination with hardware, make it possible to automate the execution of a set of tasks and ensure the functioning of

electronic information resources and information systems. The scope of the software consists of:

- system software - the software basis of hardware, which acts as an intermediary between other programs and hardware, thereby organizing the process of processing information in a computer. This tool ensures the operation of other programs, providing them with service functions, manages the hardware resources of the computing system. The main component of system software is the operating system;

- tool software – software development tools. This product is intended for use in the process of designing, developing and maintaining programs. It includes database management systems, programming languages, etc. With the help of this toolkit, application software is created that end users work with.

- application software - a set of tools that are designed to solve user problems and are designed for direct interaction with him. The result of the application of application software is the automation of workplaces.

The scope of IT services is a set of information products made available to the user. Consider the main segments of this area:

1. Project-oriented services - the implementation of activities mainly in the project form, which provides for profit only through the creation of unique products (specialized software, information systems, etc.). These services include:

- consulting (project-oriented activity that is related to information support of business processes and allows you to provide an independent expert assessment of the effectiveness of using information technologies),

- system integration (activities related to the installation and adjustment of operating systems, databases, communications, data storage devices, Internet connections, etc.),

- custom software development;

2. Outsourcing-oriented services - the process of transferring internal services and internal services of the customer company to the contractor, including through the use of its software products, hardware and infrastructure elements. The main element of these services is IT-outsourcing;

3. Services focused on support and training - a set of active learning methods aimed at developing knowledge, skills and abilities in the field of IT-technologies. These services include IT-support and IT-education and training.

Hardware and software sales area is a global wholesale and retail trade network aimed at distributing software and hardware. The main goal of this network is to create the necessary conditions for information and communication technologies to become an integral attribute of every person's life.

The information technology industry is a combination of computer technology, communication tools, software for solving the problems of effective organization of the information process, which contribute to minimizing the material and labor resources of all spheres of modern society.

The telecommunications industry is a set of telecommunications operators with interconnected technological infrastructures included in the overall production process. This industry includes telecommunications services and telecommunications equipment.

The sphere of telecommunications services is a set of products of the activities of the operator or provider of telecommunications aimed at meeting the needs of consumers in the field of telecommunications. It contains:

- wired communication services - the process of transmitting and receiving information using wire lines with metal or fiber optic cores;

- wireless communication services - activities aimed at providing telecommunications using radio technologies, during which the end equipment of at least one of the consumers can move freely while maintaining a unique identification number within the termination points of the telecommunications network that are connected to one switching center;

- data transfer services - transfer of information in the form of data using telecommunication networks;

- telematic services - services for accessing and processing data regarding the control and measurement of parameters of executive or other systems, mechanisms, instruments and equipment.

The field of telecommunications equipment is a set of industries that allow the transmission of audio / video or other information, as well as establish communication between various types of devices. This tool implements telecommunications hardware. The field of telecommunications equipment includes hubs, switches, routers, adapters, cables, connectors, and more.

The telecommunications industry is a link between the industrial sector, the service sector and consumers. By stimulating human communication through communication, modern means of telecommunications become a necessary condition for the social cohesion and cultural development of countries.

3. Internet companies. Internet companies (dotcoms) are firms that carry out their activities via the Internet, that is, by providing a service or selling products through the network. They provide various services via the Internet (search, information, research, advertising, sales, and others). The functioning of these companies is directly related to information technology, but their activities have a different specificity, and therefore they occupy a separate link in the structure of information and communication technologies.

2. The main directions of the state policy of the Republic of Uzbekistan in the field of ICT development.

The introduction of modern information and communication technologies is a necessary condition for the development of any state. Uzbekistan, following the path of democratic reforms and the development of a market economy, is no exception.

The development of modern information and communication technologies (ICT) has a purposeful tendency to intensify and diversify, covering all new sectors of the economy, including the areas of government. The events of the last decade have provided a huge

amount of evidence of the real significance of ICT for the way of life of peoples and countries. Several key studies, including the UNDP Human Development Report (2001) and the Digital Opportunity Initiative (2001), identified two divergent trends:

- countries fall into an isolated and disadvantageous position if they do not respond to the tasks put forward for a new type of "knowledge society";
- countries are in a much better position if they adopt policies that promote the use and exploitation of ICT for development.

The speed of Uzbekistan's integration into the emerging world economy depends on its ability to both shed the legacy of state socialism and quickly modernize its political institutions and governance mechanisms. In this regard, information and communication technologies (ICT), as the main driving force behind institutional reform, provide great opportunities for the country's economic development.

As noted in the Uzbekistan eReadiness Assessment report, the key policy lessons from the assessment are summarized as follows:

- development goals can best be achieved with a holistic approach. The development of ICT as a multidimensional concept requires a programmatic approach to the whole range of interrelated components.

Coordination of the efforts of the international and donor community, efficient use of financial resources, development of new partnerships, which guarantees the achievement of a greater effect.

The proposed solutions should be in line with national strategies and priorities and supported by the government of the country.

The government understands that ICTs alone are not a panacea for Uzbekistan to overcome many development challenges. However, as noted in recent government decisions and legislation, creating a favorable legal and regulatory environment that promotes the development of information infrastructures and supports education and training, the application of knowledge and information can have a significant positive impact in supporting the overall development of the country. In this regard, the application of ICT for development, as the main driver of the development process in all sectors, is becoming an important priority for the country, which was recognized in the protocol of cooperation between UNDP and the Government of the Republic of Uzbekistan for the period 2000-2004.

The development of information and communication technologies (ICT), which is the most important factor in raising prosperity and economic growth, is becoming one of the main priorities of the state policy of Uzbekistan. The general vision of the development of ICT and the Internet in Uzbekistan is reflected in the speech of the President of the Republic of Uzbekistan at the session of the Parliament of the country in May 2001. In a sweeping statement, the President called on the government to develop a common ICT development strategy in support of the country's social, cultural and economic future. The President's initiative signaled major strategic changes. The government is now clearly aware of the

importance of ICT to achieve its development goals. Therefore, in recent years, the leadership of the republic has been taking vigorous measures to develop and widely introduce ICT in various areas of social and state building. A special place in this series is occupied by the Decree of the Head of State “On the further development of computerization and the introduction of information and communication technologies” dated May 30, 2002, aimed at improving the institutions for supporting ICT.

Thus, according to the decree, the Coordinating Council for the Development of Computerization and Information and Communication Technologies was created, the Uzbek Agency for Post and Telecommunications was transformed into the Uzbek Agency for Communications and Informatization (**UzACI**) with the role of the executive body of the said Coordinating Council. In addition, the Center for the Development and Implementation of Computer and Information Technologies “**UzInfoCom**” was created under **UzACI**, and the Tashkent Electrotechnical Institute of Communications was transformed into the Tashkent University of Information Technologies.

The development and approval by the Cabinet of Ministers of the Republic of Uzbekistan of the Program for the Development of Computerization and Information and Communication Technologies for 2002-2010 (Resolution of the Cabinet of Ministers dated June 6, 2002 No. 200) confirm that the state has begun to play a significant and decisive role in creating and stimulating the development of an enabling environment for the development of ICT in Uzbekistan. At the same time, effective measures have been launched to introduce ICT in the government itself. The decision on "electronic government", i.e. the transition of government agencies to the online system, and the provision of unified electronic means of communication between the public sector and the citizens of the country, will be the most convincing example that the Republic of Uzbekistan is serious in its intentions to enter the modern information age.

It should be noted that for the first time the assessment of the e-readiness position of Uzbekistan was carried out by UNDP experts back in 2001 and, in fact, reflected the state of ICT development in the country before the start of large-scale measures. The results of these studies showed that the republic is among the countries with a fairly low level of ICT development: “Uzbekistan begins to rapidly lag behind in the development of ICT, not only in comparison with developed countries, but also in comparison with many developing countries that were with it about a few years ago. at the same level of ICT development”. In this regard, there was an urgent need for current studies, the purpose of which was to identify the dynamics and trends in indicators/indicators of e-readiness, i.e. monitoring the country's ICT development over the past period of intensive transformations.

Experts note that the general positive factor was the spread of high-speed Internet access, the development of mobile communications, the implementation of relevant government programs and the adoption of new legal acts in the field of ICT. The gradual deregulation of the telecommunications market also contributed to the development of the information society. Of particular importance was the development and implementation of government

programs for the development of ICT and the strengthening of the information infrastructure.

The role of ICT in social, economic sectors and in management.

At a meeting of the Cabinet of Ministers of the Republic of Uzbekistan, dedicated to the results of the socio-economic development of the Republic in 2013 and the main priorities of the economic program for 2014, the State Committee for Communications, Informatization and Telecommunication Technologies was tasked with the timely implementation of measures and the implementation of projects identified in the Comprehensive Program development of the National Information and Communication System. The basis for the development of information and communication technologies is the telecommunications infrastructure. Due to the measures taken in this direction, the following results have already been achieved to date.

Development of telecommunications infrastructure

The current stage of development of telecommunications technologies, networks and communications infrastructure of the country is carried out by expanding fixed and mobile broadband access networks, expanding data and voice traffic switching centers, modernizing and expanding backbone telecommunications networks, as well as creating infrastructure for the development of multimedia services. To date, the total speed of using international information networks has increased by 42.3% compared to the beginning of 2014 and amounted to 15.5 Gbps. The implementation of these projects also contributes to the development of wireless communications. Today, mobile operators are rolling out fourth-generation 4G LTE networks, which will allow users to quickly and efficiently work with a large amount of information on the Internet, download and view streaming video, upload high-quality photos, and use online applications for educational and business purposes. In order to urgently respond to user requests for information and communication services in all regions of the country, 13 call centers were created as part of the project to introduce the Unified Call Center of JSC Uzbektelecom, as well as for the needs of government agencies and business entities. These measures serve to improve the quality of telephone and Internet services provided to a higher level. Local telephone networks are being gradually modernized based on next-generation technologies. During the current year, 26 units of modern switching equipment were put into operation throughout the Republic, which significantly increases the capacity of telephone numbers.

A computer is a special device for processing data. A computer system is a combination of all the components needed to process and store data using a computer. Every computer system is made up of several pieces of hardware and software.

Data, especially facts or figures, collected for study and consideration and use, in order to help decision making or information processing in electronic form, which can be stored and used on a computer and stored on a hard disk. Data, usually transmitted electronically, can be changed into pulses of light.

Hardware is the hardware, or physical devices, associated with a computer. For example, keyboards, mice, speakers, and printers are all hardware. Devices are manufactured in various ways for mainframes, small laptops, and even smaller ones built into products such as cars and thermostats. However, the kinds of operations performed by computers of different sizes are very similar. When you think of a computer, you often think of its physical components first, but for information to be useful to a computer, it needs more than a device; the computer must be provided with instructions. Just like, for example, your stereo equipment does nothing until you provide music, which it subsequently plays. Computer hardware needs instructions that determine how and when data items are entered, how they are processed, and the form in which they are output and stored.

Software describes computer instructions that tell the hardware what to do. Software is programs, which are sets of instructions written by programmers. You can buy programs written by others that are stored on disk, or you can download them from the Internet. For example, word processing and accounting software businesses and casual computer users use programs that are designed for music and games. Also, you can write your own programs. Writing software instructions is programming.

Application software includes all programs that can be applied to task programs—word processing, spreadsheets, payroll and inventory programs, and even games.

System software includes programs that are used to manage a computer, including operating systems such as Windows, Linux, or UNIX.

Together, hardware and software perform three basic operations in most programs:

Data input - data enters the computer system and is placed in the computer's memory, which is the temporary, internal memory of the connected computer. Hardware devices that perform input operations include the keyboard and mouse. Data elements include all text, numbers, etc. that are entered into and processed by a computer. In business, many of the data items are used as facts and figures about individuals, products, customers, and personnel. However, the data may also include items such as images, sounds, and user mouse movements.

Processing - elements of data processing may include arranging to sort them, checking them for accuracy, or performing calculations on them. Processing converts input data that has been stored in memory into information suitable for output. The hardware component that performs these types of tasks is the CPU.

Output - after the data items have been processed, the resulting information is typically sent to a printer, monitor, or other output device so that people can view, interpret, and use the results. Programmers often use the term data for input elements and the term information for data that has been processed and output. Sometimes the output destination can be storage devices such as disks or flash drives. Humans cannot read data directly from these data storage devices, but information storage devices are required for later retrieval. The data sent to the storage device as output is sometimes used later as input to another program.

Classification of computers.

Traditionally, the following types can be cited in the classification of computers:

Supercomputers: the fastest and most expensive computers. These huge computers are used to solve very complex scientific and technical problems. Supercomputers get their computing power through the use of parallel data processing; they use many processors at the same time on the same task. A typical supercomputer can make up to ten trillion individual calculations every second.

Examples: Supercomputers: Jaguar, K Computer, Colombia

Server computers: are a sub-supercomputing step because they are not focused on trying to solve one very difficult problem, but trying to solve many similar small ones. An example of a server would be computers, which is stored in the Wikipedia encyclopedia. These computers must go and find the page you are looking for and send it to you. This in itself is not a big task, but it becomes a job for the server when the computers have to go and find a lot of pages for a lot of people and direct them to the right place. Some servers, like the ones Google uses something like Google docs, have apps on them, not just files like Wikipedia.

The server is a central computer that contains sets of data and programs. Also called a network server, this system allows all connected users to store and share electronic data and applications. Two important types of servers are file servers and application servers.

Workstations: high-end, expensive computers that are made for more complex procedures and designed for one user at a time. Some of the complex procedures consist of natural sciences, mathematics, and engineering calculations and can be used for computer-aided design and manufacturing.

Personal computers. PC is an abbreviation for personal computer, it is also known as a microcomputer. Its physical characteristics and low cost are both attractive and beneficial to its users. The capabilities of the personal computer have changed significantly with the advent of electronic computers. By the early 1970s, people in academic or research institutions had the ability for one person to use a computer system interactively for extended durations, although these systems would still be too expensive to be owned by one person. The introduction of the microprocessor, a single chip with all the circuits that previously occupied large cabinets, led to the proliferation of personal computers after about 1975. hobbyists and technicians.

By the late 1970s, mass-market, pre-assembled computers allowed a wider range of people to use computers that were more focused on software applications and less on hardware processor development. Throughout the 1970s and 1980s, home computers were designed for home use, offering some personal productivity, programming, and gaming, while several larger, more expensive systems (though still low cost compared to minis) - computers and high-powered computers) were aimed at office and small business use,

Today, the personal computer is a comprehensive device that can be used as a productivity tool, media server, and gaming machine. The modular design of the personal computer allows components to be easily replaced when broken or upgraded.

Minicomputers (microcontrollers) that allow the user to store data and execute simple commands and tasks. These single circuit devices have minimal memory and program length, but are generally designed for low level tasks.

Types of computer systems.

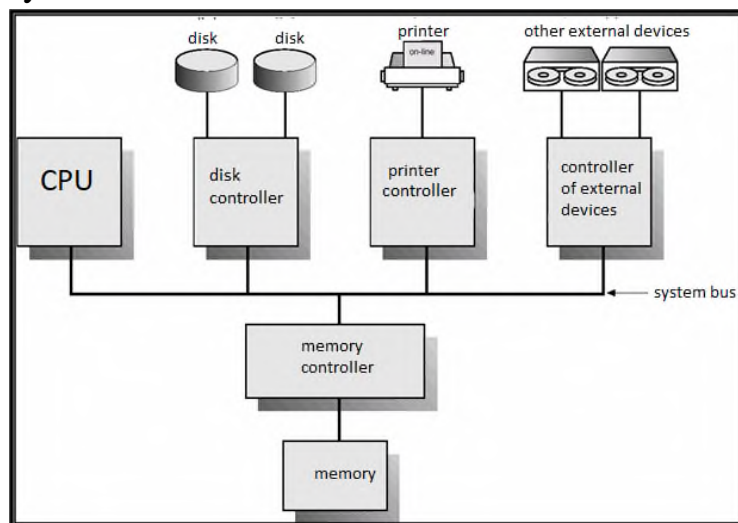


Fig. 1. Computer system architecture.

The computer system has a modular structure. For each device (memory, external devices) in the system there is a special control device (in other words, a special processor), called the device controller. All modules (central processor, memory and memory controller, external devices and their controllers) are interconnected by a system bus, through which they exchange signals. As we already know, the operation of each controller is controlled by a driver - a specialized low-level program that is part of the OS.

Consider the typical structure of a modern desktop or laptop computer system, indicating the most common types of devices and their characteristics.

The central processing unit is a device that executes commands (instructions) of a computer system. In modern computers, as a rule, it is multi-core, i.e. has in its composition from 2 to 32 cores (copies) of the processor, running in parallel on a shared memory, or hybrid, consisting of a central and graphic processors. The performance of each core is 3 - 3.2 GHz. Note that in this case, performance is understood as the clock frequency of the processor (core) - the time it takes to execute one of the simplest machine instructions. However, there are other important factors that determine the overall performance of the system - the clock speed of the memory and the system bus. In fact, the overall system performance can be measured by the slowest of these parts of the system (usually the system bus). These characteristics must be taken into account when choosing and buying a computer.

Random access (main) memory, or simply memory, is a device that stores processed data. Memory capacity - 1 - 16 *gigabytes* or more; using less memory is not recommended, as it can lead to a significant slowdown of the system. *Memory clock frequency* - 667 MHz - 1.5 GHz.

The system bus is a device to which all computer modules are connected and through which they exchange signals, for example, about interrupts. The bus clock *frequency* is 1 - 1.5 GHz (this is actually a certain total system performance). Typically, a PCI (Peripheral Component Interconnect) bus is used. A processor, memory, disks, a printer, a modem, and other external devices can be connected to it.

Ports are devices with connectors for connecting external devices to the computer. Each port has its own controller (and, accordingly, its own driver).

The most commonly used port is USB (Universal Serial Bus), with a characteristic flat connector, about 1 cm in size, with the image of a trident. Most types of devices can be connected to USB ports, and for this it is not necessary to first turn off the computer and the connected device, which is very convenient. There are several USB standards with different speeds. The USB 2.0 standard is now the most common, providing a port speed of 240 - 260 megabits per second. For comparison, the previous standard - USB 1.0 - provided only 10 - 12 megabits per second (feel the difference, as they say). You can recognize the type of USB port on your computer if you display information about devices; on Windows: My Computer / (right click) Properties / Hardware / Device Manager / USB Devices. In this case, the USB 2.0 port controller will be labeled as enhanced. If this is not the case, you need to upgrade the USB ports or the computer itself, otherwise you will have to wait 20 times longer when transferring to a flash drive (!). There are also "adapters" USB 1.0 -> USB 2.0. The latest USB 3.0 standard, which has just begun, will provide at least 1 gigabit per second of speed. You can connect a keyboard, mouse, printers, scanners, external hard drives, flash drives and even TV tuners to the USB port - devices for receiving a television signal from an antenna and displaying a television image on a computer. It is recommended that each device is always connected to the same USB port, otherwise some devices (for example, the same TV tuner) may experience problems.

Functioning of the computer system

The advantage of the described modular hardware approach is that the CPU, memory, and external devices can operate in parallel. Each device is controlled by a dedicated controller. If it is necessary to perform input-output, the central processor generates an interrupt, as a result of which the operating system is called, which in turn, as a response to the interrupt, launches the device driver, respectively, activating its controller. Each device controller has a local buffer, a specialized memory for exchanging information between the computer and the device. In order for the controller to start output to the device, the central processor (more precisely, the device driver running on it) must first transfer information from the specified area to the device buffer. Further, the device controller already performs the output of information from the buffer to the device itself (for example, writes it to a specified area of the hard disk). At the end of the exchange of information, the controller generates an interrupt signal (interrupt) on the system bus, thereby informing the processor about the end of the operation. In order to avoid repeated transfers of large amounts of information, modern computers use DMA (Direct Memory Access) - controllers -

controllers with direct access to RAM. Such controllers do not use their specialized memory when exchanging with the device, but directly the area of RAM, in which the clipboard is located.

Synchronous input-output(I/O) is input-output that causes the program to enter a wait state until the input-output operation is fully completed. At the hardware level, an input-output instruction puts the processor in the idle state until the next interrupt. In this mode, no more than one input-output request is executed at a time; there is no simultaneous input-output. Synchronous output is performed by all programmers with the usual `println(x)` statements. When using them in programs, we do not think that we are using a rather inefficient version of synchronous input-output. However, until now, the thinking of most programmers is sequential, in the sense that they think of their program as purely sequentially executed, and do not think at all about the possibility of any parallelization. When debugging a program, or if the output size is small, this is usually fine.

Asynchronous input-output is input-output that is performed in parallel with the execution of the main program. After, as asynchronous input-output begins, control returns to the user program, without waiting for the input-output to complete (the latter can be done with a special explicit operation). Thus, the operation of an asynchronous exchange, as it were, is divided into two: start input-output and finish input-output. The latter is performed so that at this point the program still waits for the completion of the exchange, when its result is necessary for further calculations. This approach to the implementation of the exchange is more difficult for programmers to understand and can lead to errors.

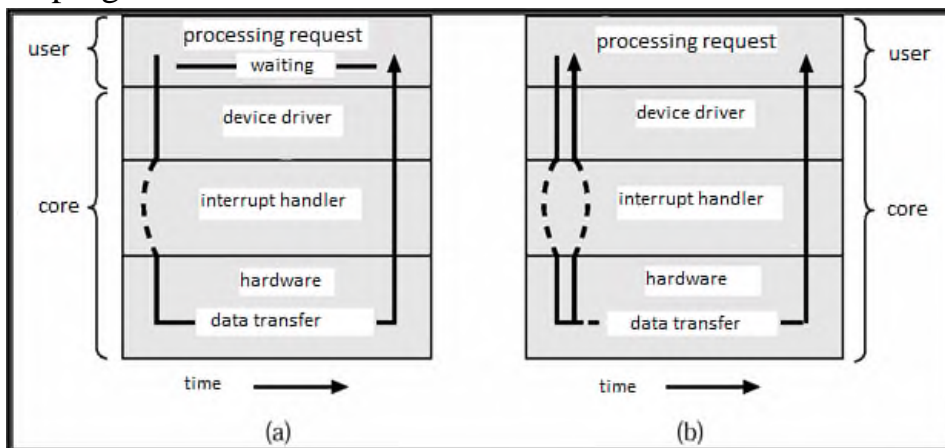


Fig.2. Synchronous and asynchronous input-output architecture.

At the system level, the following occurs during the exchange. A system call is made - a request to the OS by calling a system subroutine, in this case - to allow the user to wait for I/O to complete. The operating system maintains a device state table, in which each device has an element containing the device type, address, and state. The OS indexes the device table to determine the state of the device and modify the table entry to include interrupt information.

The architecture of synchronous (a) and asynchronous (b) I/O is illustrated in Figure 2.

The diagram shows that the hallmark of a synchronous exchange is the transition of the processor to a waiting state before the end of the I / O operation.

Direct Memory Access (DMA) is a more efficient method of operating I/O device controllers, used to operate high-speed devices that can transfer information at a speed close to the speed of memory

The DMA controller transfers a block of data from the buffer memory directly to the main memory, without the participation of the processor. The advantage of such a widely used approach is not only to avoid unnecessary data transfers from one memory area to another, but also that an interrupt in this case is generated for each block of data transferred (stored in the buffer), but not for each transferred bytes, as with a more traditional method of exchange.

A package of applied programs and their classification.

We give the following classification of applied software:

Text editors. The main functions of this class of application programs are to enter and edit text data. Additional features include automating input and editing processes.

Word processors. The main difference between word processors and text editors is that they allow not only to enter and edit text, but also to format it, that is, to format it.

Graphic editor. This is an extensive class of programs designed to create and (or) process graphic images. In this class, the following categories are distinguished: raster editors, vector editors, and software tools for creating and processing three-dimensional graphics (3D editors).

Database management systems. Databases are huge amounts of data organized in tabular structures. The main functions of the database management system are:

- creating an empty (blank) database structure;
- providing means of filling it or importing data from tables in another database;
- providing the ability to access data, as well as providing a search and filtering environment.

e-Spreadsheets. Spreadsheets provide comprehensive tools for storing various types of data and processing them. To some extent, they are similar to database management systems, but the main focus of shifts is not on storing data arrays and providing access to them, but on data transformation, moreover, in accordance with their internal content.

Computer-aided design systems (CAD-systems). Are intended for automation of design and engineering works. They are used in mechanical engineering, instrument making, architecture. In addition to drawing and graphic work, these systems make it possible to carry out simple calculations (for example, calculations of the strength of parts) and the selection of finished structural elements from extensive databases.

Desktop publishing systems. The purpose of this class of programs is to automate the process of imposition of printing publications. This class of software occupies an intermediate position between word processors and computer-aided design systems.

Expert systems. Designed to analyze the data contained in the knowledge and issue recommendations at the request of the user. Such systems are in cases where the initial data are well formalized, but extensive special knowledge is required to make a decision. Typical

areas of use of expert systems are jurisprudence, medicine, pharmacology, chemistry. Based on the totality of signs of the disease, medical expert systems help to establish a diagnosis and prescribe medications, dosages and a treatment course program. Based on the combination of signs of an event, legal experts can give a legal assessment and suggest a course of action for both the accusing party and the defending party.

HTML editors (Web editors). This is a special class of editors that combines the properties of text and graphic editors. They are designed to create and edit so-called Web documents {Web pages. In Web documents are electronic documents, the preparation of which should take into account a number of features related to the reception and / transmission of information on the Internet.

Browsers (browsers, Web viewers). This category includes software tools designed to view electronic documents made in HTML format (documents of this format are used as Web documents). Modern browsers render more than just text and graphics. They can play music, human speech, listen to radio broadcasts on the Internet, watch video conferences, work with e-mail services, with a teleconferencing system (news groups), and much more.

Integrated business systems. They are software tools for automating the workplace of the head. The main functions of such systems include the functions of creating, editing and formatting the simplest documents, centralizing the functions of e-mail, facsimile and telephone communications, dispatching and monitoring the enterprise's workflow, coordinating the activities of departments, optimizing administrative and economic activities and supplying on-demand operational and reference information.

Test questions:

1. What do you understand by the term "computer system".
2. What areas of memory are used in input-output operations?
3. Types of computer systems.
4. How the computer system is organized.
5. What is the difference between batch system, interactive system and real time system?
6. What do you understand by information technology?
7. What do you understand by information and communication technologies?
8. ICT in modern society?
9. Industries of ICT application?

Lecture 2

Modern computer-aided design systems and their use in technical systems (CAD/CAE/CAM-systems). CAD in the field of mechanics. Classification of CAD by intended purpose used in computer-aided design.

Plan:

1. Design automation systems and their functions.

2. The main stages of the design automation process.
3. Design automation systems. Their types and models.
4. CAD/CAM/CAE systems

Key terms: *computer-aided design systems (CAD), design, technological processes, modeling, subsystems, engineering calculation system, design subsystem, service subsystem, technical support, mathematical support, information support, linguistic support, methodological support.*

Automatic design systems (CAD). Design automation occupies a special place among information technologies.

Design is the process of compiling a description necessary to create an object that does not yet exist or an algorithm for its functioning under given conditions, based on the primary description of this object and (or) the algorithm for its functioning.

Design objects can be:

- products for industrial and technical purposes (means of production - technological equipment and tooling);
- technological processes, as a result of the implementation of which projects of objects are embodied in a material form;
- buildings, engineering structures; vehicles, means of communication, computer technology; organizational and management systems, etc.

The purpose of the design process is, first of all, to develop technical documentation for the manufacture of the design object based on the initial information obtained in the design process. Design includes the development of terms of reference (TOR), reflecting the needs, and the implementation of the TOR in the form of project documentation.

Design is essentially a feedback control process. The technical task generates inputs that are compared with the design results, and if they do not match, the design cycle is repeated again until the deviation from the specified technical requirements is within acceptable limits.

Design automation is understood as the systematic use of computers in the design process with a scientifically substantiated distribution of functions between the designer and the computer and a scientifically substantiated choice of methods for machine problem solving.

A scientifically substantiated distribution of functions between a person and a computer implies that a person must solve problems of a creative nature, and a computer must solve problems whose solution can be algorithmized.

Computer-aided design is a design in which individual transformations of object descriptions and (or) the algorithm of its functioning or process algorithm, as well as presentation of descriptions in various languages, are carried out by the interaction of a person and a computer. Computer-aided design is usually carried out in the mode of human-machine dialogue based on the use of special languages for human-machine communication.

Automatic design is a design in which all transformations of object descriptions and (or) the algorithm of its functioning or process algorithm, as well as the presentation of descriptions in various languages, are carried out without human intervention.

With automatic design, the start of the corresponding equipment and the input into the computer of the primary description of the object is carried out by a person.

Problems of automatic design:

1. Reduction of labor intensity and terms of design preparation of production.
2. Improving the quality of design documentation.
3. Reduction of labor intensity and terms of technological preparation of production.
4. Improving the quality of the developed technological processes.
5. Reducing the number of engineering and technical workers involved in design and construction.

A significant difference between computer-aided design and non-automated design is the ability to replace expensive and time-consuming physical modeling with mathematical modeling. Solving the problems of design automation with the help of a computer is based on a systematic approach, i.e. on the creation and implementation of CAD systems - automatic design systems for technical objects that solve the whole range of tasks from task analysis to the development of a full scope of design and technological documentation. This is achieved by combining modern technical means and software, the parameters and characteristics of which are selected with maximum consideration for the specifics of the tasks of the design process.

A computer-aided design system (CAD) is a set of design automation tools interconnected with the necessary departments of a design organization or a team of specialists (system user) that performs computer-aided design.

Automatic design system - a set of design automation tools interconnected with the necessary departments of the design organization or a team of specialists (system user) that performs automatic design.

An integrated automatic design system is an automatic design system that has alternative software and an automatic design operating system that allows you to select a set of computer programs in relation to a given design object or class of design objects.

2. Types of CAD subsystems: design and maintenance

The constituent structural parts of **CAD** are subsystems that have all the properties of systems and are created as independent systems. Each **subsystem** is a part of the **CAD** system selected according to some characteristics, which ensures the execution of some functionally completed sequences of design tasks with the receipt of appropriate design solutions and design documents.

Design decision - an intermediate or final description of the design object, necessary and sufficient for consideration and determination of the further direction or completion of the design.

Design document - a document made according to a given form, in which any design solution obtained during the design is presented.

According to the purpose of the CAD subsystem, they are divided into two types: design and maintenance.

1. Designing subsystems - object-oriented subsystems that implement a certain stage of design or a group of related design tasks. Depending on the relationship to the design object, they are divided into **object** and **invariant**.

Object (object-oriented) - performing design procedures and operations directly related to a specific type of design objects. (For example: a subsystem for designing technological systems; a subsystem for modeling a designed structure, etc.)

Invariant (object-independent) - performing unified design procedures and operations that make sense for many types of design objects. (For example: a subsystem for calculating machine parts; a subsystem for calculating cutting conditions; a subsystem for calculating technical and economic indicators, etc.)

Examples of design subsystems:

- *Functional-logical design subsystem*

At the output of this system, we get a functional circuit, followed by a logical circuit, at the output, a fundamentally electrical circuit.

- *Design engineering subsystem*

At the output, we obtain the design of the device and design documentation, including the arrangement of elements on the surface of the module and the topology of printed connections between the elements.

- *Subsystem of technological preparation of production*

At the output, we get a route map of the production process and programs for controlling machine tools with numerical control (for controlling technological equipment).

Machine layout subsystem;

Assembly unit design subsystem;

Parts design subsystem;

Control circuit design subsystem, etc.

2. Service subsystems - object-independent subsystems that implement functions common to subsystems or **CAD** as a whole, ensure the functioning of design subsystems, design, transfer and output of data, software maintenance, etc. Their combination is called the **system environment (or shell)**.

Examples of service subsystems:

- subsystem of graphic display of design objects;
- training subsystems for the development by users of technologies implemented in CAD;
- documentation subsystem;
- subsystem of strength calculations;
- information retrieval subsystem
- subsystems of project data management (PDM)

- graphics input-output subsystems.

The composition of both design and maintenance systems of modern CAD may include: expert systems. These are systems based on a knowledge base, presented either in the form of a product system or in the form of frames (FRAME). The expert system allows to formalize the knowledge of an expert in a particular subject area in order to make rational design decisions.

Decision making systems. These are systems that allow the selection of effective design solutions in the conditions of certainty and uncertainty of the initial information based on formal methods and procedures. Neural network technologies can also be used to evaluate design solutions.

Decision support systems. The design process is implemented in subsystems in the form of a certain sequence of design procedures and operations.

The design procedure is a formalized set of actions, the implementation of which ends with a design decision. Design procedures are, for example, forecasting, optimization, verification of the reality of implementation, control, correction, modeling, procedures for developing a kinematic or layout diagram of a machine tool, technology for processing products, etc. The design procedure consists of elementary design operations, has a firmly established order of their implementation and is aimed at achieving a local goal in the design process.

A design operation is an action or a formalized set of actions that are part of a design procedure (or an elementary action), the algorithm of which remains unchanged for a number of design procedures. Design operations are, for example, calculation, drawing, data tables, data input and output, calculation of allowances, solution of an equation, etc.

Unified design procedure (the term “Unified procedure” is allowed) is a design procedure, the algorithm of which remains unchanged for different design objects or different design stages of the same object.

Diagram of the computer-aided design process.

The most important issue in the creation of CAD after the formalization of the design process is the question of displaying the design activities of an engineer in software.

In general, the design process in CAD can be simplified by the diagram shown in Fig. 1. This scheme displays the elementary cell of the design process, from the chain that the real automated process consists of. All design systems created with the help of modern computer technology are automated. The most important role in these systems is played by a human engineer who develops a project of new technical means. A person in CAD solves all non-formalized design tasks and work planning tasks. Modern CAD is a tool of a highly qualified design engineer, therefore, close interaction between a person and a computer in the design process is one of the most important principles for the construction and operation of CAD.

The main block in the diagram of the computer-aided design process (Fig. 1) is the block of design decisions. Depending on the completeness of the formalization of our knowledge in a particular subject area, the design decision can be made automatically or interactively.

Based on the input data and constraints (independent design parameters), the block changes the variable parameters (decision factors) until acceptable design solutions (dependent variables) are obtained.

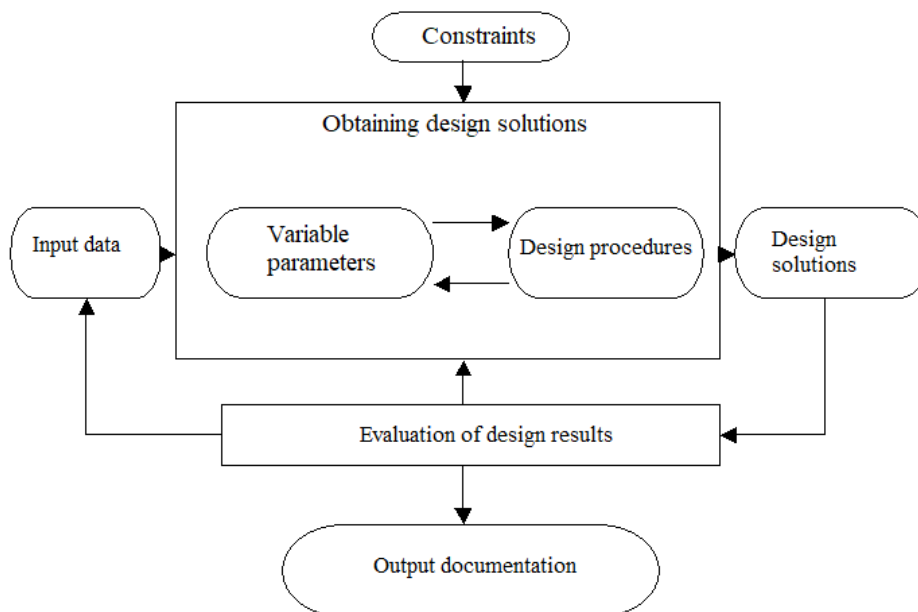


Fig. 1. Computer-aided design process system

The design results should be presented in a human-readable form and contain information on the basis of which the engineer could make a judgment about the design results. Consideration of even a simplified diagram of the design process makes it possible to clarify the division of functions between the engineer and the computer in CAD. Obtaining options for design solutions and their presentation in a form convenient for human perception can be entrusted to a computer to the extent that this will allow the mathematical support of design procedures. With the automatic receipt of options for design solutions, the most important functions remain for the engineer - input of initial data for design, final assessment and approval of design solutions. In the interactive design mode, the engineer directly participates in the course of solving problems, influencing the choice of decision factors and refining independent variables. Obtaining output documentation in accordance with existing requirements is a routine operation and should be performed automatically.

Design procedures in the automatic design system.

The software model of the automatic design procedure can be represented by the diagram shown in Fig. 2.

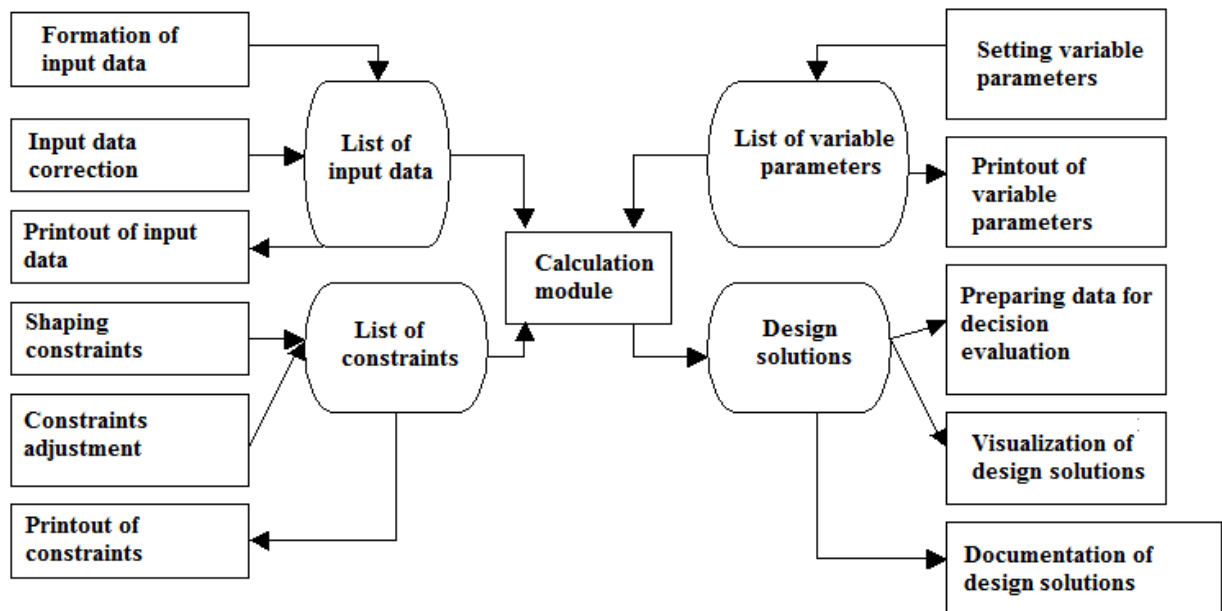


Fig. 2. Software design procedure model in CAD

The purpose of the input data generation module is to create a list of these data for design and its control when entering into the system. The structure and format of the list of input data depend on the content of the design procedure (calculation module). It is necessary to provide for the existence of several versions of the list of input data, which are stored with given names on sections of the magnetic disk. The structure of the data list is determined by the CAD developer, and it is formed either in the dialog mode by the user, or is generated automatically by the previous design procedures. The software module for correcting input data provides for editing (deleting, inserting, etc.) the list, the need for which arises due to user errors when entering data detected during control, as well as, if necessary, their clarification as a result of analysis and evaluation of design solutions.

To ensure careful control in CAD, software tools for visualizing lists of data should be provided. In general, it is necessary to be able to obtain several types of data list printouts: binary, decimal, symbolic, tabular, and by records. To meet various user requirements, the printout can be displayed on the display screen or on the ADCP. All these operations are performed by the input data printing module. The software modules for the formation, adjustment and printing of the restrictions on the design process function similarly to those described. The structure and format of constraints depend on the design module, but they are significantly less subject to change than the structure and format of the source data.

However, it is necessary to provide for the existence of several versions of these lists (for example, general requirements for technical means on the part of various customers). The creation and control of the list of variable parameters is carried out by the software modules for their assignment and printing. The calculation module of the software for the design process is designed to automatically perform all those operations of the design procedure that have been fully formalized by the computer. The received variants of design solutions are processed by the software module for preparing data for evaluating solutions and transferred to the visualization module.

Analyzing the results of the design process, the engineer must be able to view the output data on the analog-to-digital converter, display and plotter, for example, in the form of tables, diagrams and drawings. It is possible to have several versions of design decisions that are stored on a magnetic disk and can be presented in the required form using the software module for documenting design decisions. Communication between the various software modules of the design procedure and the interaction of this design procedure with others occurs through a shared memory. This allows for an interactive, automated design process with many different versions of both input data and design decisions. To fulfill the requirement of the principle of rational connection of CAD with the environment in software design, one should strive to ensure that the list of input data is the result of previous design procedures or modules. This is achieved by developing CAD information support.

Information support of automatic design systems.

The basis of CAD is a combination of various types of automatic design software necessary to solve design problems.

1. Hardware provision (HP) - a set of related and interacting technical means that ensure the operation of CAD, including various hardware (computers, peripherals, network equipment, communication lines, measuring tools).

2. Mathematical software (MS), which combines mathematical methods, models and algorithms used to solve computer-aided design problems. MS by purpose and methods of implementation are divided into two parts:

- mathematical methods
- formalized description of computer-aided design technology

3. Software provision (SP), represented by computer programs necessary for the implementation of the design process. CAD software is divided into system-wide and applied:

- General system software is designed to manage hardware components and ensure the functioning of application programs. An example of a common system software component is an operating system.

- application software implements software for direct execution of design procedures, includes software packages of application programs designed to serve certain design stages or groups of similar tasks within different stages (pipeline design module, circuit modeling package, CAD geometric solver).

4. Information support (IS) - a set of information necessary for designing, consists of a description of standard design procedures, standard design solutions, components and their models, rules and design standards. The main part of IO CAD is databases and database management systems.

5. Linguistic support (LS) - a set of languages used in CAD to provide information about the designed objects, process and design tools, as well as to carry out a designer-computer dialogue and data exchange between CAD hardware, includes terms, definitions, rules for formalizing the natural language, compression and expansion methods. In LO, a class of

various types of design and modeling languages (VHDL, VERILOG, UML, GPSS) is distinguished.

6. Methodological support (MetS) - a description of the technology of CAD functioning, methods for choosing and applying technological methods by users to obtain specific results, including the theory of processes occurring in the designed objects, methods of analysis, synthesis of systems and their components, various design methods, sometimes MS and LS are also referred to MetS.

7. Organizational support (OS) - a set of documents that determine the composition of the design organization, communication between departments, the organizational structure of the object and the automation system, activities in the conditions of the system functioning, the form of presentation of design results. The OS includes staffing tables, job descriptions, operating rules, orders, regulations, etc.

A set of data that can potentially be used in the operation of CAD or serve as a memorized result of its work form an information database (DB) of the system. Typical data groups for computer-aided design information support are classifiers and correspondence tables for them, scientific, technical and design (operational) information.

Information support of CAD can be represented as a diagram (Fig. 3), which shows what place the database occupies, and what is the interaction of the information system with design modules. This interaction is carried out through a specially organized interface that protects the design software modules from the influence of the specifics of the software implementation of the information system, thereby maintaining the independence of design operations from the type of information representation in the database. The functions of this interface also include the harmonization and interfacing of the information system and project modules in terms of record formats (information aspect), data designations (content aspect), and software tools, programming languages, etc. (program aspect).

The main requirements for information support of CAD are as follows:

Availability of the necessary information to support both automated and manual design processes. The ability to store and search for information representing the result of manual and automated design processes.

Sufficient amount of information storage. The structure of the system should allow the possibility of increasing the memory capacity along with the growth of the amount of information to be stored. At the same time, it is necessary to ensure the compactness of the stored information and the minimum wear of information carriers.

Sufficient performance of the information support system.

The ability to quickly make changes and correct information, bring these changes to the consumer, as well as obtain a hard copy of the document.

Organization of information flows in the automatic design system.

The information used in the design can be divided into static and dynamic (Fig. 3).

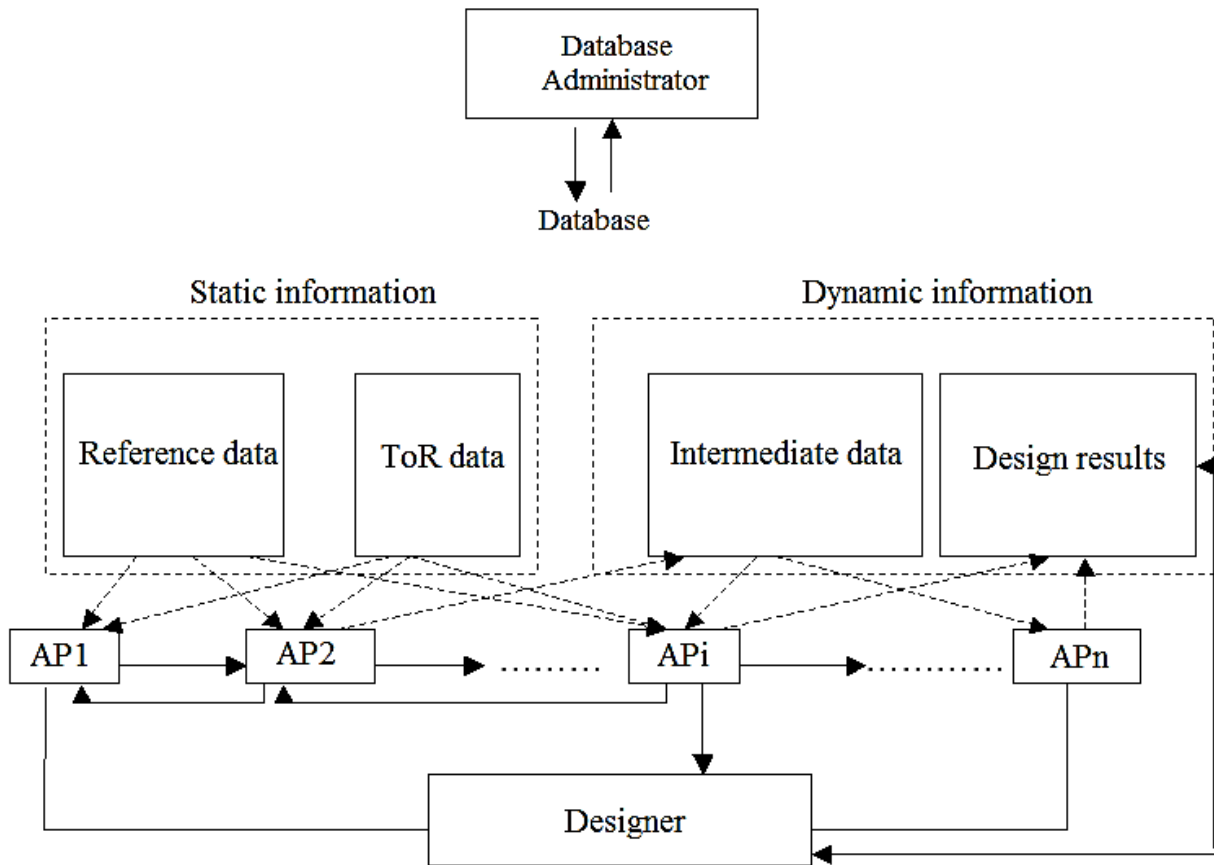


Fig.3 Scheme of information flows in CAD

Static information is characterized by relatively rare changes. This information should include the data of the TOR (terms of reference) for the design and reference data, which have a large volume. The formation, loading and correction of reference data is carried out exclusively by the database administrator, i.e. the system programmer forming the database. The database administrator maintains direct contact with the normalization and standardization service of the design organization. The volume of TOR data for the projected object is much less than the amount of reference data, but the circle of persons entitled to make changes to the TOR should be even more limited than the circle of persons entitled to correct reference data. Intermediate data is constantly changing during the operation of the CAD system. Only the executing designer and his manager have the right to make changes to the variants of design solutions.

Dynamic information becomes static in the case when the CAD is programmed to constantly replenish its database, due to newly occurring design cases (creating precedents).

The information used in the design, according to the type of its presentation, can be divided into documentary, iconographic and factographic. Documentary information is meta-information. It is a search image of a document in the database. If necessary, a set of documents that satisfy the search image can be issued. In CAD, information of this type is widely used to find information about analogues of the design object, about patents and copyright certificates, design and calculation methods, test results, etc.

The information contained in the images of documents (drawings, photographs, etc.) in an identical form of presentation is called iconographic. For its storage, special media are used (microfiches, roll microfilms, magnetic tapes of video recordings, etc.). In modern CAD, this type of information is used to store large amounts of graphic information, the search for which can be carried out using the accompanying documentary information.

General classification of CAD/CAM/CAE systems.

Over the almost 30-year period of existence of CAD / CAM / CAE systems, their generally accepted international classification has developed:

- drawing-oriented systems, which appeared first in the 70s. (and successfully used in some cases so far);
- systems that allow you to create a three-dimensional electronic model of an object, which makes it possible to solve the problems of its modeling up to the moment of manufacture;
- systems that support the concept of a complete electronic description of the object (EPD, Electronic Product Definition). EPD is a technology that enables the development and maintenance of an electronic information model throughout the entire life cycle of a product, including marketing, conceptual and detailed design, process preparation, production, operation, repair, and disposal. As a result of the development of the EPD concept, there were grounds for turning standalone CAD, CAM and CAE systems into integrated CAD/CAM/CAE systems. CAD (CAD System - Computer Aided Design System) is a system that implements design, in which all design solutions or part of them are obtained as a result of calculation and compilation of mathematical models on a computer. The main function is the implementation of computer-aided design at all and individual stages of designing objects and their components.

The purpose of using CAD is to increase the efficiency of the work of engineers, which includes:

- Reducing the complexity of design;
- Reduction of design time and clarification of the planning stage;
- Reducing the cost of design and manufacture;
- Improving the quality and technical and economic level of design results;
- Reducing the cost of modeling and testing;
- Reducing the number of marriages.

At the moment, there are three main subgroups of CAD:

- engineering CAD (MCAD - Mechanical Computer Aided Design). The term "CAD for mechanical engineering" in our country usually refers to packages that perform the functions of CAD / CAM / CAE / PDM, i.e. computer-aided design, pre-production and design, and engineering data management.

- Architectural and construction CAD (CAD / AEC - Architectural, Engineering, and Construction).

- PCB CAD (ECAD - Electronic CAD/EDA - Electronic Design Automation).

CAD - computer aided design General term for all aspects of computer aided design. Usually covers the creation of geometric models of the product. As well as the generation of drawing products and their accompaniments.

CAM - Computer Aided Manufacturing - A general term for an automated production preparation system, a general term for a PS for preparing information for CNC machines. Traditionally, the input data for such systems were geometric models of parts obtained from CAD systems.

CAE - Computer Aided Engineering - Automatic project analysis system. A general term for information support of the conditions for automated design analysis, aimed at error detection (strength calculations) or optimization of production capabilities.

PDM - Product Data Management - Production Information Management System. A tool that helps administrators, engineers, designers manage both data and product development processes in modern manufacturing plants or a group of related enterprises.

Test questions:

1. What is CAD?
2. What are the design stages?
3. Types and models of CAD?
4. What automation tools for scientific projects do you know?

Lecture 3

Modeling. The main types of modeling, their scope. Computer modelling. Classification of models. The principle of computer simulation. Expert systems

Plan:

- 1 Modeling. Main types of modeling
2. Computer simulation.
- 3 Expert systems

***Key terms:** modeling, model, computer simulation, modeling process, expert systems*

Model - analogue, prototype, template, sample used instead of the original to solve problems (get answers to questions). The model is built on the basis of a limited set of data (properties, behaviors) known to us about the original. The construction of models and the use of models (solving problems on them) is carried out in order to:

- obtaining previously unknown data, predicting new properties and future behaviors,
- benefiting from the implementation of solutions,
- systematization (generalization) of known data.

Modeling is a method, the process of replacing the original with its analogue (model) with the subsequent study of the properties and behavior of the original on the model.

The modeling process consists of:

- formalizations (designing and setting up a model, systems of models and models of systems),
- modeling itself (setting various problems and solving them on the model),
- interpretation of modeling results, integration with existing real systems.

The model instead of the original object is used in cases where the experiment is dangerous, expensive, takes place on an inconvenient scale of space and time (long-term, too short-term, extended ...), impossible, unique, invisible, etc. Let's consider this with an example:

- "experiment is dangerous" - when operating in an aggressive environment, it is better to use its layout instead of a person; an example is the lunar rover;
- "expensive" - before using the idea in the real economy of the country, it is better to test it on a mathematical or simulation model of the economy, having calculated all the "pros" and "cons" on it and getting an idea of the possible consequences;
- "long-term" - to study corrosion - a process that takes decades - more profitable and faster on the model;
- "short-term" - it is better to study the details of the process of processing metals by explosion on a model, since such a process is transient in time;
- "extended in space" — mathematical models are convenient for studying cosmogonic processes, since real flights to the stars are (yet) impossible;
- "microscopic" - to study the interaction of atoms, it is convenient to use their model;
- "impossible" - often a person deals with a situation where the object does not exist, it is still being designed. When designing, it is important not only to imagine the future object, but also to test its virtual counterpart before design defects appear in the original.

Modeling is closely related to design. Usually, the system is first designed, then it is tested, then the design is corrected again and tested again, and so on until the design meets the requirements for it. The design-modeling process is cyclical. At the same time, the cycle looks like a spiral - with each repetition, the project gets better, as the model becomes more detailed, and the level of description is more accurate. The model allows you to decompose the system into elements, connections, mechanisms, requires explaining the operation of the system, determining the causes of phenomena, the nature of the interaction of the components.

The modeling process is the process of transition from the real area to the virtual (model) one by means of formalization, then the model is studied (modeling itself) and, finally, the results are interpreted as a reverse transition from the virtual area to the real one. In the simplest case, modeling technology involves 3 stages: formalization, actual modeling, interpretation (Fig. 1.1).

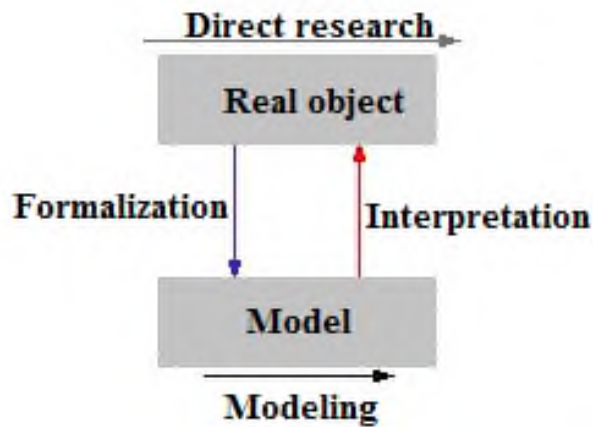


Fig. 1.1. Modeling process (basic version)

Model classification

Classification is the division of objects into groups that have one or more common features. Depending on the classification feature, the same models can be assigned to different classes.

Classification according to the area of use of the model is shown in Fig. 1.2.

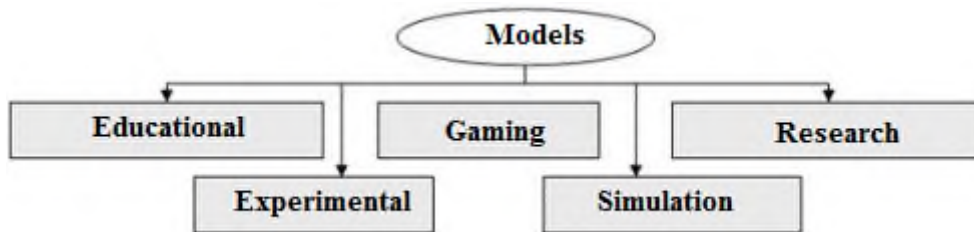


Fig. 1.2. Classification of models by area of use

Training models - visual aids, simulators, training programs.

Game models are economic, military, business games. They rehearse the object's behavior in different situations.

Research models are created to study processes or phenomena, for example, stands for testing electronic equipment.

Experimental models are reduced or enlarged copies of objects. They are used to study an object and predict its future characteristics (for example, an experimental model of a designed car).

Simulation models imitate reality, while, as a rule, the experiment is repeated many times

2. Computer simulation.

Computer simulation is one of the effective methods for studying complex systems. Computer models are easier and more convenient to study due to their ability to conduct computational experiments in cases where real experiments are difficult due to financial or physical obstacles or can give unpredictable results. The logicity and formality of computer models makes it possible to identify the main factors that determine the properties of the original object under study (or a whole class of objects), in particular, to investigate the response of the simulated physical system to changes in its parameters and initial conditions.

The construction of a computer model is based on abstraction from the specific nature of phenomena or the original object under study and consists of two stages - first, the creation of a qualitative and then a quantitative model. Computer modeling, on the other hand, consists in conducting a series of computational experiments on a computer, the purpose of which is to analyze, interpret and compare the simulation results with the real behavior of the object under study and, if necessary, further refine the model, etc.

The main stages of computer modeling include:

- statement of the problem, determination of the modeling object;
- development of a conceptual model, identification of the main elements of the system and elementary acts of interaction;
- formalization, that is, the transition to a mathematical model; creating an algorithm and writing a program;
- planning and conducting computer experiments;
- analysis and interpretation of results.

One of the effective ways to study phenomena is a scientific experiment, that is, the reproduction of the phenomenon under study under controlled conditions that can be controlled. The object under study is often replaced by a computer model due to greater convenience and economy. Due to the spread of powerful computers and information technologies, computer modeling can now be called the most effective method for studying physical, technical and other systems. Computer models make it possible to identify the main conditions that determine the properties of the studied phenomena and objects, to study the feedback of the system on changing conditions.

A computer model is a separate program or a software package that allows, using calculations and graphical display of results, to reproduce real objects and processes when exposed to various factors. Such models are also called *simulation models*.

Computer modeling is a method for solving the problem of analysis or synthesis of a complex system based on the study of its computer model. The meaning of such modeling is to obtain quantitative and qualitative results from the created model, which makes it possible to study previously unknown properties of the system. A computer model should display the maximum number of relationships and characteristics of a real object, existing restrictions. The model should be built universal in order to use it to describe similar objects; easy to manage reasonable spending on research.

The computer model is also an excellent visual and educational tool for students. When using a computer model as a learning mechanism, there are possibilities:

- consider complex phenomena and processes at an accessible level;
- focus on the main properties of the system due to the flexible form of its presentation and the presence of multimedia effects;
- observe the process in dynamics, taking into account all its changes;
- present the work of the system in a visual form: graphs, charts, diagrams;

take actions that are impossible in reality due to space-time constraints or concerns for the safety of the model and the environment.

Types of computer models.

To begin with, let's define what computer simulation can be.

Physical modeling is a simulation in which an entire installation is created for conducting experiments or a separate simulator, for example, for training in aircraft control. Such a model receives external signals, performs the necessary mathematical operations and issues the appropriate signals to control the model.

Numerical modeling - solving a system of equations by mathematical methods, conducting a computational experiment based on the input parameters of the system and external influences on it. An example is the modeling of any natural and artificial processes.

The essence of simulation modeling is to create a program that will simulate the behavior of a complex system. Such an imitation is based on a formal description of the logic of the system's existence, which takes into account the interactions of all its components. Examples are studies of biological, physical and other systems, as well as the creation of games and educational programs.

Information modeling is the creation of an information model, that is, data combined together, classified according to certain characteristics, which determine the essence of the object under study. The information model is tables, graphs, animations, charts, maps.

Modeling of knowledge, which includes the creation of artificial intelligence systems. The basis of such models is knowledge of any area, consisting of data and rules. Examples are expert systems, logic games, programs for robots, creating virtual reality effects, and so on.

Based on all of the above, computer models can be divided into:

- descriptive models describing the object under study and the factors influencing changes in its behavior.
- optimization models help determine the most appropriate way to interact with a complex system and manage it.
- predictive models predict the state of an object at specific moments in the future.
- training models used for visual training of students, their testing.
- game models create non-existent situations that imitate reality, play logic games.

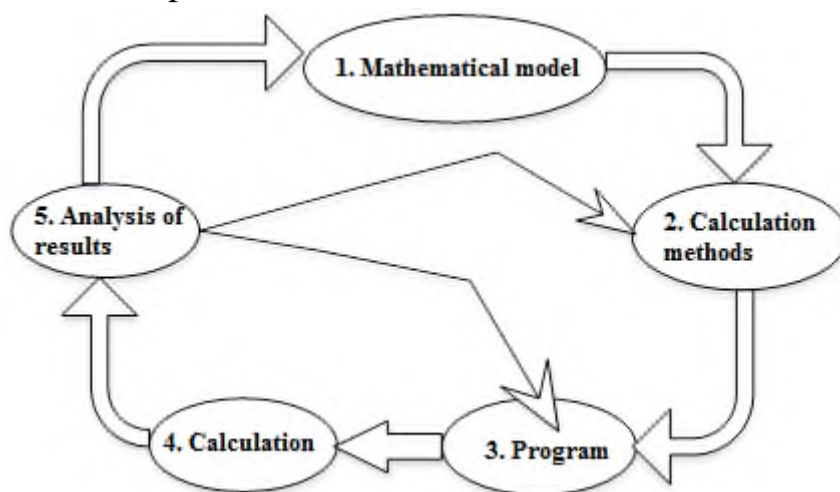
Computer modeling initially meant only simulation modeling, however, it is not difficult to see that the use of a computer for other purposes can greatly help to solve the tasks. For example, the construction of modern mathematical models based on input experimental data is impossible or difficult to achieve without the use of a computer. The first problems solved using computer modeling were related to physics and were mainly complex nonlinear problems of physics using iterative schemes and, in fact, were mathematical modeling. Good results in modeling in the field of physics have extended the use of this research method to other areas. The complexity of the problems solved by modeling depended only on the power of the computers used, and thus was limited by imperfect powers.

A kind of computer simulation is a computational experiment, which involves further numerical study of the model after its creation, which makes it possible to study the object in its various modifications and under various conditions.

With the use of computers to perform arithmetic and logical operations, the productivity of human intellectual labor has increased significantly. The first tasks for which computers were created were related to nuclear energy and space exploration. Now the computer takes part in various tasks and research, this technology of theoretical experiments is called a computational experiment. The basis of a computational experiment is mathematical modeling, the theoretical basis is applied mathematics, and the technical basis is powerful electronic computers.

Computer simulation and computational experiment are becoming a new method of scientific knowledge for the study of complex models of systems. The cycle of a computational experiment is usually divided into several stages for a better understanding of the essence of this method.

Cycle of computational experiment.



- Building a mathematical model. At the same time, the formulation of assumptions takes place, in which the results will be real for this model. The mathematical model, as a rule, is a differential or integral equation.

- Choice of numerical calculation methods. These methods are a set of sequences of mathematical formulas, according to which calculations should be carried out. A necessary condition is that computational methods must be efficient in order to obtain an exact solution with minimal time and resources.

- Creation of a program that implements a computational algorithm.

- Carrying out calculations and processing the received information.

- Analysis of calculation results, comparison with natural experiment (if possible). The results of this stage can be of two types: there is a need to revise the model and refine it, or the obtained calculations are checked for adequacy and the experiment is considered completed. Most often, there is a need to correct the created model, change the numerical methods and the program.

Thus, the algorithm is improved, the mathematical model is refined.

Expert system.

An expert system is the most well-known and widespread type of intelligent systems. Although this term is used very widely, there is no precise definition yet. One can only indicate a number of features that are unique to expert systems.

The first feature of expert systems is that they are intended for users whose field of activity is far from artificial intelligence, programming, mathematics, and logic. For such users, the expert system acts as a kind of system that helps them in their daily work. Communicating with expert systems, working with them should be as simple as simple, for example, controlling a TV, washing machine or car. The expert system is a typical human-machine system, so it should include a block called "intelligent interface". Its task is to provide a dialogue with the user in his usual language. The composition of the intelligent interface may include visualization tools, with the help of which the necessary images are formed on the display screen, used in the process of user communication with the system (drawings, diagrams, drawings, etc.).

Communication with the user takes place in the "question-answer" mode, and questions can be asked by both the user and the system. Expert systems are used primarily as advisory systems in those situations where the specialist doubts the choice of the right solution. The expert knowledge stored in the system's memory is deeper and more complete than the user's corresponding knowledge. When working online, when a person has very little time to think and choose decisions (for example, in the event of an emergency in the power system or in a flying plane), the wrong decision is made not because the person could not find a better one, but because of his mental features. Stress leads to the fact that even obvious solutions are not so easy to find. In these cases, prompt systems can be used. According to the current situation, such a system begins to give advice at a pace sufficient for a person to react to them.

Operational management systems may also have less knowledge (stored in a database) than a specialist working in tandem with the system. But on the other hand, the speed and accuracy of the reaction of the system is much higher than that of a person.

There is another class of systems that do not have their own name and therefore are often called expert. Unlike classical expert systems, they are designed not for a user who is a beginner or an average specialist in a certain field of activity, but for the experts themselves. For such specialists, it is not a consulting or advising system that is needed, but a system that can help them in their scientific work. Systems of this kind are called research automation systems. An example would be systems that, based on the private knowledge of an expert, can detect hidden connections and patterns in empirical material.

Experts gather knowledge about the effect of compound structure on biological activity. This knowledge is put into the knowledge base, and the system tries to generalize them in the form of a pattern based on new material. Found dependencies are given to a specialist who tries to understand and interpret them. He enters his new ideas into the system, and the joint work continues.

When creating expert systems, the most time-consuming step is filling the knowledge base with the information necessary for the functioning of the future system. This work is performed by a special specialist - a knowledge engineer.

An expert system is a computer system that uses expert knowledge and inference procedures to solve problems that require expertise and provide an explanation for the results.

ES has the ability to accumulate knowledge, issue recommendations and explain the results obtained, the ability to modify the rules, prompt the conditions missed by the expert, manage the goal or data. ES are distinguished by the following characteristics:

- intellectuality;
- ease of communication with a computer;
- the possibility of building modules;
- integration of heterogeneous data;
- the ability to resolve multi-criteria tasks, taking into account the preferences of decision makers (DM);
- work in real time;
- documentation;
- confidentiality;
- unified form of knowledge;
- independence of the inference mechanism;
- ability to explain results.

Currently, the following main areas of application of ES can be distinguished:

- diagnostics,
- planning,
- simulation modeling,
- pre-project survey of enterprises,
- office activities,
- as well as some others.

The task of this direction includes the research and development of programs (devices) that use knowledge and inference procedures to solve problems that are difficult for human experts. ES can be classified as general-purpose AI systems - systems that not only execute given procedures, but also generate and apply procedures for solving new specific tasks based on search meta-procedures.

The huge interest in ES on the part of users is caused by at least three reasons.

• Firstly, the systems are focused on solving a wide range of problems in non-formalized areas, on applications that until recently were considered inaccessible to computer technology.

• Secondly, with the help of ES, specialists who do not know programming can independently develop applications of interest to them, which can dramatically expand the scope of computer technology.

- Thirdly, when solving practical problems, ES achieve results that are not inferior, and sometimes even exceed the capabilities of human experts who are not equipped with ES.

Currently, ES are used in various fields of human activity. The most widespread in the production of RES ES received in the design of integrated circuits, in troubleshooting and programming automation.

The following characteristics of ES are distinguished: purpose, problem area, depth of analysis of the problem area, type of methods and knowledge used, system class, stage of existence, tools.

The purpose is determined by the following set of parameters: the purpose of creating an expert system is for training specialists, for solving problems, for automating routine work, for replicating expert knowledge, etc.; the main user is a non-specialist in the field of expertise, a specialist, a student.

The problem area can be defined by a set of parameters of the subject area and tasks solved in it. Each of the parameters can be considered from the point of view of both the end user and the developer of the expert system.

From the user's point of view, the subject area can be characterized by its description in terms of the user, including the name of the area, the list and relationships of subdomains, etc., and the tasks solved by existing expert systems - by their type. Typically, the following types of tasks are distinguished:

- interpretation of symbols or signals - drawing up a semantic description according to the input data;
- diagnostics - definition of faults;
- prediction - determination of the consequences of observed situations;
- design - development of an object with specified properties subject to the established restrictions;
- planning - determination of the sequence of actions leading to the desired state of the object;
- tracking - observation of the changing state of the object and comparison of its indicators with the established or desired ones;
- control - the impact on the object to achieve the desired behavior.

Test questions:

1. What is meant by an expert system?
2. Name the features of expert systems.
3. Give the structure of the expert system.
4. What qualities should an ES have?

Lecture 4

Math modeling. The use of mathematical packages for the study and analysis of mathematical models, mathematical packages (3D Max, Solid Works, Matlab and MathCAD).

Plan:

- 1 Mathematical modeling
- 2 Math packages
3. Modular structure of mathematical modeling systems

***Keywords:** Model and simulation, mathematical packages, analysis, mathematical models, modular structure.*

Mathematical modeling is one of the most important objects of the product design process, which makes it possible to widely use the methods and means of mathematics in the analysis of the behavior of the designed object. The mathematical model allows developers to simulate the behavior of the designed object in various conditions of its operation without making a physical prototype. This leads to a significant reduction in the development and implementation time of a particular production process, as well as savings in the costs required for prototyping.

Model and simulation are universal concepts, attributes of one of the most powerful methods of cognition in any professional field, cognition of systems, processes, phenomena. Model and simulation bring together specialists from different fields working to solve different problems.

Mathematical modeling is symbolic modeling, in which the description of an object is carried out in the language of mathematics, and the model is studied using certain mathematical methods. The quality of mathematical modeling is determined by the completeness of the mathematical description of the considered physical object, the accuracy of the computational procedures used in the implementation of the MM and the reliability of the initial data on the physical object itself. Therefore, special attention in modeling should be paid to the adequacy of the model in relation to a specific problem. The adequacy of the developed model is assessed by comparing the simulation results with the parameters of the real system and determining the limits of its applicability

Currently, it is one of the most effective and most frequently used methods of scientific knowledge. The advantages of mathematical modeling compared to other types of modeling are:

- cost-effectiveness, saving the resources of a real system;
- the possibility of modeling hypothetical, i.e. objects and systems not implemented in nature;

- the possibility of implementing regimes that are dangerous or difficult to reproduce in reality;
- the ability to change the time scale;
- versatility of hardware and software, the availability of application software packages for a wide range of work;
- the ability to predict and identify common patterns;
- the possibility of a relatively simple multivariate analysis.

Considering that the system is a set of interrelated elements (objects) in a certain sense isolated from the environment and interacting with it as a whole, it is possible to formulate the definition of the mathematical model of the system.

A mathematical model of a system is a set of mathematical models of elements that are interconnected and interact with each other and adequately reflect the properties of the system.

Stages of building mathematical models:

- Statement of the problem
- Formalization
- Setting goals and objectives for modeling
- Choice of numerical apparatus and carrying out calculations / solving equations
- Debugging and correction of the model
- Assessment of accuracy and interpretation of results
- Integration (integration of solutions into old systems)

The rapid development of computer technology has led to the emergence of computer-aided design systems that allow you to raise design work to a qualitatively new level. They increase the pace and quality of design, allow you to more effectively solve many complex engineering problems that were previously considered only simplistic. In the field of engineering design, there are three main sections:

- CAD - Computer Aided Design - design and creation of drawings;
- CAM - Computer Aided Manufacturing - means of technological preparation for the production of products;
- CAE - Computer Aided Engineering - automation tools for engineering calculations, analysis and simulation of physical processes, perform dynamic modeling, verification and optimization of products.

Math packages are an integral part of the world of CAE systems. This part cannot be considered secondary, because. Some tasks are generally impossible to solve without the help of a computer. Moreover, today even theorists (the so-called pure, not applied mathematicians) resort to the systems of symbolic mathematics, for example, to test their hypotheses. Professional mathematical packages are programs (software packages) that have the means to perform various numerical and analytical (symbolic) mathematical calculations, from simple arithmetic calculations to solving partial differential equations, solving optimization problems, testing statistical hypotheses, tools for constructing

mathematical models and others. tools necessary for carrying out various technical calculations. All of them have advanced graphical tools, a convenient help system, as well as reporting tools. Modern mathematical packages can be used both as a regular calculator, and as a means to simplify expressions when solving any problems, and as a graphics or even sound generator. Means of interacting with the Internet have also become standard, and the generation of HTML pages is now performed right in the process of computing.

Mathematical packages can be divided into two groups:

- programs of symbolic mathematics
- programs for numerical problem solving.

Numerical problem solving programs are designed to solve mathematical problems using numerical methods. These packages include: Statistica, SPSS, SAS, Stadia, Statgraphics, Matlab, and others. Some packages, such as MathCAD, Maple, Maxima, and Mathematica, include both the capabilities of symbolic mathematics packages and packages using numerical methods.

At present, almost all modern CAE programs have built-in functions for symbolic calculations. However, Maple, MathCad, Mathematica and MatLab are considered the most famous and adapted for mathematical symbolic calculations.

Mathematical package. This package was developed by Wolfram Research, Inc. and is regarded as the world leader among computer systems of symbolic mathematics for PC, providing not only the ability to perform complex numerical calculations with the output of their results in a graphical form, but also to carry out especially laborious analytical transformations and calculations.

Versions of the package for Windows have a modern user interface and allow you to prepare documents in the form of Notebooks. They combine source data, descriptions of algorithms for solving problems, programs and solution results in a wide variety of forms (mathematical formulas, numbers, vectors, matrices, tables and graphs). At the same time, the package provides a dynamic connection between the cells of documents in the style of spreadsheets, even when solving symbolic problems, which fundamentally and favorably distinguished it from other similar packages. Mathematica was conceived as a package that automates the work of scientists and mathematicians-analysts as much as possible. It is a powerful and flexible mathematical toolkit that can provide invaluable assistance to most scientists, university and university professors, students, engineers, and even schoolchildren.

From the very beginning, much attention was paid to graphics, including dynamic ones, and even multimedia capabilities - dynamic animation playback and sound synthesis. The set of graphics functions and options that change their action is very wide. Graphics is the strength of the various versions of Mathematica and gives them the lead among computer mathematics packages. As a result, Mathematica quickly became the market leader in symbolic math packages.

Mathematica is, on the one hand, a typical programming system based on one of the most powerful high-level problem-oriented functional programming languages, designed to solve various problems (including mathematical ones), and on the other hand, an interactive system for solving most mathematical problems in an interactive mode without traditional programming.

Mathematica as a programming system has all the capabilities to develop and create almost any control structures, organize I / O, work with system functions and maintain any peripheral devices, and with the help of extension packages (Add-ons), it becomes possible to adapt to the needs of any user. Mathematica has a machine-independent core of mathematical operations that allows the package to be ported to various computer platforms. Extension packages are prepared in Mathematica's own programming language and are the main means for developing the package's capabilities and adapting them to solving specific classes of user problems. The package has a built-in electronic help system containing e-books with real examples.

The disadvantages of the Mathematica package include a very unusual programming language, which, however, is facilitated by a detailed help system.

Maple package. Maple package - one of the first packages of symbolic mathematics is still one of the leaders among the universal systems of symbolic calculations. The symbolic analyzer of the Maple package is the most powerful part of this software, so it was borrowed and included in a number of other CAE packages, such as MathCad and MatLab, as well as in Scientific WorkPlace and Math Office for Word packages for preparing scientific publications. The Maple package provides the user with a convenient intellectual environment for mathematical research at any level and is especially popular in the scientific community.

Maple provides a convenient environment for computer experiments, during which various approaches to the problem are tried, particular solutions, and if programming is needed, those requiring special speed fragments. The package allows you to create integrated environments with the participation of other systems and high-level universal programming languages. To formalize the results, you can use the tools of the Maple package to visualize data and prepare illustrations for publication.

In addition, the package can prepare an article, report, book. Work in the package is interactive: the user enters commands and immediately sees the result of their execution on the screen. At the same time, the Maple package is not at all like a traditional programming environment, where a strict formalization of all variables and actions with them is required.

Here, the choice of appropriate types of variables is automatically ensured and the correctness of the operations is checked, so that in the general case, the description of variables and strict formalization of the notation are not required.

The Maple package consists of a core (procedures written in C), a library written in the Maple language, and a rich front-end. The kernel performs most of the basic operations, and the library contains many commands - procedures that are executed in interpretation mode.

The Maple interface is based on the concept of a worksheet or document containing I/O lines and text, as well as graphics. The package is operated in interpreter mode. In the input line, the user specifies a command, presses the Enter key, and receives the result - an output line (or lines) or a message about an erroneously entered command. An invitation to enter a new command is immediately issued, etc.

The computational capabilities of the package allow you to use Maple at the most elementary level of its capabilities - as a very powerful calculator for calculating given formulas, but its main advantage is the ability to perform arithmetic operations in symbolic form. So, when working with fractions and roots, the program does not convert them to decimal form during calculations, but makes the necessary reductions and conversions to a column, which allows you to avoid rounding errors. To work with decimal equivalents, Maple has a special command that approximates the value of an expression in floating point format.

Maple supports hundreds of special functions and numbers found in many areas of mathematics, science and technology.

Maple also has many powerful tools for evaluating expressions with one or more variables. The package can be used to solve problems of differential and integral calculus, calculation of limits, series expansions, summation of series, multiplication, integral transformations, studies of continuous or piecewise continuous functions, solving ordinary differential equations (ODE), as well as partial differential equations (PDE), including problems with initial conditions (IVP) and problems with boundary conditions (BVP). One of the most used in Maple is the linear algebra package, which contains a powerful set of commands for working with vectors and matrices.

The Maple package supports both 2D and 3D graphics. In this way, you can graphically represent explicit, implicit, and parametric functions, as well as multivariate functions and simple datasets, and look for patterns visually.

Maple graphics tools allow you to build two-dimensional graphs of several functions at once, create graphs of conformal transformations of functions with complex numbers, and plot functions in logarithmic, double logarithmic, parametric, phase, polar and contour forms. You can graphically represent inequalities, implicit functions, solutions to differential equations, and root locus.

Maple can generate surfaces and curves in 3D, including surfaces defined by explicit and parametric functions, as well as solutions to differential equations. At the same time, it can be presented not only in a static form, but also in the form of two- or three-dimensional animation. The Maple package uses a procedural language that is designed for rapid development of math routines and custom applications. The syntax of this language is similar to the syntax of high-level universal languages: C, Fortran, Basic and Pascal. Maple can generate code compatible with programming languages such as Fortran or C, and with the LaTeX typing language, which is very popular in the scientific world and is used for publications.

For example, using the Maple package, you can develop a specific mathematical model, and then use it to generate C code, corresponding to this model.

The disadvantages of the Maple package include only its some "thoughtfulness", and not always justified, as well as the very high cost of this program. Modular structure of mathematical modeling systems. No matter how different the simulators may seem, their modular structure is practically unchanged:

- The graphical interface is human-oriented and is responsible for presenting the mathematical model in a form understandable to a wide range of specialists. These can be block diagrams, physical circuit diagrams, hybrid state maps, etc.

- The database management system is responsible for storing the objects of the model compiled by the user and the required transformations of the structure of its storage.

- Mathematical core takes on the main computational load and in a cycle (according to a given program, guided by the readiness of arguments, and in rare disputable cases - the appearance of which can always be avoided by the priority of mathematical operations) ensures the execution of flows of mathematical functions.

- Servers for visualization and online influences provide an interface between the functioning mathematical core and the user. Results visualization servers – oscilloscope, indicating and indicating devices – depending on situational requirements, can operate either in synchronous or asynchronous modes.

- Servers of online influences on the model are strictly synchronized with the mathematical core.

Test questions:

1. What is model and simulation?
2. What is a mathematical model and mathematical modeling?
3. What are the goals of modeling.
4. What are the types of modeling?

Lecture 5

Graphic modeling. Processing of numerical and graphic information in engineering problems. Implementation of static and dynamic mathematical models in Matlab and MathCAD systems.

Plan:

- 1 Basic concepts of graphic modeling. Scope of application
2. Stages of development of multimedia products
3. 2D design and 3D modeling

Keywords: *multimedia, computer graphics, modeling, visualization, image recognition, geometric model, volume modeling, rendering.*

The scope of computer graphics has become wider due to the advent of personal computers. Computer graphics has become a familiar and necessary tool for specialists in many industries. At present, it is difficult to imagine an area of scientific research where various mathematical models and computer graphics would not be used. Graphic modeling combines the possibility of simultaneously obtaining an image of an object, a process in various information representations: graphics, text, sound, video, and the implementation of motion dynamics, transformation of objects in the form of animation. Examples of graphical information models are diagrams, maps, drawings, graphs, diagrams and much more. Graphic modeling combines the possibility of simultaneously obtaining an image of an object, a process in various information representations: graphics, text, sound, video, and the implementation of motion dynamics, transformation of objects in the form of animation.

With regard to technical systems, graphic modeling is the process of replacing the object of study with some of its model and conducting research on the model in order to obtain the necessary information about the object or system visually. Despite the successes achieved in the theory and practice of modeling, currently require further study of the problem of increasing the efficiency of mathematical modeling of systems by expanding the possibilities of graphical representation of information. This determines the relevance of the development and improvement of methods of graphic visualization - the development of controlled graphical models (UGM), displaying the structure and properties of objects, visualizing the transformation of models, allowing carry out complex studies of the modeling object in one software system.

The construction of controlled graphical models is possible in many computer mathematics systems (MatLab, MathCAD, Maple, Mathematical In the work, the possibilities of the proposed methods and techniques are shown in the MathCAD system, which has wide mathematical and graphical capabilities, a simple interface that allows you to develop high-quality documents containing calculation expressions, graphics and text.

Multimedia is a modern computer information technology that allows you to combine text, sound, video, graphics and animation in a computer system. This concept defines information technology based on a software and hardware complex, which has a core in the form of a computer with the means of connecting audio and video equipment to it. Multimedia technology makes it possible to ensure, when solving problems of automation of intellectual activity, the combination of computer capabilities with traditional means for our perception of sound and video information for the synthesis of three elements (sound, text and graphics, live video).

The tasks to be solved cover all areas of intellectual activity: science and technology, education, culture, business, and are also used in the service environment when creating electronic guides with immersion in the real environment, multitechs. The emergence of multimedia systems, of course, produces revolutionary changes in such areas as education, computer training, in many areas of professional activity, science, art, computer games, etc., including computer-aided design systems. A sharp breakthrough in this direction, which has

occurred over the past few years, is provided primarily by the development of technical and systemic means. This is progress in the development of personal computers: a sharply increased amount of memory, speed, graphics capabilities, characteristics of external memory, achievements in the field of video technology, laser discs, as well as their mass introduction. An important role was also played by the development of methods for fast and efficient compression (scan) of data.

Multimedia technologies are one of the most promising and popular areas of information technology. Historically, computers were originally designed to process only numerical information, however, most multimedia data are difficult to represent and difficult to process in numerical form (require huge amounts of memory and processor power for processing). As a result, the professional creation and processing of multimedia information remains an expensive and not accessible procedure to this day. Further development of multimedia is hampered by the difficulties with the necessary expansion of the means of influence: the mechanism of taste and smell, technology of tactile effects, etc.

Several types of key terminals are used as input devices within CAD, of which alphanumeric terminals are the most widespread and are available in almost all interactive graphic systems. The alphanumeric terminal can be either a conventional screen terminal or a paper-based document terminal. For graphical purposes, the screen terminal has the advantage of being fast, easy to edit, and saves a lot of paper. However, it sometimes happens that permanent recording of information is desirable, and this is most easily provided by a documenting terminal. Many CAD systems use a graphical screen to display alphanumeric information, but it is still more convenient to have a separate screen terminal for this, so as not to disturb the image on the graphical screen and not to make inscriptions on top of this image. The alphanumeric terminal is used to enter commands, functions and a variety of additional information, the control of which is provided by displaying the CRT or printing. Similarly, system messages are displayed to the user. In the process of interactive communication, the computer can display menu lists, program listings, error messages, and so on. The set of commands used to construct graphic images constitutes the so-called graphic language, which can be classified as a higher-level problem-oriented language. The most common graphic language commands can be displayed on the screen to make it easier for the designer to remember them.

Multimedia technologies are one of the most promising and popular areas of information technology. Historically, computers were originally designed to process only numerical information, however, most multimedia data are difficult to represent and difficult to process in numerical form (require huge amounts of memory and processor power for processing). As a result, the professional creation and processing of multimedia information remains an expensive and not accessible procedure to this day. Further development of multimedia is hampered by the difficulties with the necessary expansion of the means of influence: the mechanism of taste and smell, technology of tactile effects, etc.

Several types of key terminals are used as input devices within CAD, of which alphanumeric terminals are the most widespread and are available in almost all interactive graphic systems. The alphanumeric terminal can be either a conventional screen terminal or a paper-based document terminal. For graphical purposes, the screen terminal has the advantage of being fast, easy to edit, and saves a lot of paper. However, it sometimes happens that permanent recording of information is desirable, and this is most easily provided by a documenting terminal.

Many CAD systems use a graphical screen to display alphanumeric information, but it is still more convenient to have a separate screen terminal for this, so as not to disturb the image on the graphical screen and not to make inscriptions on top of this image. The alphanumeric terminal is used to enter commands, functions and a variety of additional information, the control of which is provided by displaying the CRT or printing. Similarly, system messages are displayed to the user. In the process of interactive communication, the computer can display menu lists, program listings, error messages, and so on. The set of commands used to construct graphic images constitutes the so-called graphic language, which can be classified as a higher-level problem-oriented language. The most common graphic language commands can be displayed on the screen to make it easier for the designer to remember them.

There are three ways to represent graphic images: **raster**, **vector** and **fractal**.

A raster image represents a set of dots located on a grid canvas. Each dot can take on different colors, at a minimum black and white. Scope - processing of photographs, drawings, scanned images, etc. The advantage of this type of images is the ability to transfer a large amount of information (photos). The disadvantage is the large amount of memory required to store the image. To solve this problem, a method of compressing images using special data storage formats (jpg, gif, etc.) is used. Raster graphics programs include: Adobe Photoshop, Corel PhotoPaint, Ms Paint (text editor).

A vector image represents a set of actions for creating a drawing using various lines, shapes, color fill commands, and other commands. The scope is the creation of diagrams, drawings, advertising posters, etc. The advantage of this type is the small amount of memory occupied by the pattern. The disadvantage is the artificiality of the image, consisting of a set of primitives. Main programs: Corel Draw, Visio, AutoCad, Arhcad.

Fractal graphics, like vector and 3D graphics, are computational. Its main difference is that the image is built according to an equation or a system of equations. Therefore, nothing but the formula needs to be stored in the computer's memory to perform all calculations. The mathematical basis of fractal graphics is fractal geometry. Here, the method of constructing images is based on the principle of inheritance from the so-called "parents" of the geometric properties of objects-successors. The advantages of fractal graphics are in several factors: - Small size with a large-scale drawing. - There is no end to scaling, the complexity of the picture can be increased indefinitely. - There is no other tool that will allow you to create complex shapes.

The principles of work in these programs are similar to those in word processors and spreadsheets. When creating a new drawing document, you need to use a set of tools (brush, pencil, eraser, shapes and lines, sprayers, etc.) to create a drawing, as if it were done in an album. In this case, you can often apply various commands for transforming, filtering, and applying various effects using the commands in the main menu. Specific commands can be learned using the help system in the program you need to work with. Computer graphics (CG) is a field of computer science whose interests include all aspects of the formation of images using computers.

The most important function of a computer is information processing. Of particular note is the processing of information associated with images. It is divided into three main areas:

- visualization,
- treatment
- image recognition.

Visualization is the creation of an image based on the description (model) of some object. There are a large number of visualization methods and algorithms that differ depending on what and how should be displayed: a graph of a function, a diagram, a map or an imitation of three-dimensional reality - images of scenes in computer entertainment, feature films, simulators, in systems architectural design. Important and interconnected factors here are: the rate of change of frames, the saturation of the scene with objects, the quality of the image, and the consideration of the features of the graphic device.

Image processing is the transformation of images. Examples of image processing are contrast enhancement, sharpening, color correction, smoothing, noise reduction, etc. The processing material can be satellite images, scanned images, radar, infrared images, etc. The task of image processing can be either an improvement depending on a certain criterion (restoration, restoration), or a special transformation that radically changes the image. In the latter case, image processing can be an intermediate step for further image recognition. For example, before recognition, it is often necessary to select contours, create a binary image, and separate the original image by colors. Image processing methods can differ significantly depending on how it was obtained: synthesized by the system, obtained as a result of digitizing a black and white or color photograph.

Image recognition - obtaining a description of the depicted objects. Recognition methods and algorithms were developed primarily to provide vision for robots and for special-purpose systems. But recently, computer-based image recognition systems are increasingly appearing in everyday practice, for example, office text recognition systems or vectorization programs.

Areas of application of computer graphics:

- CAD (computer-aided design systems);
- business graphics (graphical presentation of data);
- visualization of processes and phenomena in scientific research (computer graphic modeling);

- medicine (computed tomography, ultrasound, etc.);
- geodesy and cartography (GIS);
- polygraphy (diagrams, posters, illustrations);
- sphere of mass information (graphics on the Internet, photo);
- cinematography (special effects, computer animation);
- everyday life (computer games, graphic editors, photo albums).

Computer graphics became so widespread with the advent of interactive graphics systems.

Interactive computer graphics (ICG) - the ability of a computer system to create graphics and conduct a dialogue with a person. In the ICG system, the user perceives on the display an image representing some complex object, and can make changes to the description (model) of the object. Such changes can be the input and editing of individual elements, setting numerical values for any parameters, various operations for entering information based on human perception of images. At present, almost any program can be considered an interactive computer graphics system.

Geometric modeling is a section of mathematical modeling that allows you to solve problems in two-dimensional, three-dimensional and, in general, in multidimensional space.

The geometric model includes systems of equations and algorithms for their implementation. The mathematical basis for constructing the model is the equations that describe the shape and movement of objects. The whole variety of geometric objects is a combination of various primitives - the simplest figures, which in turn consist of graphic elements (points, lines and surfaces). Geometric modeling of objects is widely used in computer graphics systems.

Fundamentals of 3D Modeling.

Three-dimensional graphics (3D Graphics, three dimensions of an image) is a section of computer graphics, a set of techniques and tools (both software and hardware) designed to depict three-dimensional objects.

A three-dimensional image on a plane differs from a two-dimensional one in that it involves the construction of a geometric projection of a three-dimensional model of a scene onto a plane (for example, a computer screen) using specialized programs (however, with the creation and implementation of 3D displays and 3D printers, three-dimensional graphics do not necessarily include projection onto a plane). In this case, the model can either correspond to objects from the real world (cars, buildings, hurricane, asteroid), or be completely abstract (projection of a four-dimensional fractal). Three-dimensional graphics is actively used to create images on the plane of a screen or a sheet of printed matter in science and industry, for example, in design work automation systems (CAD; to create solid elements: buildings, machine parts, mechanisms), architectural visualization (this includes the so-called "virtual archeology"), in modern medical imaging systems.

The widest application is in many modern computer games. Also as an element of cinematography, television, printed matter.

3D graphics usually deals with a virtual, imaginary three-dimensional space that is displayed on a flat, two-dimensional surface of a display or sheet of paper. Currently, there are several ways to display three-dimensional information in a three-dimensional form, although most of them represent three-dimensional characteristics rather conditionally, since they work with a stereo image. From this area, stereo glasses, virtual helmets, 3D displays capable of demonstrating a three-dimensional image can be noted. To obtain a three-dimensional image on a plane, the following steps are required:

- *modeling* - creation of a three-dimensional mathematical model, scene and objects in it;
- *texturing* - assigning raster or procedural textures to the surfaces of models (it also implies setting material properties - transparency, reflections, roughness, etc.);
- *lighting* - installation and configuration of light sources;
- animation (in some cases) - giving movement to objects;
- *dynamic simulation* (in some cases) - automatic calculation of the interaction of particles, solid / soft bodies, etc. with the simulated forces of gravity, wind, buoyancy, etc., as well as with each other;
- *rendering* (visualization) - building a projection in accordance with the selected physical model;
- *output of the resulting image to the output device* - display or printer.

Modeling

Scene modeling (virtual modeling space) includes several categories of objects:

- *geometry* (built using various techniques (eg, creating a polygonal mesh) model, such as a building);
- *materials* (information about the visual properties of the model, such as the color of the walls and the reflective / refractive power of windows);
- *light sources* (direction, power, lighting spectrum settings);
- *virtual cameras* (selection of point and angle of projection construction);
- *forces and influences* (settings for dynamic distortion of objects, used mainly for animation);
- *Additional effects* (objects simulating atmospheric phenomena: light in fog, clouds, flames, etc.)

The task of 3D modeling is to describe these objects and place them in the scene using geometric transformations in accordance with the requirements for the future image.

Purpose of materials: For a real-sensor camera, the materials of real-world objects differ in how they reflect, transmit, and scatter light; virtual materials are set to match the properties of real materials - transparency, reflections, light scattering, roughness, relief, etc.

The most popular purely modeling packages are:

- Pixologic Zbrush;
- Autodesk Mudbox;
- Robert McNeel & Assoc. Rhinoceros 3D;

- Google SketchUp.

To create a three-dimensional model of a person or creature can be used as a prototype (in most cases).

Texturing.

Texturing involves projecting bitmap or procedural textures onto the surface of a 3D object according to a UV coordinate map, where each vertex of the object is mapped to a specific coordinate in 2D texture space.

As a rule, multifunctional UV-coordinate editors are part of universal 3D graphics packages. There are also standalone and plug-in editors from independent developers, such as Unfold3D magic, Deep UV, Unwrella, and others.

Lighting. It consists in creating, directing and configuring virtual light sources. At the same time, in the virtual world, light sources can have a negative intensity, taking light from the zone of their "negative illumination". Typically, 3D graphics packages provide the following types of lights:

- Omni light (Point light) – omnidirectional;
- Spot light - conical (spotlight), source of diverging rays;
- Directional light - source of parallel beams;
- Area light (Plane light) - a light portal that emits light from the plane;
- Photometric — light sources modeled according to the glow brightness parameters in physically measurable units, with a given glow temperature.

There are also other types of light sources that differ in their functional purpose in different 3D graphics and visualization programs. Some packages provide the ability to create sources of volumetric glow (Sphere light) or volumetric lighting (Volume light), within a strictly specified volume. Some provide the ability to use geometric objects of arbitrary shape.

Animation. One of the main vocations of three-dimensional graphics is to give movement (animation) to a three-dimensional model, or to simulate movement among three-dimensional objects. Universal packages of three-dimensional graphics have very rich possibilities for creating animation. There are also highly specialized programs created purely for animation and with a very limited set of modeling tools:

- Autodesk Motionbuilder;
- PMG Messiah Studio.

Rendering. At this stage, the mathematical (vector) spatial model turns into a flat (raster) picture. If you want to create a movie, then a sequence of such pictures - frames is rendered. As a data structure, an image on the screen is represented by a matrix of dots, where each dot is defined by at least three numbers: the intensity of red, blue, and green.

Thus, rendering converts a 3D vector data structure into a flat matrix of pixels. This step often requires very complex calculations, especially if you want to create the illusion of reality. The simplest kind of rendering is to draw the outlines of the models on the computer screen using projection, as shown above. Usually this is not enough and you need to create

an illusion of the materials from which the objects are made, as well as calculate the distortion of these objects due to transparent media (for example, liquid in a glass).

There are several rendering technologies, often combined together.

For example:

- Z-buffer (used in OpenGL and DirectX 10);
- Scanline (scanline) - calculation of the color of each point of the picture by constructing a ray from the point of view of the observer through an imaginary hole in the screen at the place of this pixel "into the scene" before crossing with the first surface. The color of the pixel will be the same as the color of this surface (sometimes taking into account lighting, etc.);
- Ray tracing (ray tracing, English ray tracing) - the same as scanline, but the color of the pixel is specified by constructing additional rays (reflected, refracted, etc.) from the point of intersection of the gaze ray. Despite the name, only reverse ray tracing is used (that is, just from the observer to the light source), direct ray tracing is extremely inefficient and consumes too many resources to obtain a high-quality image;
- Global illumination (English global illumination, radiosity) - calculation of the interaction of surfaces and media in the visible spectrum of radiation using integral equations.

The line between ray tracing algorithms is now almost erased. So, in 3D Studio Max, the standard renderer is called the Default scanline renderer, but it considers not only the contribution of diffuse, reflected, and intrinsic (self-illumination colors) light, but also smoothed shadows. For this reason, more often Raycasting refers to reverse ray tracing, while Raytracing refers to forward ray tracing.

Test questions.

1. Define 3D graphics.
2. What steps are required to get a 3D image?
3. Describe the contents of the steps for obtaining a 3D image.
4. List the main software for 3-D modeling.
5. Tell us about the history of the development of 3D modeling.
6. Tell us about the main modeling systems.

Lecture 6

Fundamentals of simulation modeling. Classification of simulation software. Computational experiment. Varieties of simulation modeling. Simulation modeling using the Simulink package.

The problems that we face in various areas of our lives are constantly becoming more complex. This determines the need to improve existing and develop new methods and

procedures for their solution. Simulation modeling is an effective tool for solving complex problems. Simulation models can be used for:

- studies of the boundaries and structures of systems in order to solve specific problems;
- determination and analysis of critical elements, components and points in the systems and processes under study;
- synthesis and evaluation of proposed solutions;
- forecasting and planning the future development of the systems under study.

A simulation model will be called a logical-mathematical description of the system, which can be studied in the course of experiments on a PC and, therefore, can be considered a laboratory version of the system. After the development of the simulation model is completed, machine experiments are carried out with it, which allow us to draw conclusions about the behavior of the system:

- without its construction, if it is a designed system;
- without interfering with its functioning, if it is an operating system, experimentation with which is either too expensive or unsafe;
- without destroying it, the purpose of the experiment is to determine the limits of impact on the system.

Simulation modeling (simulation) - a method that allows you to build models that describe processes as they would take place in reality. Such a model can be "played" in time for both one test and a given set of them. In this case, the results will be determined by the random nature of the processes. Based on these data, fairly stable statistics can be obtained. Simulation modeling can be considered as a kind of experimental testing. Unlike the prototype system ("in hardware"), it:

- less expensive;
- has the ability to conduct experiments by changing the key parameters;
- dynamic (can describe behavior over time).

The process of sequential development of a simulation model begins with the creation of a simple model, which then gradually becomes more complex in accordance with the requirements of the problem being solved.

In the process of simulation modeling, the following main stages can be distinguished:

1. Formulation of the problem; description of the problem under study and definition of the objectives of the study.
2. Development of the model: a logical and mathematical description of the system being modeled in accordance with the formulation of the problem.
3. Data preparation: identification, specification and data collection.
4. Translation of the model: translation of the model into a language acceptable for the PC used.
5. Verification: establishing the correctness of machine programs.
6. Validation: assessment of the required accuracy and compliance of the simulation model with the real system.

7. Strategic and tactical planning: determining the conditions for conducting a machine experiment with a simulation model.

8. Experimentation: running a simulation model on a PC to obtain the required information.

9. Analysis of the results: studying the results of a simulation experiment to prepare conclusions and recommendations for solving the problem.

10. Implementation and Documentation: Implementation of the recommendations derived from the simulation and documentation of the model and its use.

Stages of simulation modeling.

Simulation modeling as a special information technology consists of the following main stages.

1. Structural analysis of processes. The structure of a complex real process is formalized by decomposing it into subprocesses that perform certain functions and have mutual functional links according to the legend developed by the working expert group. The identified sub-processes, in turn, can be divided into other functional sub-processes.

The structure of the general simulated process can be represented as a graph having a hierarchical multilayer structure, as a result, a formalized image of the simulation model appears in graphical form. Structural analysis is especially effective in modeling economic processes, where (unlike technical ones) many of the constituent sub-processes do not have a physical basis and proceed virtually, since they operate with information, money, and the logic (laws) of their processing.

2. Formalized description of the model. The graphic representation of the simulation model, the functions performed by each sub-process, the conditions for the interaction of all sub-processes and the behavior of the simulated process (temporal, spatial and financial dynamics) should be described in a special language for subsequent translation.

3. Building a model. Usually this is translation and editing of links (model assembly), verification (calibration) of parameters. Translation is carried out in different modes: in interpretation mode, or in compilation mode. Each mode has its own characteristics. The interpretation mode is easier to implement. A special universal interpreter program, based on a formalized description of the model, launches all simulating subroutines.

This mode does not result in a separate simulator that could be transferred or sold to the customer (you would have to sell both the model and the simulation system, which is not always possible).

Building a model in the Simulink system.

The Simulink package is designed for modeling and simulation at the system level, which allows you to conduct a comprehensive study of the system being developed in a single design environment. Modeling and simulations allow you to test the behavior of the system in critical conditions or emergency scenarios.

Simulink is a simulation and model-based design environment for dynamic and embedded systems integrated with MATLAB. Simulink is a dataflow graphical

programming language tool for simulating, simulating and analyzing multi-domain dynamic systems. Basically it is a graphical flowchart tool with a customizable set of block libraries. This allows you to include MATLAB algorithms in models as well as export simulation results to MATLAB for further analysis.

Simulink supports:

- system design
- modeling
- automatic code generation
- testing and validation of embedded systems

There are several other add-on products provided by MathWorks and third-party hardware and software products that are available for use with Simulink.

The following list gives a brief description of some of them - Stateflow allows you to develop state machines and flowcharts. Simulink Coder allows you to generate C source code for automatic implementation of real-time systems. xPC Target, together with x86-based real-time systems, provide an environment for simulating and testing Simulink and Stateflow models in real time on a physical system. The built-in encoder supports certain built-in targets.

HDL Coder allows you to automatically generate synthesized VHDL and Verilog. SimEvents provides a library of graphical building blocks for simulating queuing systems. Stateflow allows you to develop state machines and flowcharts. Simulink Coder allows you to generate C source code for automatic implementation of real-time systems.

Simulink is able to systematically test and validate models through modeling style checking, requirements tracking, and model coverage analysis.

Simulink Design Verifier allows you to identify design errors and generate test scripts to validate your model. The Simulink simulation system is a component of The MathWorks' MatLab integrated engineering environment. Simulink combines the visibility of analog machines with the precision of digital computers. Simulink provides the user with access to all the features of the MatLab package, including a large library of numerical methods.

When modeling using Simulink, the principle of visual programming is implemented, according to which the user creates a device model on the screen from the library of building blocks and performs calculations. Simulink provides an interactive simulation environment where the behavior of the model and the results of its operation are displayed as you run, and it is possible to change the parameters of the model even while it is running. Simulink allows you to create your own blocks and block libraries with access from MatLab, Fortran or C programs, link blocks with previously developed Fortran and C programs that contain already tested models.

Stages of building a model in the Simulink system.

Before building the model, you must first load the Matlab system and run the Simulink subsystem. In both cases, the Simulink Library Browser window will open (Simulink library browser), shown in Fig. 1.

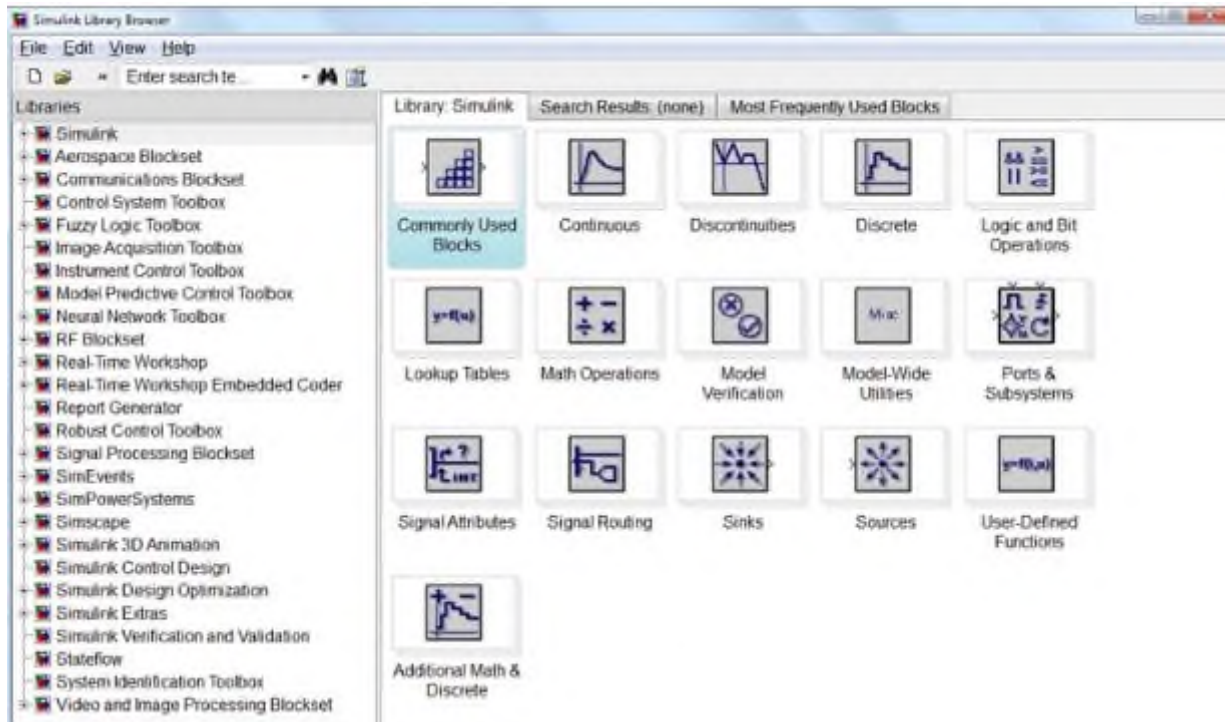


Fig. 1. Tree of standard libraries of the Simulink system.

At the top of this window, the two leftmost buttons serve respectively to create a new and open an existing model.

After pressing the left button, a window for building a new model will appear on the screen (Fig. 2., a). The process of building a Simulink model involves building the model and setting the required parameters. The layout consists in selecting the necessary blocks from the Simulink libraries, placing them in the window that opens (Fig. 2., b) and interconnecting them (Fig. 2., d). Further, for each block, the corresponding parameters are set (Fig. 2., c), which meet the requirements of the simulated system. In order to build a Simulink model, you need to know what types of blocks are provided to the user.

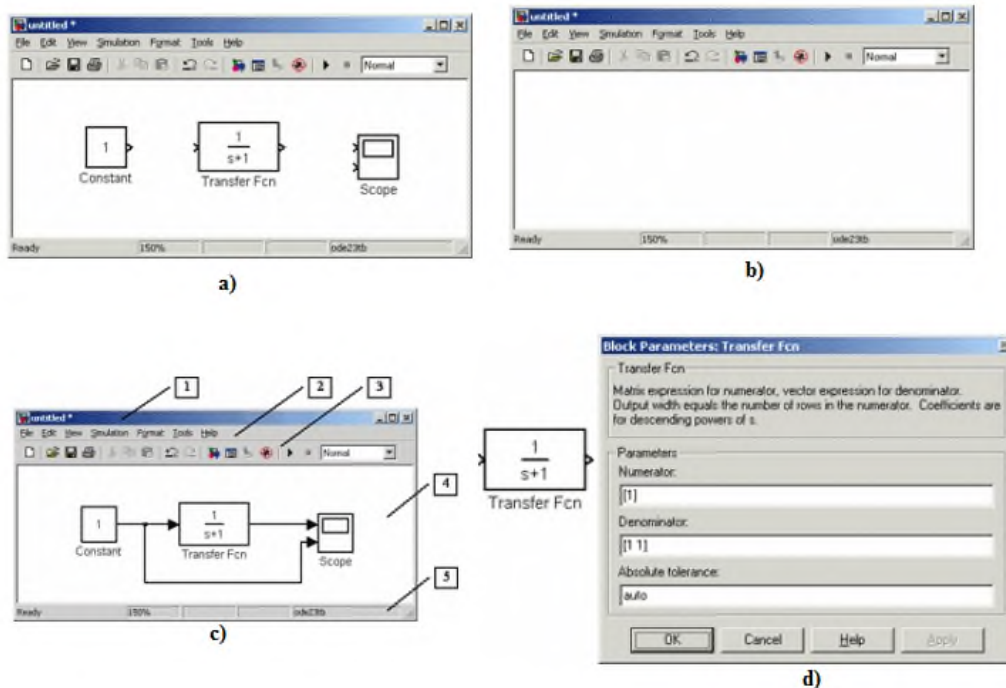


Fig 2. Working windows of the Simulink subsystem when creating a model.

Test questions:

1. What is the model?
2. What is simulation modeling?
3. Where are simulation models used?
4. List the stages of building a simulation?

Lecture 7

Programming in MATLAB. Modules and their functions. Methods for building graphical models in MATLAB

Plan:

1. Basic programming tools
2. Control structures.
3. Graphing
4. Graphs of functions of two variables (3D graphics)

Key terms: *programming, programming tools, script file, program modules, compilers, procedural programming, functional programming, logic programming, structured programming, visual-oriented programming.*

Using the command mode (command line mode, command window) is not the main one when using the capabilities of the MATLAB environment. However, when solving a number of serious problems, it becomes necessary to preserve the used sequences of calculations, as well as their further modification, i.e. there is a need for programming

problem solving. It is practically impossible to foresee in one, even the largest and most powerful, mathematical system the possibility of solving all the problems that may interest the user.

Programming in the MATLAB system is an effective means of expanding and adapting it to solve specific problems. It is implemented using the system programming language. Programs in the MATLAB programming language are saved as text *m-files*. In this case, both entire programs can be saved in the form of script files, as well as individual program modules - functions. In addition, it is important that the program can change the structure of calculation algorithms depending on the input data and data generated during the calculations. From the perspective of a programmer, the system programming language is a typical high-level domain-specific programming language. More precisely, it is a super-high-level language containing complex operators and functions that would require a lot of effort and time to implement in ordinary languages (for example, BASIC, Pascal or C). Such functions include matrix functions, fast Fourier transform (FFT) functions, etc., and operators — operators for constructing various graphs, generating matrices of a certain type, etc.

Basic programming tools. Programs in the MATLAB system are m-files of a text format containing the recording of programs in the form of program codes. The programming language of the MATLAB system has the following facilities:

- data of various types;
- constants and variables;
- operators, including operators of mathematical expressions;
- built-in commands and functions;
- user functions;
- governing structures;
- system operators and functions;
- language extension tools.

Program codes in the MATLAB system are written in a high-level language that is understandable enough for users of moderate programming skill. The MATLAB programming language is a typical interpreter. This means that each program instruction is recognized and immediately executed, which makes it easier to provide an interactive mode of communication with the system. There is no compilation step for all instructions, i.e. the complete program. Interpretation means that MATLAB does not create executable end programs. They exist only as m-files. The MATLAB environment is required to run the programs. However, for programs in the MATLAB language, compilers have been created that translate MATLAB programs into the codes of the C and C++ programming languages. This solves the problem of creating executable programs originally developed in the MATLAB environment.

Basic data types. The data types array and numeric are virtual ("apparent"), since no variables can be assigned to them. They serve to define and complete certain types of data.

Thus, the following basic data types are defined in MATLAB, which in the general case are multidimensional arrays:

single - numeric arrays with single precision numbers;

double — numeric arrays with double precision numbers;

char - string arrays with character elements;

sparse - inherits the properties of double, sparse matrices with double precision number elements;

cell - arrays of cells; cells, in turn, can also be arrays;

struct - arrays of structures with fields that can also contain arrays;

function_handle - function handles:

- int32, uint32 — arrays of 32-bit signed and unsigned numbers;
- int16, uint16 — arrays of 16-bit signed and unsigned integers;
- int8, uint8 - arrays of 8-bit signed and unsigned integers.

Types of programming. The MATLAB programming system is positioned as a high-level language for scientific and technical calculations. The programming language of the MATLAB system has incorporated all the tools necessary to implement various types of programming:

- procedural;
- operator;
- functional;
- logical;
- structural (modular);
- object-oriented;
- visually oriented.

For the MATLAB system language, the distinction between commands (executed by keyboard input) and program statements (executed from a program) is a convention. Both commands and program statements can be executed both from the program and in the direct calculation mode. Hereinafter, commands are mainly understood as means that control peripheral equipment, and operators are means that perform operations with operands (data). The function converts one data to another. Many functions are characterized by returning values in response to accessing them with a list of input parameters - arguments.

M-files of scripts and functions. Both on the command line and in the texts of m-files, functions are written only in small letters. For functions that return a series of values or arrays (for example, X, Y, Z...), the entry is as follows:

```
[X,Y,Z, ...]=f_name (Parameter_list)
```

There are two types of m-files:

- script files
- function files.

In the process of their creation, they undergo syntactic control using the m-file editor/debugger built into the MATLAB system.

M-script files. A script file, also called a Script file, is simply a record of a series of commands with no input or output parameters. The following properties of script files are important:

- they do not have input and output arguments;
- work with data from the workspace;
- are not compiled during execution;

They represent a sequence of operations fixed in the form of a file, completely similar to the one used in the session.

Example script file

```
%Plot with color red
%Plots a sine wave with a red line
%with a scaled grid in the interval [xmin.xmax]
x=xmin:0.1:xmax;
plot(x, sin(x),'r')
grid on
```

The first three lines here are the comment, the rest are the body of the file.

Structure of the function-file The M-file-function is a typical object of the programming language of the MATLAB system. At the same time, it is a full-fledged module from the point of view of structured programming, since it contains input and output parameters and uses the apparatus of local variables. The structure of such a module with one output parameter is as follows:

```
function var=f_name(Parameter_list)
%Main comment
%Additional comment
File body with any expressions
var=expression
```

Function file structure. If there are more output parameters, then they are indicated in square brackets after the word function. The structure of the module looks like this:

```
function [var1,var2....]=f_name(Parameter_list)
%Main comment
%Additional comment
File body with any expressions vag1=expression vag2=expression
```

Function File Examples

```
function y=Norm(x,m,D);
sko=sqrt(D);
y=exp(-(x-m).^2)/(2*D))/(sko*sqrt(2*pi));
```

Using file function in M-file

```
figure(1); x=1:70;
m1=25;
D1=20;
```

```

m2=40;
D2=30;
y1=Norm(x,m1,D1);
y2=Norm(x,m2,D2);
y3=y1+y2;
plot(x,y1,x,y2,x,y3);
grid on;

```

Control structures. In addition to programs with a linear structure, the instructions of which are executed strictly in order, there are many programs whose structure is non-linear. At the same time, program branches can be executed depending on certain conditions, sometimes with a finite number of repetitions - cycles, sometimes in the form of cycles that are completed when a given condition is met. Almost any serious program has a non-linear structure. To create such programs, special control structures are needed. They are available in any programming language, and in particular in MATLAB.

if-else-end

if condition

```

    Expression_1
    Relation_Operator_Expression_2
    Instructions_1
    else Statements_2
    end

```

The following operators are used as **Relation Operators**:

==, , <=, >= or ~=.

All of these operators are pairs of characters with no spaces between them.

if-elseif-else-end v if

Condition v Statements 1 v elseif Condition v Statements _2 v else v Statements_3 v end.

Loop statements are used to program repeatedly performed actions with different data.

- Syntax of the loop statement with a fixed number of repetitions (FOR):

for <counter> = <ini. value> : <increment. counter> : <end. value>

<MatLab statements>

end

The counter increment determines how much the counter value will change after the execution of the loop body. If the increment of the counter is equal to one, then it can be omitted.

For example. Compose a vector of six elements of a geometric progression. The elements of a geometric progression are calculated by the formula $P_n = P_1 * q^{n-1}$. Let $P_1 = 3$, $q = 2$, $n = 6$.

```
>>P(1) = 3; q = 2; % preparation for the cycle
```

```
>> for i = 1:6
```

```
P(i+1) = P(i)*q^i;
```

```
end;
```

```
>>P
```

```
P=
```

```
3 6 24 192 3072 98304 6291456
```

- Syntax of loop operator with precondition (WHILE):

```
while <loop condition>
```

```
<MatLab statements>
```

```
end
```

Statements are repeated as long as the loop condition is met.

Find the maximum value of the matrix of random numbers v:

```
>> v = rand(3)
```

```
v =
```

```
0.9501 0.4860 0.4565
```

```
0.2311 0.8913 0.0185
```

```
0.6068 0.7621 0.8214
```

```
>> f = v;
```

```
>> while length(f)~ = 1
```

```
f = max(f)
```

```
end
```

```
f =
```

```
0.9501 0.8913 0.8214
```

```
f =
```

```
0.9501
```

if statement

The if statement can be used in its simple form to execute a block of commands when some condition is met, or in the **if-elseif-else** construct to write a branching algorithm:

Syntax **if** <condition>

```
<MatLab statements>
```

```
end
```

For branching:

```
if <condition>
```

```
<MatLab statements>
```

```
elseif <condition>
```

```
<MatLab statements>
```

```
elseif <condition>
```

```
<MatLab statements>
```

```
else <MatLab statements> end
```

Depending on the fulfillment of one or another condition, the corresponding branch of the program works, if all the conditions are incorrect, then the commands placed after the else are executed.

Complex conditions are specified using logical operators. Logical operators and examples of their use are shown in Table 1.

Table 1

Logic operators

Operator	Condition	Entry in Matlab	Equivalent entry
Logic AND	$x < 3$ and $k = 4$	<code>and (x < 3, k == 4)</code>	<code>(x < 3) & (k == 4)</code>
Logic OR	$x = 1, 2$	<code>or (x == 1, x == 2)</code>	<code>(x == 1) (x == 2)</code>
Negative "not"	$a \neq 1.9$	<code>not (a == 1.9)</code>	<code>~ (a == 1.9)</code>

Example. Organize a loop to introduce a row vector of three integers. In this case, if the entered value of the current element of the vector is negative, it is squared; if it is positive, it is cubed.

```
>>t = 'yes'; % preparation for the cycle
>>while t~ = 'no' % while t is not equal to the character constant 'no' the loop body is
executed
```

```
B = input('input vector(1*3) ');
if length(B)~ = 3
er = 'There were not three elements entered into the vector'
else
for i = 1:3
if B(i)<0
B(i) = B(i)^2
else B(i) = B(i)^3
end; % for the condition if B(i)<0
end; % for the for loop
end; % for condition if length(B)~ = 3
t = input('Continue looping through vector input? (yes/ no) ');
end; % end of loop body
```

switch statement

The switch statement allows you to make a choice from an arbitrary number of available options.

Syntax

```
switch <key>
case <key_value-1>
<MatLab statements>
case <key_value-2>
<MatLab statements>
...
```


otherwise

<MatLab statements>

end

The transition to a certain branch of the switch statement is performed when the <key> variable takes on the value specified after the case (key values must be integers). If there is no suitable value for the variable, then the branch of the program corresponding to otherwise is executed.

Example. Enter from the keyboard a vector of three odd integers in the range from 5 to 9. Depending on the value of the input element, a matrix of 33 dimensions should be generated with element values equal to the value of the input element. If the value of an element in the vector does not belong to the specified range, an error message is issued.

```
>> P = input('Enter a vector of three odd elements from 5 to 9');
```

```
Enter a vector of three odd elements from 5 to 9 [5 7 9]
```

```
>> for i = 1:3
```

```
switch P(i)
```

```
case 5
```

```
    repmat(5,3,3)
```

```
case 7
```

```
    7*ones(3)
```

```
case 9
```

```
    9*ones(3)
```

```
otherwise
```

```
    'element value is not in range'
```

```
end end
```

GRAPHING

There are several charting modes:

- Displaying the next graph overwrites the current one - the default mode. To save the current and display the next chart, a new current window is created using the figure function.
- The mode of overlaying graphs on each other without overwriting is set by the hold function.
- Display of charts in different subareas of one window. The subplot function is used to split the output window into subareas.

Graphs of functions of one variable (2D graphics)

The plot function is used to plot graphs in the Cartesian coordinate system.

Syntax:

```
plot(X1,Y1,S1,X2,Y2,S2, X3,Y3,S3,..Yi,Xi,Si,...)
```

– draws graphs of functions $Y_i(X_i)$ on the same coordinate axes, where X_i and Y_i are the vectors of the function arguments and its values, respectively, and S_i is the string constant for setting the style of the envelope function.

The elements of the string constant S_i can be the characters presented in Table 2.

Graph style options

Color	Marker	Line style
y – yellow	. – point	Solid
m – magenta	o - circle	: - dotted
c-cyan	x-x-mark	.- dashdot
r-red	^-triangle(up)	-- dashed
g-green	<-triangle(left)	
b-blue	>-triangle(right)	
w-white	p-pentagram	
k-black	h-hexagram	
	*-star	
	v-triangle(down)	
	+plus	
	s-square	
	d-diamond	

To build graphs with widely varying x and y values, logarithmic scales are often used instead of linear ones.

The following are functions that plot graphs on a logarithmic scale.

Syntax:

loglog(X1,Y1,S1,X2,Y2,S2,...,Xi,Yi,Si,...)

– the syntax of the function is similar to that previously considered for the plot(...) function. The logarithmic scale is used for the X and Y coordinate axes. The line style string constant Si is the same as the Si function plot.

To plot graphs on a semi-logarithmic scale, the following functions are used:

Syntax:

semilogx(X1,Y1,S1,X2,Y2,S2,...)

- plots the function on a logarithmic scale (base 10) along the X axis and linear on the Y axis.

semilogy(X1,Y1,S1,X2,Y2,S2,...)

- plots the function on a logarithmic scale along the Y axis and linear along the X axis.

The stem function is used to plot discrete readings.

A graph of discrete samples of the Y(x) function, when each sample is represented by a vertical line topped with a marker, and the height of the marker corresponds to the Y(x) coordinate, is built using the stem function:

Syntax :

stem(X,Y,S)

– plots the function $Y(X)$ (discrete samples), S is a string constant that defines the style of discrete samples, the elements of this constant are similar to the elements included in the string constant of the plot function.

Graphs of functions of two variables (3D graphics)

Surfaces in three-dimensional space are usually described by a function of two variables $z(x,y)$. The specificity of building three-dimensional graphs requires not just setting a series of x and y values, that is, x and y vectors, but two-dimensional arrays for X and Y - matrices. The meshgrid function is used to create such arrays.

Syntax :

$[X,Y] = \text{meshgrid}(x,y)$

– convert the area specified by the x and y vectors into X and Y arrays, which are used to calculate the function of two variables and build three-dimensional graphs.

The function $[X,Y] = \text{meshgrid}(x)$ is the same as $[X,Y] = \text{meshgrid}(x,x)$. Instead of variables x,y , you can set their values.

For example:

```
[X,Y] = meshgrid(-pi:0.1:pi),  
>> x = -pi:0.1:pi; y = [3 7 0 5 7 0 1 5 ];  
>> [X,Y] = meshgrid(x,y);  
or  
>> [X,Y] = meshgrid(-pi:0.1:pi, [3 7 0 5 7 0 1 5 ]);
```

Building 3D Graphs

The $\text{plot3}(\dots)$ function is analogous to the $\text{plot}(\dots)$ function, but with respect to a function of two variables. $z(x,y)$. It is presented in the following forms:

Syntax:

$\text{plot3}(X,Y,Z,S)$

– builds points with X,Y,Z coordinates and connects them with line segments, sets the construction style using the string constant S

There are three groups of commands for plotting graphs with functional coloring of grid nodes.

Syntax:

$\text{mesh}(X,Y,Z,C)$

- displays a mesh surface $Z(X,Y)$ with node colors specified by array C .

$\text{mesh}(X,Y,Z)$

- similar to the previous command with $C = Z$, while using functional coloring, in which the color is naturally set by the height of the surface.

Commands:

- **colormap(...)** allows you to set the coloring of the cells (the list of palettes, along with other information, is displayed by the help graph3d command)

- **shading** with three modes:

- **faceted** - display cell edges, used by default,
- **flat** - edges are not displayed
- **interp** - edges are not displayed and colors between adjacent sections are smoothed sets the coloring method,
- **colorbar** is used to display the color scale.

To build a combined drawing, use the command

• **surfc(...)** – similar to **surf(...)**, but in addition to the surface plot, displays its image as lines of equal level.

Construction of contour plots.

The contour and contourf functions are used to construct contours.

Syntax :

contour(X,Y,Z,N)

- X,Y - arrays that define the nodes of the coordinate grid, on which the surface Z is built.

N is the number of contour lines.

For each level line, you can specify the value that the function under investigation takes on it using the clabel function.

Syntax :

clabel(Cmatr,h)

– **Cmatr** – a matrix with information about the level lines, h – a pointer to the chart on which the markup should be applied.

Test questions:

1. How can you create a structure?
2. Which operator performs the input of information from the keyboard?
3. What is the loop statement with a fixed number of repetitions?
4. What is the reason for the end of the loop statement with a precondition?
5. What is the if, switch statement?
6. List logical operators and give examples of their use.
7. What commands are required to write information to a file?
- 8 Which 2D plotting functions output the envelope of a plot?
9. Which 2D plotting function displays discrete readings?
10. What are the input and output arguments for the meshgrid command?
11. What functions are used to build 3-dimensional graphs?

Lecture 8

Information security systems. Mathematical foundations of cryptology. Issues of information security in computer networks.

Plan:

1. Basic concepts and definitions of information security
2. Types of information threats
3. Cryptographic data protection

Key terms: *cryptology, cryptanalysis, encryption key, plain text, costs, alphabet, natural number, decryption, information, decryption, message recipient, key, encryption, symmetric encryption, asymmetric encryption, data block, electronic signature, data encryption*

Basic concepts and definitions of information security.

In connection with the development of information technologies and the computerization of the economy, one of the most important issues in the company's activities is ensuring information security.

Information security is the preservation and protection of information, as well as its most important elements, including systems and equipment designed to use, store and transmit this information. In other words, it is a set of technologies, standards and management practices that are necessary to protect information security. Information is one of the most valuable and important assets of any business and must be properly protected.

The main components of **information security** is a set of elements that includes openness, confidentiality and integrity of information resources and supporting infrastructures.

The goal of information security is to protect information data and the supporting infrastructure from accidental or intentional tampering that could result in data loss or unauthorized modification. Information security helps ensure business continuity.

Information security tools are divided into:

- Organizational. This is a set of organizational and technical (providing computer rooms, setting up a cable system, etc.) ...
- Software.
- Technical (hardware).
- Mixed hardware and software.

The main goal of information security systems is to guarantee the protection of data from external and internal threats.

For the successful implementation of information security systems in an enterprise, it is necessary to adhere to three main principles:

- **Privacy.** This means putting in place controls to ensure that enterprise data, assets and information are sufficiently secure at various stages of business operations to prevent unwanted or unauthorized disclosure. Confidentiality must be maintained while storing information, as well as in transit through ordinary organizations, regardless of its format.

- **Integrity.** Integrity deals with the controls that are concerned with ensuring that corporate information is internally and externally consistent. Integrity also ensures that information is not corrupted.

- **Availability.** Availability ensures reliable and efficient access to information by authorized persons. The network environment must behave in a predictable manner in order to access information and data when needed. System failure recovery is an important factor when it comes to information availability, and such recovery must also be provided in a way that does not adversely affect operation.

The main requirement of information security is the full protection of confidential information, ensuring its integrity in the complete absence of the risk of harming the work of the enterprise.

How to ensure data security? To ensure information security of data stored and transmitted by technical means, the following are used:

1. Authentication;
2. Regulation of access to objects;
3. File encryption system;
4. Keys;
5. Secure connections;
6. IPsec.

You need to understand that only a systematic and integrated approach to protection can ensure information security. The information security system must take into account all current and probable threats and vulnerabilities. This requires continuous real-time monitoring. Control should be carried out 24/7 and cover the entire life cycle of information - from the moment it enters the organization, and until its destruction or loss of relevance.

Types of information threats.

Information threats can be caused by:

- natural factors (fire, flood, etc.);
- human factors.

The latter, in turn, are divided into:

- threats of an accidental, unintentional nature. These are threats associated with errors in the process of preparing, processing and transmitting information;
- Threats caused by intentional, intentional actions of people. These are threats related to unauthorized access to AIS resources.

Deliberate threats aim to harm AIS users and, in turn, are divided into active and passive.

- Passive threats, as a rule, are aimed at unauthorized use of information resources without affecting their functioning (eavesdropping).

- Active threats are aimed at disrupting the normal functioning of the system through a targeted impact on hardware, software and information resources. The sources of active threats can be the direct actions of intruders, software viruses, etc.

Deliberate threats are also divided into internal, arising within the managed organization, and external.

1. Internal threats are understood as threats to information security, the insider (performer) of which is an internal subject (insider) in relation to the resources of the organization.

2. External threats are understood as information security threats initiated (performed) by an entity external to the organization's resources (remote hacker, intruder).

Internal Threats:

- Information leaks;
- Unauthorized access;
- External threats;
- Malicious programs (viruses, Trojans, worms, etc.);
- Hacker attacks;
- DDos attacks;
- Targeted attacks;
- Spam;
- Phishing;
- Industrial threats (stuxnet, flame, duqu);
- Spyware (spyware, adware);
- botnets (botnets or zombie networks).

1. Hacker attacks.

The term "hacker" used to be used to refer to highly skilled programmers. Now this is the name of those who use vulnerabilities in software to infiltrate a computer system. It's the electronic equivalent of breaking into a room. Hackers are constantly breaking into both individual computers and large networks.

Once they gain access to the system, they steal sensitive data or install malware. They also use hacked computers to send spam. Modern applications are extremely complex; they are compiled from thousands of lines of code. But they are created by people, and people tend to err. Therefore, it is not surprising that bugs creep into programs, making them vulnerable to attack. These loopholes allow hackers to get into the system, and virus writers use errors in application code to ensure that malware automatically runs on the computer.

Hackers are electronic hackers who get into your computer system using special loopholes - software vulnerabilities. You can protect yourself from them with the help of a special application - a firewall. Often it is part of antivirus programs. A firewall, or firewall, recognizes hacking attempts and makes your computer invisible to hackers.

2. Phishing attacks.

Phishing is a special (modern) type of computer fraud.

Phishing attacks are organized as follows: cybercriminals create a fake site that looks exactly like a bank site or a site that makes financial transactions via the Internet. The scammers then try to trick the user into visiting the fake site and entering their sensitive data,

such as a username, password, or PIN, on it. Using them, attackers steal money from the accounts of users who fell for the bait.

Typically, users are lured to a fake site using a bulk email that looks like it was sent by a bank or other real financial institution, but contains a link to the fake site. By clicking on the link, you are taken to a fake site where you are prompted to enter your credentials.

Often, phishing emails use the same logos and design as the emails from the real bank, as well as links that look like the bank's real address on the Internet. In addition, the message may contain your name, as if it were really addressed to you personally. The scammers' emails usually provide a plausible reason requiring you to enter your details on the "bank" website.

Much attention is currently paid to information security issues, and this is not accidental. Telecommunication systems, which have been actively developing in recent years, are the arteries of modern global information systems. The information circulating in such systems is of significant value and therefore vulnerable to various types of abuse. Therefore, it is in recent decades that the problem of information security has become so urgent.

3. Cryptographic data protection

People have been dealing with the problem of protecting information during its transmission between subscribers throughout their history. Currently, qualified specialists in complex information protection are engaged in solving this problem. Among the various means of information security, cryptographic methods occupy a special place. On the one hand, this is due to the fact that cryptographic methods of protecting messages have been known to people and have been successfully used by them for more than one millennium.

On the other hand, new achievements in cryptography make it possible to solve not only the classical problem of protecting data from unauthorized access, but also many other tasks that are inaccessible to other types of information protection tools. This is the task of authenticating users of information systems, and the problem of generating a digital signature for electronic documents, and the possibility of using the so-called electronic money.

Mankind has invented many ways to somehow hide the meaning of the transmitted messages from the enemy. In practice, several groups of methods for protecting secret messages have been developed.

The first way is the physical protection of the material carrier of information from the enemy. Paper, computer media (DVD, flash card, magnetic disk, computer hard disk, etc.) can act as a data carrier. To implement this method, you need a reliable communication channel that is inaccessible to interception. At various times, carrier pigeons, special couriers, radio broadcasts on a secret frequency were used for this. Methods of physical protection of information are also used in modern automated data processing systems. So, for example, complex information security systems are impossible without fencing and physical isolation systems, as well as without security systems.

The second way to protect information, known since ancient times, is steganographic information protection. This method of protection is based on an attempt to hide from the enemy the very fact of the presence of information of interest to him. With the steganographic method of protection from the enemy, a physical data carrier is hidden or secret messages are masked among open, unclassified information. Such methods include, for example, "hiding" a microphotograph with secret information in an unclassified place: under a stamp on a postal envelope, under the cover of a book, etc. Steganography also includes such well-known techniques as "hiding" a secret message in book spines, in buttons, in heels, in tooth fillings, etc. Some of the methods have been developed since ancient times. So, for example, the Greeks found an unusual solution: they shaved the slave's head and scratched their message on it. When the hair on the slave's head grew back, he was sent to deliver a message. The recipient would shave the slave's head and read the text. Unfortunately, it took several weeks to send a message and receive a response in this way. In later times, chemical (sympathetic) inks were most widely used in this direction. Text written in this ink between the lines of an unclassified message is invisible. It appeared only as a result of applying a certain manifestation technology.

In the context of the widespread use of information technology, new steganographic techniques are emerging. For example, a method is known in which a secret message is hidden in a graphic image file. Using this method, the least significant bit in the description of each image pixel is replaced with a message bit. By dividing the entire original message into bits and placing these bits throughout the graphic file, we send the image with the masked message to the recipient. The graphic image does not change too much, especially if a mode with a large number of colors was used, for example, with a color depth of 24 bits per pixel. This is due to the fact that the human eye cannot distinguish such a large number of colors. As a result, a picture of only 32 by 32 pixels can contain a secret message 1024 bits or 128 bytes long.

The third way to protect information - the most reliable and common today - cryptographic. This method of information protection involves the transformation of information to hide its meaning from the enemy. Cryptography in Greek means "secret writing". Currently, cryptography is engaged in the search and study of mathematical methods for transforming information.

Along with cryptography, cryptanalysis is developing and improving - the science of overcoming the cryptographic protection of information. Cryptanalysts are exploring the possibilities of decrypting information without knowing the keys. A successful cryptanalysis yields the encryption key, or the plaintext, or both. Sometimes cryptography and cryptanalysis are combined into one science - cryptology (kryptos - secret, logos - science), dealing with issues of reversible transformation of information in order to protect against unauthorized access, assessing the reliability of encryption systems and analyzing the strength of ciphers.

Currently, cryptography has firmly entered our lives. We list only some areas of application of cryptography in a modern informatized society:

1. Encryption of data during transmission over open communication channels (for example, when making a purchase on the Internet, transaction details such as address, phone number, credit card number are usually encrypted for security purposes);
2. Maintenance of bank plastic cards;
3. Storage and processing of user passwords on the network;
4. Delivery of accounting and other reports through remote communication channels;
5. Banking services for enterprises through a local or global network;
6. Safe storage of data from unauthorized access on the computer's hard drive (the Windows operating system even has a special term - Encrypting File System (EFS)).

Until the beginning of the twentieth century, cryptographic methods were used only to encrypt data in order to protect against unauthorized access. In the twentieth century, in connection with the development of technology for transmitting information over long distances, interest in cryptography increased significantly. Thanks to the creation of new cryptographic methods, the range of cryptographic tasks has also expanded. Currently, it is believed that cryptography is designed to solve the following problems:

1. Actual data encryption to protect against unauthorized access;
2. Message authentication: the recipient of a message can verify its origin;
3. Verification of the integrity of the transmitted data: the recipient can check whether the message was changed or spoofed during the transfer;
4. Ensuring the impossibility of refusal, that is, the impossibility for both the recipient and the sender to refuse the fact of transfer.

The main definitions used in the study of cryptographic methods of information protection

5. Cipher - a set of predetermined ways to transform the original secret message in order to protect it.
6. The original messages are usually referred to as plain texts. In foreign literature, the term plaintext is used for plain text.
7. A symbol is any character, including a letter, number or punctuation mark.

An alphabet is a finite set of symbols used to encode information. For example, the Russian alphabet contains 33 letters from A to Z. However, these thirty-three characters are usually not enough to record messages, so they are supplemented with a space character, a dot, a comma, and other characters. The alphabet of Arabic numerals is the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. This alphabet contains 10 characters and can be used to write any natural number. Any message can also be written using the binary alphabet, that is, using only zeros and ones.

A message received after conversion using any cipher is called an **encrypted message** (closed text, cryptogram).

The conversion of plaintext into a cryptogram is called **encryption**. The reverse action is called **decryption**. In the English literature, the terms "encryption/decryption" correspond to the terms "enciphering/deciphering".

The key is the information needed to encrypt and decrypt messages. From the point of view of the Russian language, the terms "decryption" and "decryption" are synonymous. However, in works on cryptography of recent decades, these words are often distinguished. We will assume that the terms "decryption" and "decryption" are not synonymous. Let us assume that the legal recipient of the message (the one who knows the key) is decrypting, and the person to whom the message is not intended, trying to understand its meaning, is decrypting.

An encryption system, or cipher system, is any system that can be used to reversibly change the text of a message to make it incomprehensible to all but those to whom it is intended.

Cryptographic resistance is a characteristic of a cipher that determines its resistance to decryption without knowing the key (i.e., the ability to resist cryptanalysis).

Thus, taking into account all the definitions made, it is possible to give a more precise definition of the science of "cryptography". Cryptography is the study of the construction and use of encryption systems, including their strength, weaknesses, and vulnerability to various attack methods.

All methods for converting information in order to protect against unauthorized access are divided into two large groups: private key encryption methods and public key encryption methods. Private key encryption (secret key encryption or symmetric encryption) has been used by humans for quite a long time. To encrypt and decrypt data, these methods use the same key, which both parties try to keep secret from the adversary.

Public key encryption (asymmetric encryption) began to be used for cryptographic closing of information only in the second half of the twentieth century. This group includes encryption methods in which two different keys are used to encrypt and decrypt data. In this case, one of the keys (public key) can be transmitted over an open (unprotected) communication channel.

An electronic (digital) signature is a block of data usually attached to a message, obtained using cryptographic transformation. An electronic signature allows, when another user receives a text, to verify the authorship and authenticity of the message.

Cryptographic information security system is an information security system that uses cryptographic methods to encrypt data.

Requirements for cryptographic information protection systems

For the currently developed cryptographic information security systems, the following generally accepted requirements are formulated:

- an encrypted message must be readable only if the key is present;
- knowledge of the encryption algorithm should not affect the reliability of protection;
- any key from the set of possible ones must provide reliable protection of information;

- the encryption algorithm must allow both software and hardware implementation.

Not for all encryption algorithms, the listed requirements are fully met. In particular, the requirement that there are no weak keys (keys that allow an attacker to more easily open an encrypted message) is not satisfied for some "old" block ciphers. However, all newly developed encryption systems meet the listed requirements.

Test questions:

1. Name the problems that cryptographic methods can be used to solve.
2. What is the difference between cryptography and steganography?
3. What tasks does modern cryptography solve?
4. Formulate the requirements for cryptographic information protection systems.

Lecture 9

Cyberspace and the basics of cybersecurity. Computer technology objects used in cybersecurity.

Plan:

1. The concept of "cybersecurity".
2. Types of cybersecurity threats.
3. Goals of cybersecurity.

Key terms: *cybersecurity, information security, infosphere, cyberspace.*

The definition of cybersecurity includes a wide variety of methods, technologies and processes that are aimed at protecting the integrity of networks, programs and information from digital attacks. The key goal of cyberattacks is gaining unauthorized access to confidential data, their copying, modification or elimination.

They can also serve to extort banknotes from users or disrupt work processes in an organization. Cybersecurity is sometimes referred to as computer security or IT (information technology) security.

Cybersecurity is the protection of Internet-connected systems such as hardware, software, and information from cyber threats. This practice is used by individuals and organizations to eliminate unauthorized access to information processing centers and other computerized systems.

The purpose of introducing cybersecurity is to ensure reliable protection of computers, servers, networks, mobile devices and data that are located on the presented devices from intruders. Cyberattacks can be designed to access, destroy, or extort confidential information of an organization or user, making cybersecurity quite meaningful. Medical, government, corporate and financial enterprises are capable of storing vital information, for example, about a person.

Cybersecurity (sometimes referred to as computer security) is a set of techniques and practices for protecting computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks.

Cybersecurity is the protection of Internet-connected systems (hardware, software, and data) from cyber threats.

The terms "cybersecurity" and "information security" are often used interchangeably. However, in reality, these terms are very different and are not interchangeable. Cybersecurity refers to protection against attacks in cyberspace, while information security refers to the protection of data from any form of threat, whether analog or digital.

Cybersecurity practices can be applied in a variety of areas - from industrial enterprises to mobile devices of ordinary users:

Critical infrastructure security - measures to protect computer systems, networks of critical information infrastructure (CII). CII objects include electrical networks, transport networks, automated control systems and information and communication systems, and many other systems, the protection of which is vital for the security of the country and the well-being of citizens.

Network Security - Protecting the underlying network infrastructure from unauthorized access and misuse, as well as information theft. The technology includes building a secure infrastructure for devices, applications, and users.

Application Security - security measures applied at the application level to prevent theft, compromise of application data or code. The methods cover security issues that arise in the development, design, deployment, and operation of applications.

Cloud security is an interconnected set of policies, controls, and tools to protect cloud computing systems from cyber threats. Cloud security measures are aimed at ensuring the security of data, online infrastructure, as well as applications and platforms. Cloud security shares a number of concepts with traditional cybersecurity, but the field also has its own best practices and unique technologies.

User training. A security awareness program is an important step in building a strong company defense. Employee digital hygiene practices help improve endpoint security. For example, users who are informed about current threats will not open attachments from suspicious emails, stop using untrusted USB devices, and stop attaching login and password stickers to their monitors.

Business continuity disaster recovery (planning) is a set of strategies, policies and procedures that determine how an organization should respond to potential threats or unforeseen natural disasters in order to properly adapt to them and minimize negative consequences.

Operational security is a security and risk management process that prevents sensitive information from falling into the wrong hands. The principles of operational security were originally used by the military to prevent sensitive information from reaching the enemy.

Operational security practices are now widely used to protect businesses from potential data breaches.

Types of cybersecurity threats.

Cybersecurity technologies and best practices protect critical systems and sensitive information from a rapidly growing volume of sophisticated cyberattacks.

The main types of threats that modern cybersecurity is struggling with include:

6. Malicious software (malware)

Any program or file that can cause damage to a computer, network, or server. Malware includes computer viruses, worms, Trojan horses, ransomware, and spyware. Malware steals, encrypts, and deletes sensitive data, alters or hijacks basic computing functions, and monitors computer or application activity.

7. Social engineering

An attack method based on human interaction. Malefactors ingratiate themselves with users and force them to violate security procedures and give out confidential information.

8. Phishing

A form of social engineering. Fraudsters send emails or text messages to users that look like messages from trusted sources. In mass phishing attacks, attackers lure bank card information or credentials from users.

9. Target attack

A sustained and targeted cyberattack in which an attacker gains access to a network and remains undetected for an extended period of time. Targeted attacks are usually aimed at stealing data from large enterprises or government organizations.

10. Internal threats

Security breaches or losses caused by insiders—employees, contractors, or customers—with malicious intent or negligence.

11. DoS attack, or denial of service attack

An attack in which attackers try to make it impossible to provide a service. In a DoS attack, one system sends malicious requests; A DDoS attack comes from multiple systems. As a result of the attack, it is possible to block access to almost everything: servers, devices, services, networks, applications, and even certain transactions within applications.

12. Stalker software

Software designed for covert surveillance of users. Stalker applications are often distributed under the guise of legitimate software. Such programs allow attackers to view photos and files on the victim's device, peep through the smartphone's camera in real time, find out location information, read correspondence in instant messengers and record conversations.

13. Cryptojacking

A relatively new type of cybercrime, in which malware hides in a system and steals a device's computing resources so that attackers can use them to mine cryptocurrency. The

process of cryptojacking is completely hidden from the eyes of users. Most victims become suspicious when they notice an increase in electricity bills.

14. Attacks on the supply chain

Supply chain attacks exploit the trust relationship between an organization and its counterparties. Hackers compromise one organization and then move up the supply chain to gain access to the systems of another. If one company has a strong cybersecurity system, but there is an unreliable trusted provider, then attackers will try to hack this provider in order to then penetrate the network of the target organization.

15. Attacks using machine learning and artificial intelligence

In such attacks, the attacker tries to trick the machine algorithm into giving wrong answers. Typically, cybercriminals use the “data poisoning” method, offering neural networks for training a deliberately incorrect sample.

The goal of cybersecurity is to protect data (both in transit and/or exchange and in storage). Countermeasures may also be applied to ensure data security. The main goal of cybersecurity is to prevent the theft or compromise of information. An important role in achieving this goal is played by the triad of a secure IT infrastructure - confidentiality, integrity and availability.

Confidentiality in this context refers to a set of rules that restrict access to information. Integrity ensures that information is accurate and reliable. Availability, in turn, is responsible for the reliability of access to information by authorized persons. Considering the principles of the triad together helps companies develop security policies that provide strong protection.

Lecture 10

Modern programming technologies. Program languages.

Plan.

1. Programming languages, their characteristics.
2. A modern approach to programming. OOP.
3. Purpose and possibilities of C++.
4. Features of the C++Builder6 version

Key terms: *object-oriented approach, functional approach, object, property, class, method, domain, inheritance, event handling, class hierarchy, software, imperative language, visual, data abstraction, encapsulation, polymorphism, sections, computational science.*

Programming languages are usually divided into five generations. The first generation includes languages created in the early 50s, when the first computers were just born. It was the first assembly language created on the principle of "one instruction - one line".

The heyday of the second generation of programming languages came in the late 50s - early 60s. Then a symbolic assembler was developed, in which the concept of a variable appeared. It became the first full-fledged programming language. Thanks to its appearance, the speed of development and reliability of programs have noticeably increased.

The emergence of the third generation of programming languages is usually attributed to the 60s. At this time, high-level universal languages were born, with their help it is possible to solve problems from any area. Such qualities of languages as relative simplicity, independence of a particular type of computer, and the ability to use powerful syntactic constructions, made it possible to dramatically increase the productivity of programmers. The structure of these languages, understandable to most users, attracted a significant number of specialists from non-computer fields to writing small programs (usually of an engineering or economic nature). The vast majority of languages of this generation are successfully used today.

The beginning of the 70s was the period of fourth generation languages. These languages are designed to implement large projects, increase their reliability and speed of creation. They are usually focused on specialized areas of application, where good results can be achieved using not universal, but domain-specific languages, operating with specific concepts of a narrow subject area. As a rule, operators are built into these languages, which allow one line to describe such functionality, which would require thousands of lines of source code in the languages of younger generations.

Brief description of high-level programming languages.

Fortran. It is the first compiled language created by Jim Backus in the 1950s. Programmers who developed programs exclusively in assembler expressed serious doubts about the possibility of a high-level language, so the main criterion in the development of Fortran compilers was the efficiency of the executable code. Although Fortran was the first to implement a number of important programming concepts, the convenience of creating programs was sacrificed for the possibility of obtaining efficient machine code.

However, a huge number of libraries have been created for this language, ranging from statistical complexes to satellite control packages, so Fortran continues to be effectively used in many organizations. Currently, work is underway on the next Fortran standard, which appeared in 2000. There is a standard version of Fortran HPF (High Performance Fortran) for parallel supercomputers with multiple processors.

Cobol is a compiled language for economic and business applications developed in the early 1960s. It has a lot of "verbosity" - its operators sometimes look like ordinary English phrases. Cobol has implemented very powerful tools for working with large amounts of data stored on various external media. A lot of applications have been created in this language, which are actively exploited today. Suffice it to say that Cobol programmers in the US are among the highest paid employees.

Algol. Compiled language created in 1960. It was intended to replace Fortran, but due to its more complex structure, it was not widely used. In 1968, the Algol-68 version was

created, which, in terms of its capabilities, is still ahead of many programming languages, but due to the lack of sufficiently efficient computers, it was not possible to create good compilers for it in a timely manner.

Pascal. The Pascal language, created in the late 70s by Niklaus Wirth, the founder of many ideas of modern programming, in many ways resembles Algol, but it tightens a number of requirements for the structure of the program and has features that allow it to be successfully used when creating large projects.

Basic. There are both compilers and interpreters for this language, and it ranks first in popularity in the world. It was created in the 60s as an educational language and is very easy to learn.

C. This language was created in the Bell Labs and was not originally considered as a mass language. It was planned to replace assembler in order to be able to create equally efficient and compact programs, and at the same time not depend on a particular type of processor. C is in many ways similar to Pascal and has additional facilities for direct memory manipulation (pointers). Many applied and system programs and a number of well-known operating systems (UNIX) were written in this language in the 70s.

C++. This language is an object-oriented extension of the C language created by Bjarne Stroustrup in 1980. A lot of new powerful features, which made it possible to dramatically increase the productivity of programmers, were superimposed on a certain low-level inherited from the C language, as a result of which the creation of complex and reliable programs required a high level of professional training from developers.

Java. The language was created by Sun in the early 90s based on C++. It is designed to simplify the development of applications based on C++ by eliminating all low-level features from it. But the main feature of the language is compilation not into machine code, but into platform-independent bytecode (each command occupies one byte). This bytecode can be executed using an interpreter, the Java Virtual Machine, versions of which have been created for any platform. Due to the presence of many Java machines, Java programs can be ported not only at the source code level, but also at the binary bytecode level, which is why Java today ranks second in popularity in the world after BASIC.

Particular attention in the development of this language is paid to two areas: support for all kinds of mobile devices and microcomputers embedded in household appliances (Jini technology) and the creation of platform-independent software modules that can run on servers in global and local networks with various operating systems (Java Beans technology). So far, the main drawback of this language is its low performance, since the Java language is interpreted.

Database programming languages. This group of languages differs from algorithmic languages primarily in the tasks they solve. A database is a file (or a group of files) that is an ordered set of records that have a uniform structure and are organized according to a single template (usually in a tabular form). A database can consist of several tables. It is convenient to store various information from directories, file cabinets, accounting journals,

etc. in databases. The first databases appeared a very long time ago, as soon as there was a need to process large amounts of information and select groups of records according to certain criteria. To do this, the Structured Query Language (SQL) was created. It is based on powerful mathematical theory and allows you to perform efficient database processing by manipulating not individual records, but groups of records. DBMS (Database Management Systems) have been developed to manage large databases and process them efficiently.

In almost every DBMS, in addition to supporting the SQL language, there is also a unique language that is focused on the features of this DBMS and is not portable to other systems. With the advent of personal computers, so-called desktop DBMS were created. The ancestor of modern database programming languages for PCs is considered to be the dBase II DBMS, the language of which was interpreted. Then compilers were created for it, FoxPro and Clipper DBMS appeared, supporting dialects of this language. Today, similar but incompatible versions of the dBase family of languages are implemented in Microsoft's Visual FoxPro and Inprise's Visual dBase.

Programming languages for the Internet.

With the active development of the global network, many implementations of popular programming languages have been created, adapted specifically for the Internet. All of them differ in characteristic features: languages are interpreted, interpreters for them are distributed free of charge, and the programs themselves are in source texts. Such languages are called scripting languages. Let's briefly describe some of them.

HTML. A well-known language for paperwork. It is very simple and contains commands for forming text, adding pictures, setting fonts and colors, organizing links and tables. All Web pages are written in HTML or use its extensions.

Perl. In the 80s, Larry Wall developed the Perl language. It was conceived as a tool for efficient processing of large text files, generation of text reports and task management. Perl is much more powerful than languages like C. It includes many commonly used functions for working with strings, arrays, all kinds of data conversion tools, process management, system information, etc.

Tel/Tk. In the late 80s, John Austyraut came up with the popular Tel scripting language and the Tk library. In Tel, he tried to realize the vision of an ideal scripting language. Tel is focused on automating routine processes and consists of powerful commands designed to work with abstract untyped objects. It is independent of the type of system and at the same time allows you to create programs with a graphical interface.

VRML. In 1994, the VRML language was created to organize virtual three-dimensional interfaces on the Internet. It allows you to describe in text form various three-dimensional scenes, lighting and shadows, textures (coverings of objects), create your own worlds, travel around them, "fly around" from all sides, rotate in any direction, scale, adjust lighting, etc.

Modeling languages. When creating programs and forming database structures, formal ways of representing them are often used - formal notations, with which you can visually

represent (depict with the mouse) database tables, fields, program objects and the relationship between them in a system that has a specialized editor and generator source texts of programs based on the created model. Such systems are called CASE systems. They actively use IDEF notations, and recently the graphical modeling language UML is gaining more and more popularity.

Other programming languages.

PL/1. In the mid-60s, IBM decided to take the best of Fortran, Cobol, and Algol. As a result, in 1964, a compiled programming language was born, which was called Programming Language One. Many unique solutions were implemented in this language, the usefulness of which can only be appreciated 33 years later, in the era of large software systems. In terms of its capabilities, PL/1 is much more powerful than many other languages (C, Pascal). For example, PL/1 has a unique ability to specify the accuracy of calculations - even C++ and Java do not have it. This language continues to be supported by IBM today.

LISP. The interpreted programming language, created in 1960 by John McCarthy, is focused on the data structure in the form of a list and allows you to organize the efficient processing of large amounts of textual information.

Prolog. Created in the early 70s by Alan Colmeroe. The program in this language, which is based on the mathematical model of the theory of predicate calculus, is built on a sequence of facts and rules, and then a statement is formulated that Prolog will try to prove using the introduced rules. A person only describes the structure of the problem, and the internal "engine" of Prolog itself looks for a solution using search and matching methods.

Ada. Named for Lady Augusta Ada Byron, daughter of the English poet Byron and his distant relative Anabella Milbank. In 1980, hundreds of experts from the US Department of Defense selected this particular language from 17 options, developed by a small group led by Jean Ishbia. At that time, it met all the requirements of the Pentagon, and today tens of billions of dollars have been invested in its development. The structure of the language itself is similar to Pascal. It has means of strictly restricting access to various levels of specifications, the power of control structures has been brought to the limit.

Modern approach to programming.

The birth of fifth-generation languages occurred in the mid-1990s. They also include a system for automatically creating application programs using visual development tools, without programming knowledge. The main idea that is embedded in these languages is the ability to automatically generate the resulting text in universal programming languages (which then needs to be compiled). Instructions are entered into the computer in the most visual form using methods that are most convenient for a person who is not familiar with programming. Such languages are called the language or system of OOP (object-oriented programming), including a visual environment for creating a program interface.

A new generation of programming languages implement programs and are controlled by graphical operating systems of the Windows and Linux family and are distinguished by a modern approach to the description of algorithms, namely, they represent tools for

accelerated program development. First of all, this is an object-oriented approach to creating application projects on the one hand, and the efficiency of describing system programs on the other. This is dictated by both the capabilities of hardware and the development of modern programming technologies. Modern OOP languages include Delphi, Visual C++, C++Builder, C# systems. These programming systems provide not only the possibility of implementing task algorithms, but also a set of tools for creating interfaces for ongoing projects and forming libraries of application objects.

The creator of C++ (C with classes) - Bjarne Stroustrup, an employee of the famous AT & T Bell Labs, where the UNIX operating system and the C language were developed, came up with one of the most complex programming languages (in the early 1980s). For a beginner, C++ can be a tough nut to crack, since both its syntax and the principles of building programs differ from the languages usual for students and everyone who starts programming. But the language itself is more than worth the effort spent on its study. Structural object-oriented programming in the C++Builder system uses an object-oriented language (C++), which combines, on the one hand, the expressive power and ease of programming characteristic of 4GL languages (4th generation languages), and on the other hand, the efficiency of the 3GL language. Programmers can immediately start producing working applications without having to learn Windows event programming. C++ fully supports advanced programming concepts (encapsulation, inheritance, polymorphism, and event management). The C++ programming language, like other OOP languages, assumes a trinity of data, methods, and events that describe each program object. A program object is a specially designed fragment that contains these components.

Data is the objects of the program. (values that are processed) Methods are procedures that define actions on objects. Properties characterize the features of the objects of a particular program. Objects increase the performance and quality of programs and have the properties of functionality and indivisibility. Therefore, objects can be transferred from one program to another. Programming in the OOP language involves the creation of objects of a certain type (i.e. classes) with a set of methods and properties, and their use not only in the application being developed, but also in other applications.

C++Builder6 supports OLE(object linking and embedding), DDE, and VBX.

This is a very important feature of the language for developers in the Windows environment, because in Windows applications already developed, the programmer can integrate what was developed using C++.

Purpose and possibilities of C++ BUILDER.

The C++ BUILDER programming language is a further development of the C, C++ languages. The C(SI) language was developed by Bell Labs employee Dennis Ritchie in 1972 while working with Ken Thompson on the Unix operating system. The author's goal was to create a convenient and useful language. A new software product with phenomenal characteristics has become a powerful tool for practicing programmers. Further development of the language using new approaches to programming led to the emergence

of new versions of the language, such as Turbo C, C ++, Visual C ++, C # (C Sharp). Developed by Borland Corporation Microsoft C++Builder6 today is widely used as an OOP tool.

C++Builder6 is a combination of several key technologies:

- High performance compiler to native code
- Object-oriented component model
- Visual (and, therefore, high-speed) building applications from software prototypes.
- Scalable tools for building databases.

C++Builder6 is a programming system for building programs with accelerated development tools. This acceleration is achieved due to two characteristic features: the visual design of forms and the extensive use of the library of visual components.

Programming in C++Builder6 includes:

- Integrated environment.(ICE)
- Application design.
- Creation of projects.
- Compiling applications.
- Debugging and deployment of applications.
- Application of component libraries.
- Use of the object module.
- Using the basis of objects.

Visual form design saves the programmer from many aspects of developing the program interface, since C++Builder6 automatically prepares the necessary program blanks and the corresponding resource file. The programmer uses a special window called the form window as a prototype of the future program window and fills it with components that implement the necessary interface properties (all sorts of lists, buttons, scrollbars, etc.). After placing the next component on the form, C++Builder6 automatically inserts a link to the component into the module associated with the form and corrects the special form description file with the CFM extension, which is converted into a Windows resource file after compilation.

The Visual Components Library provides the programmer with a huge variety of program blanks created by the C++Builder6 developers, which are immediately or after a simple setup ready to work within the program. Components are characterized by an important property: they include the program code and all the data necessary for its operation. The power and flexibility of the C++Builder6 language makes it a modern object-oriented language suitable for creating programs of any complexity. Numerous components provide a solution to a wide variety of tasks.

Features of the C++Builder6 version.

Unlike other programming languages, C++Builder6, like other previous versions, is typed, which makes it possible to detect errors during compilation. The ability to use Assembler as a machine-oriented language attracts system programmers. The feature of the language includes many technologies that simplify the creation of programs for databases

and the Internet. This technology allows you to re-encode any program and send it over the network, so that it can run not only Windows, but also Linux, Solaris, OSMac. C++Builder6 has a CASE tool, an automated programming tool. Innovations have been introduced in the technology of creating applications on the Internet.

Conclusions: The C++Builder6 language is a high-level language with extensive use of object-oriented programming tools and object typing.

C++Builder6 is a modern, popular and efficient high-level programming language. C++Builder6 is a powerful and flexible language for solving a wide range of problems, both technical and any other area, including graphics.

The C++Builder6 language uses the module insertion capabilities of the Accembler language.

With an object-oriented approach, a program is a description of objects, their properties (or attributes), collections (or classes), relationships between them, ways of their interaction and operations on objects (or methods).

The undoubted advantage of this approach is the conceptual proximity to the subject area of arbitrary structure and purpose. The mechanism of inheritance of attributes and methods allows you to build derivative concepts based on basic ones and thus create a model of an arbitrarily complex subject area with specified properties.

Another theoretically interesting and practically important property of the object-oriented approach is the support for the mechanism for processing events that change the attributes of objects and model their interaction in the subject area.

Moving along the class hierarchy from general concepts of the subject area to more specific ones (or from more complex to simpler ones) and vice versa, the programmer gets the opportunity to change the degree of abstractness or concreteness of the view of the real world he is modeling.

The use of previously developed (perhaps by other teams of programmers) libraries of objects and methods can significantly save labor costs in the production of software, especially typical software.

Objects, classes and methods can be polymorphic, which makes the implemented software more flexible and versatile.

The complexity of an adequate (consistent and complete) formalization of the object theory gives rise to difficulties in testing and verifying the created software. Perhaps this circumstance is one of the most significant shortcomings of the object-oriented approach to programming.

The most famous example of an object-oriented programming language is the C++ language, which developed from the imperative C language. Its direct descendant and logical continuation is the C# language, which is studied in this course. Other examples of object-oriented programming languages: Visual Basic, Java, Eiffel, Oberon.

The transition from a structured procedural approach to object-oriented programming, like the transition from low-level programming languages to high-level languages, requires

significant learning costs. Naturally, the price for this is an increase in the productivity of programmers in the design and implementation of software. Another advantage of OOP over the imperative approach is a higher percentage of code reuse that has already been developed.

At the same time, unlike previous approaches to programming, the object-oriented approach requires a deep understanding of the basic principles, or, in other words, the concepts on which it is based. The fundamental concepts of OOP usually include data abstraction, inheritance, encapsulation, and polymorphism.

Often, in practical and training courses on programming, students do not have a clear mathematical foundation for the formation of a sufficiently complete and clear understanding of the basics of OOP. The advantage of the proposed course is that the sections of computer science already studied in the first part of the course (for example, lambda calculus and combinatorial logic) allow you to form a deep and accurate understanding of the fundamental concepts of object-oriented programming. In particular, the concept of abstraction - the basic operation of the lambda calculus - is already well known to us. Microsoft Corporation has proposed an innovative component-oriented approach to programming, which is the development of the object-oriented direction.

According to this approach, the integration of objects (possibly of a heterogeneous nature) is based on interfaces that represent these objects (or program fragments) as independent components. This approach greatly facilitates the writing and interaction of software "molecules"-components in a heterogeneous design and implementation environment. It standardizes the storage and reuse of software project components in a distributed networked computing environment, where different computers and users exchange information, for example, interacting within a research or business project.

A significant advantage should be considered the possibility of practical implementation of the principle "every entity is an object" in a heterogeneous software environment. Much of this has been made possible by the improved, generic Common Type System, or CTS, which will be discussed in more detail in a later lecture.

Strict hierarchical organization of spaces for types, classes and names of program entities allows to standardize and unify the implementation.

A new approach to integrating application components in the Internet computing environment (or so-called web services) makes it possible to accelerate the creation of applications for a wide range of users. The .NET Framework generic interface provides integrated design and implementation of application components developed according to different programming approaches.

Speaking of .NET as a technological platform, one cannot fail to note the fact that it provides simultaneous support for the design and implementation of software using various programming languages. At the same time, dozens of programming languages are supported, ranging from the very first (in particular, COBOL and FORTRAN) to modern ones (for example, C # and Visual Basic). Early programming languages are still actively

used today, in particular, to ensure compatibility with previously created business-critical applications (say, COBOL was very widely used to create applications that support financial activities).

The use of web services technology is not just a fad on the Internet, but a real (and perhaps the most acceptable) opportunity to ensure scalability and interoperability of applications. Scalability is understood as the possibility of a smooth increase in the response time of a software system to a request with an increase in the number of simultaneously working users; in the case of web services, scalability is realized by distributing computing resources between the server that runs the application (or stores data) and the user's computer.

Interoperability refers to the possibility of integrated processing of heterogeneous data coming from heterogeneous application programs. It is thanks to interoperability that it is possible to unify the interaction of users through an application with the operating system based on a specialized application programming interface, or API (Application Programming Interface).

It should also be noted that the new .NET technology is not only in demand by the world community, but also officially recognized, which is reflected in the relevant ECMA (European Computer Manufacturers Association) standards.

Now let's look at the .NET tool capabilities as a means of designing and implementing software, that is, in fact, programming in the broadest sense of the word.

First of all, it is necessary to note the support of the multilingual application development environment CLR (Common Language Runtime).

This capability comes from the Common Language Infrastructure, or CLI, which supports the development of software components in a variety of programming languages.

At the same time, the undoubted advantage for programmers is that they can develop (or modify) software in the most suitable programming language. Here, one should take into account the nature of the task (say, recursion or symbolic processing is more naturally implemented in a functional programming language, and the formalization of the structure of the subject area in an object-oriented language). In addition, it is necessary to take into account the experience of programmers in the development team and the programming language in which the system was originally created.

We note two more important facts. Firstly, the main developer services provided by the .NET environment (debugging, code analysis, etc.) are independent of a particular programming language, and therefore, programmers do not need to re-learn the features of the development environment if they want to switch from one language to another. Secondly, despite the fact that not all programming languages are supported by .NET yet, it is possible to independently develop a translator for any programming language, and this does not cause difficulties even for programmers with little or no professional training in the field of compiler development. The COM Component Object Model is the primary Microsoft standard for component-based software design and implementation. Today it is the most developed and, perhaps, the most successful model in practical terms, which

provides the ability to initialize and use components both within a single process, and between processes, or between computers, regardless of the implementation language. The COM model is supported in the .NET ideology for a number of programming languages (C#, SML, Visual Basic, C++, etc.), it is the basis for ActiveX, OLE, and many other Microsoft technologies.

Unlike COM, the Java Beans model, Sun Microsystems' base standard for components, turns out to be implementation language dependent. First of all, we list the fundamental concepts that characterize each of the approaches. Then we compare these approaches with each other in order to find analogies between them.

In the object-oriented approach, the concepts of class and interface, in particular, should be considered key. Note that in the component-oriented approach, these concepts are also system-forming.

In this case, a class is understood as a basic entity, defined as a set of elements.

An interface is a set of semantically related abstract elements. For a component-oriented approach, the concept of an interface is of paramount importance, since it is solely through this mechanism that a client in a COM architecture can directly interact with a COM class. Note that interfaces increase code security, since interaction with an object does not occur directly, but through a pointer (reference). The concepts of a property (as an attribute of an object) and a method (as an operation on an object), as well as an event mechanism (correlations over objects of the subject area), are characteristic of both approaches.

Fundamentally new is the presence in the COM model of assemblies - self-sufficient units of information for installing and deploying software products.

In general, the component approach is more convenient from a practical point of view, although the mechanisms implemented in it are fundamentally comparable with the capabilities of OOP.

Despite the innovations listed above in the field of theory, technology and practical implementation, due to the scale of the ideology and the novelty of the problem under study, the .NET approach is not without certain shortcomings, most of which, apparently, are of a temporary nature. We list, in our opinion, the most significant of them. Firstly, the developers note rather high requirements for hardware (in particular, the amount of RAM must be at least 256 megabytes, the free hard disk space for working with Microsoft Visual Studio .NET is at least 10 gigabytes).

In addition, non-commercial versions of Microsoft software products, which often provide significant new features, are not robust enough; Some of the new software features are not fully documented.

Support for a number of theoretically interesting and practically useful programming languages is limited. Because a number of compilers for programming languages are provided by third-party developer companies or non-profit institutions, the results of their activities are subject to control and customization with restrictions.

The set of software tools that implement the .NET approach (including compilers for programming languages) has not been fully ratified according to international standards.

Undoubtedly, .NET is an outstanding achievement of the modern programming industry. Suffice it to say that Microsoft considers .NET to be its strategic ideology and technological platform for the next decade.

The undoubted qualitative superiority over the existing means of computer-aided design and rapid implementation of applied software is achieved due to the following main factors:

- interoperability and cross-language interaction;
- multi-level, flexible and reliable security policy;
- integration with web services technology;
- simplification of the procedure for deploying and using the created software.

Despite some incompleteness of the solution for wide commercial use due to the conceptual novelty and grandeur of the project, the .NET approach certainly has a significant impact on the commercial programming industry as a whole and contributes to a radical improvement of the industry. C++Builder supports the basic principles of object-oriented programming - encapsulation, polymorphism and multiple inheritance, as well as new specifications and keywords in the C++Builder language standard provides high performance when compiling and building 32-bit applications for modern Windows 95 operating systems and Windows NT, including OLE client-server interaction.

The system even displays the time spent on the main stages of building programs. The resulting programs are well optimized in terms of execution speed and memory consumption. While low-level debugging is fully integrated into the C++Builder environment, debugging also took some getting used to. Form designer. The Object Inspector and other tools remain available while the program is running, so you can make changes while debugging.

C++Builder comes in three flavors: Standard. Professional (for professional developers focused on network architecture) and Client/Server Suite (for developing systems in the client/server architecture). The last two options complement the standard visual component source code, a multi-scale data dictionary, new SQL query language features for databases, the Internet Systems Support Package, a program monitoring service, and a number of other tools.

Experiments with test programs within the framework of the standard version formed the basis of the material presented here. While testing the system, I moved several applications previously written in Borland C++ version 4.5 to C++ Builder.

Thanks to the visual components, the "code shell" of processing Windows messages and resource files has disappeared from programs, and only meaningful code remains. The application user interface has acquired a complete professional look.

While C++Builder appears to be a very reliable system, the corporation has yet to refute the common claim that every well-oiled program (including commercial ones) has at least one bug. Apparently, it is precisely this desire that explains the excessive, in my opinion,

haste with advertising the "improved and extended" version of Borland C ++ version 5.02. C++ Builder supports connection with various databases of 3 types:

dBASE and Paradox: Sybase, Oracle, InterBase and Informix; Excel, Access, FoxPro and Btrieve. The BDE (Borland Database Engine) makes maintaining database links surprisingly simple and transparent. The Database Explorer allows you to display relationships and database objects graphically. Using database components, I built an electronic notebook from a dBASE table in half an hour of work on a computer. Inheritance of ready-made forms and their "adjustment" to specific requirements significantly reduce the time spent on solving such problems.

The C++ Builder Help Desk has helped me in this and many other similar situations. There is a complete description of each control component, including lists of properties and methods, as well as numerous examples. The presentation of the material has been greatly improved and systematized thanks to the information I gleaned from the help desk.

Lecture 11

Object programming systems. Basic constructions of languages and features of programming in the system. Classes, methods and properties

Plan:

1. C++ Builder 6 Integrated Environment
2. Basic C++ constructs
3. Keywords C++
4. Simple, complex types and their scope
5. Basics of programming in visual mode.
6. Classes, methods and properties

Keywords: *classes, encapsulation, polymorphism, multiple inheritance, events.*

C++Builder is a complete C++ programming environment for developing Windows applications. The system's integrated environment provides faster visual design and productivity of reusable components. Using the graphical tools of the integrated environment, you can quickly master the techniques of object-oriented programming - encapsulation, polymorphism, multiple inheritance, and the syntax and code structure of the programs developed by the environment are very attractive. Together with Delphi (the most popular related system), C++Builder is able to compete with any software products. The unique relationship between Delphi and C++Builder allows you to easily move from one development system to another when creating an application. C++Builder includes the C++ language, compiler, Integrated Development Environment (IDE), debugger, and various tools. C++Builder contains a set of common controls, access to the Windows API, the Visual Component Library (VCL), components and tools for working with databases.

C++Builder adds to the process of programming in C++ the ability to quickly visually develop the application interface. In addition to the OWL (Object Windows Library) and MFC (Microsoft Foundation Classes) libraries, it uses the VCL library and allows you to include dialogs with the user in the form, leaving the developer to implement only the functional part that embodies the algorithm for solving the problem.

C++Builder shares a class library with Delphi, some of which remain written in Object Pascal. Thanks to this, as well as the inclusion of C++ and Object Pascal compilers in C++Builder, applications can use components and code written in Object Pascal, as well as Delphi forms and modules.

C++Builder components. Creating an application's user interface consists of adding objects, called components, to the form window. C++Builder allows the developer to create their own components and customize the Component Palette.

Components are divided into visible (visual) and invisible (non-visual). Visual components appear at both runtime and design time. Non-visual components appear at design time as icons on the form. They are not visible at runtime, but have functionality. To add a component to a form, you can select the required component in the Components Palette with the mouse and left-click in the desired place of the designed form. The component will appear on the form, and then it can be moved and changed.

Each C++ Builder component has three characteristics: properties, events, and methods. The Object Inspector automatically shows the properties and events that can be used with the component. Properties are attributes of a component that define its appearance and behavior.

The Object Inspector displays published component properties in the properties page and is used to set published properties at design time. To change the properties of a component at runtime, you need to add the appropriate code. In addition to published properties, components can have public properties that are only available at runtime.

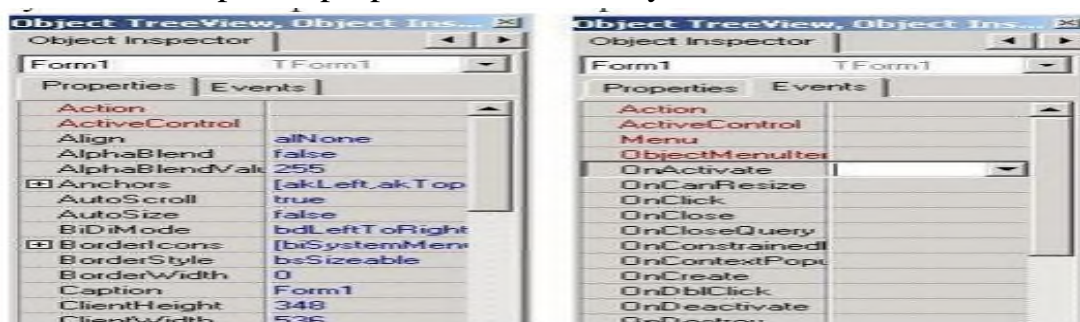


Fig.1. Object Inspector window

Developments. The Events page of the Object Inspector shows a list of events recognized by the component and fired when the component's state changes. Each component instance has its own set of functions - event handlers. By creating an event handler, you are instructing the program to execute the specified function if the event occurs. To add an event handler, select the component, then open the Events page of the Object Inspector and double-click next to the event. This will cause C++Builder to generate an empty function

text with the cursor where the code should be entered. Next, you need to enter the code that should be executed when this event occurs.

Development environment (IDE). C++ Builder is an application whose main window contains a menu (top), toolbar (left), and Component Palette (right). In addition, when you launch C++ Builder, the Object Inspector window and ObjectTreeView window (left) and the new application form (right) appear. Below the application form window is the Code Editor window.

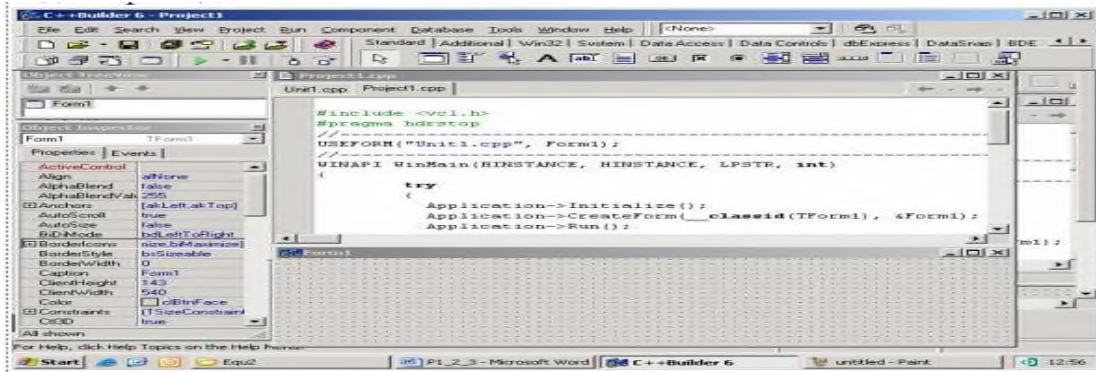


Fig.2. IDE main window

Building applications in C++Builder. The first step in developing a C++ Builder application is to create a project. To create a new project, select the menu item File|New|application.

C++ Builder creates a Project.bpr file, as well as a Project.cpp project head file that contains the WinMain() function. The WinMain() function in Windows applications is used instead of the main() function. When a new form is added, C++Builder updates the project file and creates the following additional files:

- form file with .dfm extension containing information about the form;
- module file with .cpp extension containing C++ code;
- header file with .h extension, containing a description of the form class.

To compile the current project, select the Compile menu item. To compile the project and create an executable file, select Run from the Run menu. As a result of execution, the following form will be obtained:



Fig.3. Application execution result

Project file structure. For each C++Builder application, a Project.bpr project xml file and a resource file are created. Another file is the project's head file containing the WinMain() function, which is generated when the File|New Application menu item is selected. Initially, this file is named Project1.cpp by default. If forms and modules are added during application

development, C++Builder updates the file. To view the file, select the menu item Project|View Source.

The project head file has a certain set of key elements:

- The `#include <vcl.h>` preprocessor directive is intended to include a header file that refers to the VCL class definitions.
- The `#pragma hdrstop` directive is intended to limit the list of header files available for precompilation.
- The `USEFORM` directive shows the modules and forms used
- in project.
- The compiler's `USERES` directive attaches resource files to the executable. When you create a project, a `.res` resource file is automatically created to store cursors, icons, and other resources.
- `Application->Initialize()`. This statement initializes the application.
- `Application->CreateForm()`. This statement creates an application form. Each form in the application has its own statement
- `CreateForm`.
- `Application->Run()`. This statement starts the application.
- The `try...catch` block is used to terminate the application gracefully if an error occurs.

A typical project head file looks like this:

```
//Project1.cpp -----  
#include <vcl.h> #pragma hdrstop USERES("Project1.res");  
USEFORM("Unit1.cpp", Form1); //-----
```

```
-----  
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)  
{  
try  
{  
Application->Initialize(); Application->CreateForm(__classid(TForm1), &Form1);  
Application->Run();  
}  
catch (Exception &exception)  
{ Application->ShowException(&exception); } return 0;  
}
```

The structure of the Project1.bpr file. The `Project1.bpr` file represents an XML project (C++Builder XML Project) that contains a description of the application to be created. This is a text file containing instructions on which files should be compiled and linked into the project, as well as the paths to the directories to be used.

Module structure. The module contains the implementation of the functional part of the object in the C++ language and by default is the `Unit1.cpp` file. Each such file is compiled

into an object file with an .obj extension. When a new form is added to a project, a new module is generated.

The name of the module source file and the shape file (*.dfm) must be the same. When creating an event handler in the text of the module, an event handler function template is generated, into which the code is entered that is executed when the handled event occurs.

Below is the module text generated for the original form:

```
//Unit1.cpp -----  
#include <vcl.h> #pragma hdrstop #include "Unit1.h"  
#pragma package(smart_init) #pragma resource "*.dfm"  
TForm1 *Form1;// a pointer to an object  
//-----  
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner) { } //
```

Header file structure. The header file (a file with the .h extension, Unit1.h by default) is generated when a new module is created and contains a description of the form class. Such descriptions are generated automatically and change when new components are added to the form or new event handlers are generated. The header file contains the interface, and the module itself contains the implementation of the methods.

When components are removed from a form, their descriptions are removed from the header file. Renaming components changes their descriptions in the header file, as well as the names and descriptions of event handlers. However, this does not change references to these components and event handlers used in other functions. Because of this, it is recommended that you rename components and event handlers as soon as they are created, before they are referenced.

A module may contain classes and functions that are not described in the header file, but their visibility in this case is limited by this module.

Below is the header file for the original form:

```
//Unit1.h-----  
#ifndef Unit1H #define Unit1H  
//-----  
#include <Classes.hpp> #include <Controls.hpp> #include <StdCtrls.hpp> #include  
<Forms.hpp>  
class TForm1 : public TForm  
{  
__published:  
// IDE-managed Components  
private: // User declarations  
public:
```

```
// User declarations
__fastcall TForm1(TComponent* Owner);
}; //-----
extern PACKAGE TForm1 *Form1; #endif
//-----
```

Shape file. The form is one of the most important elements of a C++ Builder application. The form editing process occurs when components are added to the form, their properties are changed, and event handlers are created. When a new shape is added to a project, three separate files are created:

- 1) the module file (*.cpp) contains the code of the methods associated with the form;
- 2) the header file (*.h) contains the description of the form class;
- 3) the form file (*.dfm) contains information about the published (available in the Object Inspector) properties of the components contained in the form.

When a component is added to a form, the header file and the form file are modified. When you edit the properties of a component in the Object Inspector, those changes are saved in the shape file. Although C++ Builder saves the *.dfm file in binary format, its contents can be viewed using a code editor. To do this, right-click over the form and select View as Text from the context menu of the form.

The most important feature of C++Builder is the automatic generation of program code. When you add a component to the form, the declaration of the object of the class of this component appears in the text of the Unit1.h file. For example, transferring a TButton button component to an empty form will generate a declaration of the Button1 object, and defining an OnClick event will generate a declaration of the Button1Click method, which is the handler for this event.

Classes, Components, and Objects.

C++Builder uses the concept of components - special classes that contain, in addition to the usual data members of the class and methods, also properties and events. Properties and events allow you to manipulate the appearance and functional behavior of components both at the design stage of the application and during its execution.

Bean properties are an extension of the concept of data members and use the __property keyword for declaration. With the help of events (events), the component informs the user that it has been affected in some way. Event handlers are methods that implement the program's reaction to the occurrence of events. Typical events are pressing a button or a key on the keyboard. Components have a number of features:

- All components are direct or indirect descendants of the TComponent class. In this case, the inheritance hierarchy is as follows: Tobject->Tpersistent-> Tcomponent->Tcontrol->... .

- Components are used directly, they cannot serve as base classes for building new subclasses.

- Components are placed only in dynamic memory using the new operator.
- Components can be added to the Component Palette and manipulated with the Form Editor.

Class Development

C++Builder allows you to declare a base class that encapsulates property, data, method, and event names. Each declaration within a class defines an access privilege to class names depending on which section the name appears in. Each section begins with one of the keywords: private, protected and public, which determine the accessibility of the elements of the corresponding section. Class declarations and method definitions are usually stored in separate files (with extensions .h and .cpp, respectively). The following example shows that if methods are defined outside the class, then their names should be qualified with the class name. C++Builder allows you to declare a derived class that inherits the properties, data, methods, and events of all its predecessors in the class hierarchy, and can also declare new characteristics and overload some of the inherited functions. You can declare a derived class like this:

```
class derivedClass : [<access specifier>] parentClass { private:
<private member data> <private methods> protected:
<protected data members>
<protected methods> public:
<public properties> <public data members> <public constructors> <public destructor>
<public methods>
__published:
<published properties> <published data members> <declarations of friend functions> };
```

Note the introduction of a new section with the __published keyword, an addition that C++Builder introduces to the ANSI C++ standard for declaring published members of coclasses. This section differs from the public section only in that the compiler generates RTTI (Runtime Information) information about the object's properties, data members, and methods, and C++Builder arranges for this information to be passed to the Object Inspector.

When a class is derived from a base class, all base class names in the derived class automatically become private by default (unless an access specifier is specified when inheriting). But this can be changed by specifying the following access specifiers

When inheriting the base class:

- protected. Inherited (i.e. protected and public) base class names become protected in instances of the derived class.
- public. The public names of the base class and its predecessors will be public in instances of the derived class, and any protected ones will remain protected.

C++Builder uses the concept of components - special classes that contain, in addition to the usual data members of the class and methods, also properties and events. Properties and

events allow you to manipulate the appearance and functional behavior of components both at the design stage of the application and during its execution.

Bean properties are an extension of the concept of data members and use the `__property` keyword for declaration. With the help of events (events), the component informs the user that it has been affected in some way. Event handlers are methods that implement the program's reaction to the occurrence of events. Typical events are pressing a button or a key on the keyboard. Components have a number of features:

- All components are direct or indirect descendants of the `TComponent` class. In this case, the inheritance hierarchy is as follows: `TObject->TPersistent->TComponent->TControl->... .`
- Components are used directly, they cannot serve as base classes for building new subclasses.
- Components are placed only in dynamic memory using the `new` operator.
- Components can be added to the Component Palette and manipulated with the Form Editor.

Class Development

`C++Builder` allows you to declare a base class that encapsulates property, data, method, and event names. Each declaration within a class defines an access privilege to class names depending on which section the name appears in. Each section begins with one of the keywords: `private`, `protected` and `public`, which determine the accessibility of the elements of the corresponding section.

Consider an example of a class declaration. Note the declaration of the `Count` property in the `protected` section, and the `SetCount` method, which implements writing to the given `Fcount`, in the `private` section.

`C++Builder` allows you to declare a derived class that inherits the properties, data, methods, and events of all its predecessors in the class hierarchy, and can also declare new characteristics and overload some of the inherited functions. You can declare a derived class like this:

```
class derivedClass : [<access specifier>] parentClass { private:
<private member data> <private methods> protected:
<protected data members>
<protected methods> public:
<public properties> <public data members> <public constructors> <public destructor>
<public methods>
__published:
<published properties> <published data members> <declarations of friend functions> };
```

Note the introduction of a new section with the `__published` keyword, an addition that `C++Builder` introduces to the ANSI C++ standard for declaring published members of co-

classes. This section differs from the public section only in that the compiler generates RTTI (Runtime Information) information about the object's properties, data members, and methods, and C++Builder arranges for this information to be passed to the Object Inspector.

When a class is derived from a base class, all base class names in the derived class automatically become private by default (unless an access specifier is specified when inheriting). But this can be changed by specifying the following access specifiers when inheriting the base class:

- `protected`. Inherited (i.e. `protected` and `public`) base class names become `protected` in instances of the derived class.
- `public`. The public names of the base class and its predecessors will be `public` in instances of the derived class, and any `protected` ones will remain `protected`.

Let's consider the application of the technique of extending and limiting characteristics using the example of creating button varieties when inheriting the base `TButtonControl` component from the Visual Components Library. The base class `TButtonControl` is capable of displaying the button as two nested rectangles using the parent `Draw` method: an outer frame and an inner filled area. To create a simple button without a border, you need to derive a `SimpleButton` class using `TButtonControl` as the parent and override the `Draw` method:

```
class SimpleButton: public TButtonControl { public:
SimpleButton(int x, int y) ;
void Draw() ; ~SimpleButton() { } };
SimpleButton::SimpleButton(int x, int y) : TButtonControl(x, y) { }
void SimpleButton::Draw() {outline->Draw();}
```

The only purpose of the object constructor for `SimpleButton` is to call the base class constructor with two parameters. It is the override of the `SimpleButton::Draw()` method that prevents the button's outline from being drawn (as happens in the parent class). To change the code of a method, you need to study it using the source code of the base `TButtonControl` component.

Let's create a button with an explanatory name. To do this, you need to build a derived `TextButton` class from the base `TButtonControl` and overload the `Draw` method with an extension of its functionality:

```
class Text { //Helper class public:
Text(int x, int y, char* string) { } void Draw() { }
};
class TextButton: public TButtonControl { Text* title;
public:
TextButton(int x, int y, char* title); void Draw();
~TextButton() { } };
TextButton::TextButton(int x, int y, char* caption): TButtonControl(x, y) {
```

```

title = new Text(x, y, caption);}
void TextButton::Draw () { TButtonControl::Draw() ; title->Draw() ;}

```

Control questions:

1. What files does a C++Builder application consist of, and how are these files organized?
2. What codes are automatically generated in the application's head file when this application is created?
3. What is the purpose of the Object Inspector windows?
4. What does a Form View (.dfm) file contain?
5. What is an event and how to create an event handler for a component?

Lecture 12

Logic programming technology. The logical structure of the program. Conditional, unconditional, and select statements

Plan:

- 1.Logical structure of the program
- 2.Conditional, unconditional statements
- 3.Cyclic computing processes

Key terms: *arithmetic operations, logical operations, operations with strings, relational operations, conditional operators, selection operators.*

Logical structure of the program

Operations are divided into arithmetic operations, logical operations, string operations, set operations, relational operations, and the @ operation (address obtaining operation). Relational and logical operations result in False or True logical constants. With their help, you can set various conditions for variants of complex branching algorithms.

Relation operations

The types of operands and the results of relational operations are given in Table 1.

Table 1

Operation	Action	Operand Types	Result Type
==	Equal	Compatible simple, pointer, string multiple boolean	logical
!=	Not equal	Compatible simple, pointer, multiple, boolean	logical
<	Less Than	Compatible simple, pointer, multiple, string boolean	logical

>	Greater than	Compatible simple, pointer, plural	logical
<=	Less than or equal to	Compatible simple, pointer, union	logical
>=	Greater than or equal to	Compatible prime, pointer, union	logical

Logical operations

The basic logical operations are the operations of logical addition OR (||), logical multiplication AND(&&), logical negation NOT(!). With the help of logical operations, you can compose complex logical expressions. The operands in such an expression are boolean variables, numeric variables, and functions linked by relational operations. The priority of performing operations in a complex logical expression is determined by the following order:

1. Arithmetic operations. (* / % + -)
2. Relationship operations (in order)
3. Logical operations.(! && ||)

The result of evaluating a logical expression will be the value of the logical constants true or false. It is these values that the conditional jump operators of the C++Builder 6 language react to when implementing programs of a branching structure.

Here are the truth tables for these operations, assuming that NOT is a unary operation, and all the others are operations of several operands.

Truth table for logical addition.

A B A || B

True	True	True
False	True	True
True	False	True
False	False	False

Truth table for logical multiplication.

A B A && B

True	True	True
False	True	False
True	False	False
False	False	False

Truth table for logical negation.

A ! A

true	True
false	True

Truth table for the logical negation operation or

A B A xor B

True	True	False
False	True	True
True	False	True
False	False	False

The types of logical operations are shown in Table 2.

Table 2

Operation	Action	Types of operands	Type of results
!	Inversion	logical	logical
&&	And (bit)	logical	logical
	Or	logical	logical
^	Excepting Or		

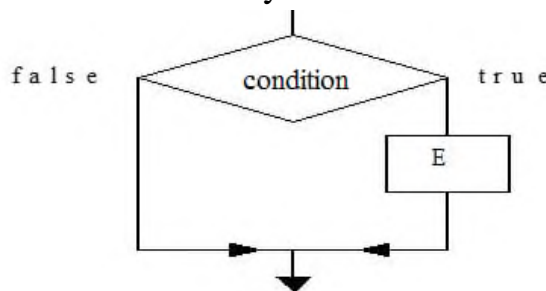
Conditional Jump Operators

Conditional statements allow you to select one of the compound statements to execute (or select none).

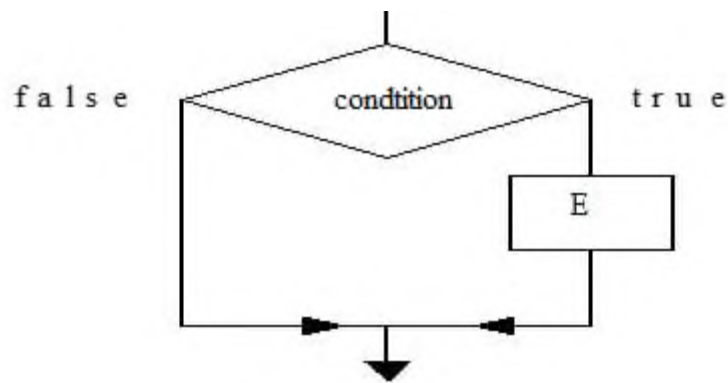
The expression must produce a result that has a standard Boolean type. If the expression evaluates to a true value (True), then the statement following the then keyword is executed. (short form of the statement)

If B then E

where B is a logical expression and E is any C++Builder 6 language operator.



If the expression evaluates to False and the else keyword is present, then the statement following the else keyword is executed. Such an operator is called a complete conditional operator.



If B then E; else E2 ; (full form).

A feature of the conditional operator is that it can have a nested structure. Moreover, the number of investments is unlimited.

An example program using a conditional statement.

```

//-----
#include <vcl.h>
#pragma hdrstop
#include "Z.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TForm1::Button1Click(TObject *Sender)
{
float x;
x=StrToFloat(Edit1->Text);
if(StrToFloat(Edit1->Text)>0)
{RadioGroup1->ItemIndex=0;
x=x*x*x; Label1->Caption="Otvet="+FloatToStr(x);}
if (StrToFloat(Edit1->Text)<0)
{RadioGroup1->ItemIndex=1;
x=x*x; Label1->Caption="Otvet="+FloatToStr(x);}
if(StrToFloat(Edit1->Text)==0)
{RadioGroup1->ItemIndex=2;
Label1->Caption="Otvet="+FloatToStr(x);}
}

```

```
}  
//-----
```

Variant operator (switch)

A switch statement consists of an expression (switch or selector) and a list of statements, each preceded by one or more constants (called choice constants). The switch (selector) must have an ordinal type (byte or word size). Thus, string type and long integer type are invalid switch types. All selection constants must be unique and have an ordinal type that is compatible with the radio button type.

The switch variant statement results in the execution of a statement preceded by a selection constant equal to the switch value or the selection range in which the selector value lies. If no such selection constant or such selection range exists and the default branch is present, then the statement following that keyword is executed.

Here are some examples of the variant operator:

S-character expression:

```
switch (s)  
{  
case '+': X = X+Y;  
        break;  
case '-': X = X-Y;  
        break;  
case '*': X = X*Y;  
        break;  
default s;  
}
```

The variant (selection) operator is used in those cases when, depending on the values of a variable or expression, called a selector, certain operators must be executed. Operators can be simple or compound. If there are only two options, then the conditional jump operator IF is used. But if there are many options, for example, ten? In such cases, you can use a nested IF structure, but the visibility and clarity of the program is lost. It is optimal to use the variant operator. The choice list is a set of label values and operators.

```
switch (n)  
{  
case n1: S1; break;  
case n2: S2; break;  
case n3: S3; break;  
default s;  
}
```


where *switch*, *case* - service words; n is a variable or expression called a selector;

n1 ,*n2*.. - labels

S1, *S2*, *S3* are variant operators.

The *break* statement breaks the execution of the *switch* and transfers control to the statement following the closing curly brace.

The *switch* statement transfers control to that statement *S*, one of the labels of which matched the value *N*. If the value of the selector *N* did not match any of the labels, then the statement following the reserved word *default* is executed.

Example 1. Write a program that prints the schedule for the number entered.

```
//-----  
#include <vcl.h>  
#include <iostream.h>  
#include <conio.h>  
#pragma hdrstop  
//-----  
#pragma argsused  
int main()  
{  
  Int n;  
  cin>>n;  
  switch ( n% 7)  
  {  
  case 1: cout<<" Physics,PI";  
          break;  
  case 2: cout<<"descriptv.,English";  
          break;  
  case 3: cout<<" 'informatics,Uzbek ";  
          break;  
  case 4: cout<<"introd to spec,lab physics ";  
          break;  
  case 5: cout<<" practice informatics,descriptv";  
          break;  
  case 6: cout<<" math, jurisprudence ";  
          break;  
  case 7: cout<<" week end ";  
          break;  
  default n;  
  }  
}
```

```

getch();
    return 0;
}
//-----

```

Example 2. in visual mode:

Calculate x^2 if $x > 0$, x^2 if $x < 0$, output x if $x = 0$.

Used components RadioGroup, Lable, Edit, Button

RadioGroup1: TRadioGroup

The Edit number depends on how much data the user needs to enter manually. Button1 - The button is used to calculate Y.

The RadioGroup component has a RadioGroup1.ItemIndex property, the value of which depends on the strings of variants of the specified conditions. This component shows how the way Y is calculated changes depending on the variable X entered by the user.

Program code:

```

/-----
#include "Unit1.h"
#include <vcl.h>
#pragma hdrstop
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"

//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    {
float x;
x=StrToFloat(Edit1->Text);
if(x>0)
RadioGroup1->ItemIndex=0;
else
if (StrToFloat(Edit1->Text)<0)

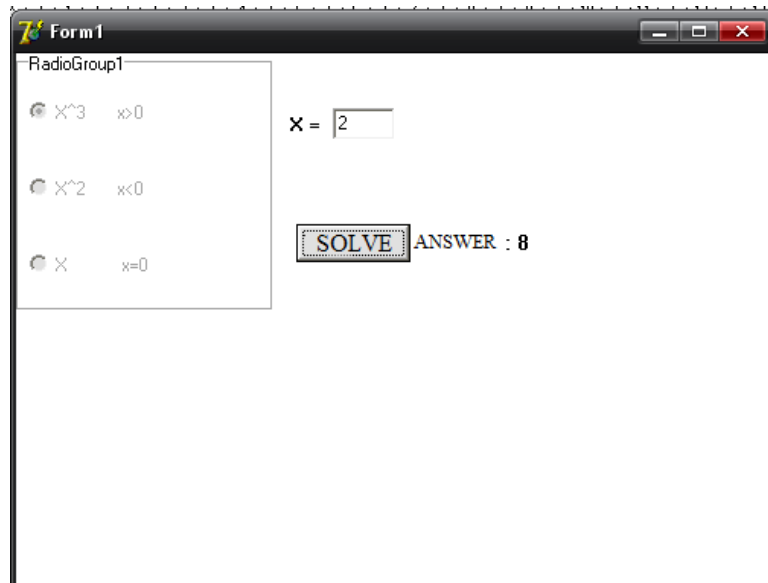
```

```

RadioGroup1->ItemIndex=1; else
RadioGroup1->ItemIndex=2;
Label1->Caption="Answer="+FloatToStr(x);
int n=RadioGroup1->ItemIndex ;
switch (n)
{
case 0: x=x*x*x; Label1->Caption="Answer="+FloatToStr(x);
        break;
case 1: x=x*x; Label1->Caption="x="+FloatToStr(x);
        break;
case 2: Label1->Caption="Answer="+FloatToStr(x);
        break;
}
}
}

```

Result:



Test questions:

1. What is logic programming?
2. What operators are used in logical operations?
3. In what situations are variant operators used?

Lecture 13

Components used in visual programming. Loop operators. Their different forms (parametric, conditional check before and after). Complex algorithms

When solving many problems, it becomes necessary to repeat a certain set of program statements. For these purposes, the C++Builder 6 language provides for the use of special loop operators.

Cyclic computing processes are divided into cycles with a known, unknown number of repetitions and the so-called nested cycles (cycles within a cycle). C++Builder 6 uses three types of cycle operators.

The loop statement specifies the repeated execution of certain statements. If the number of repetitions is known in advance, then the appropriate construct is the for statement. Otherwise, the while or do while statements should be used.

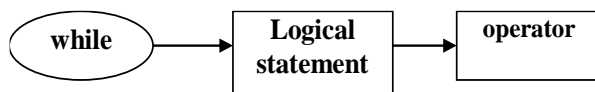
You can use the standard Break and Continue procedures to control the repetition of statements. Break terminates the loop statement, and Continue continues with the next iteration of that statement.

C++Builder 6 Loop Statements

Cycle statements are used to execute the statements (the so-called loop body statements) that are part of them several times (in a particular case, once or not at all). In C++Builder 6, as mentioned above, there are three kinds of loop statements: while, do while and for.

2.1. Loop statement with while precondition.

Syntax:



while A T;

Where A - boolean expression; T - instruction.

The value of the expression A is evaluated before each execution of the statement T, which is why the While loop is also called a precondition loop.

If the value of A is TRUE(true), then the statement T is executed and control is transferred to evaluate the value of the expression A; if the value of the expression A is FALSE (false), then the statement T is not executed and the loop exits.

Notes: (1) if the initial value of the expression A is FALSE, then the statement T will never be executed; (2) you can use a compound statement in the while loop statement; (3) to avoid endless repetition (looping), it is necessary to change at least one variable included in the condition in the body of the loop operator. Moreover, these changes must be such that the boolean expression evaluates to FALSE sooner or later. If the Boolean expression is initially true and never becomes false under any circumstances, then the loop statement will never terminate.

Finally, the while loop is used, as a rule, in cases where the number of repetitions of the loop is not known in advance. In this regard, one should remember a simple but very important rule - the "innermost" cyclic construction should be formulated with special care in order to minimize the computational costs and increase the efficiency of the program as much as possible.

```
while i > 0
{
```

```

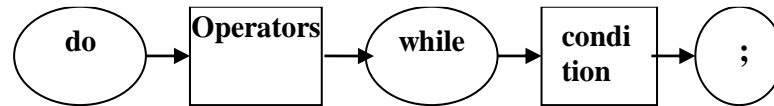
if (i% 2==0) z :=z *x;
i := i /2;
x= x*x); }

```

2.2. Loop statement with postcondition do while

The do while loop (a loop with a postcondition) is usually used in cases where the number of repetitions of the loop body statements is not known in advance.

The syntax for the Repeat loop operator is:



```

do t while a;

```

where t is an operator (possibly composite);
a is a Boolean expression.

The operator "works" as follows: the operators t are executed, the value of the expression a is calculated; if its value is FALSE, then the t statements are executed again, if the value of the expression a is TRUE, then the loop ends. If the value of the expression a is TRUE from the very beginning, then the t statements are executed only once. If the expression a never evaluates to TRUE, then the group of statements t is executed an infinite number of times, then a "loop" occurs.

Note that the lower bound of the cyclic part statements is clearly marked with the word until, so there is no need to enclose the cyclic part statements in operator brackets {}. For example:

```

a)
do
{
    k := i % j;
    i = j;
    j = k;
}
while( j= =0);

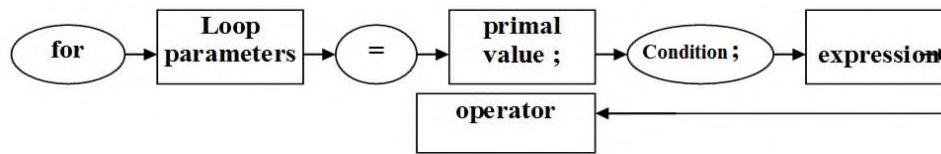
b)
do
{
    cin>>"Enter value (0..9):";
    cin>>i;
}

while (i >= 0) && (i <= 9);

```

2.3. Loop statement with for parameter

The for loop operator is used to organize a loop with a parameter and is used in cases where it is known in advance how many times the cyclic part of the program should be repeated. Operator syntax:



For by increasing values of parameter i:

```
for (i= n1; i<n2 ; i++) t ;
```

n1 is the initial value of the cycle parameter, and n2 is the final value; 3) t is an operator (possibly composite).

The algorithm for executing the loop statement is as follows:

The loop variable is initialized.

The value of the condition is determined. if true, then the loop body is executed, otherwise the statement will not be executed.

An expression is executed to change the loop parameter. (it is incremented by 1).

The variables i, n1, n2 must be of the same type, but not of type float, and the value of n2 must be greater than the value of n1. The variable i takes consecutive values of this type from n1 to n2. In the special case when both n1,n2 are integers and i is a variable of type int, then the step is always equal to one.

For example, if for (i=1 ;i< 20; i++) a=a+1, then for i=1,2,3,...,20 the statement a:=a+1 will be executed.

If n1 and n2 are of a character type and have values, for example, 'A' and 'Z' respectively, then the variable i takes on consecutive values in alphabetical order: 'A','B','C',...,' Z'.

Note that it is possible to organize a cycle by decreasing values of the parameter i. The reserved word downto is used for this.

The syntax of the For operator by decreasing values of the i parameter is:

For (i:=n2; n2== N1; i--) T; where: 1) i - variable (parameter) of the cycle; 2) n2 is the initial value of the cycle parameter, and N1 is the final value; 3) T is an operator (possibly composite).

In this case, the parameter i takes successive decreasing values of this type from n2 to n1 (the step is -1. For example, if

For (i=20; i==1; i--) a=a+1, then for i=20,19,18,...,1 the operator A:=A+1 will be executed.

Let's give an example of a loop operator with a parameter.

```
a) for (i := 2; i< 63 ;i++)
```

```
    if ((i %2 ) != 0) nch = nch+1;
```

```

b) for(i= 1 ;i< 10;i++)
    for (j = 1; j< 10;j++)
    {
        X = 0;
        for (k= 1; k< 10;k++)
            x = x + k*j+i;
        cout<<x;
    }

```

In the loop statements, the break and continue instructions can be used, which are used to terminate the loop, and in the first case, the loop block is exited, and in the second, the current iteration of the loop is completed.

Here are some examples of using loop statements:

Example 1

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{50}.$$

Calculate sum

a) Using a while loop

```

#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
float s;
int n;
n=50; i=1;
while (i<n){
s=s+1/i; i=i+1;
}
cout<<s;
getch();
return 0;
}

```

b) Using a for loop

```

{
float s;

```

```

int i,n;
n=50; s=0;
for ( i=1; i<n; i++)
s=s+1/i;
cout<<s;
getch();
    return 0;
}

```

c) Using the do while loop statement

```

{
float s;
int i,n;
n=50; i=1;
do
{
s=s+1/i; i=i+1;
}
while (i<n);
cout<<s;
getch();
    return 0;
}

```

Example 2

Finding the smallest positive number Eps such that $1+Eps >$

```

//-----
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
    const c1 = 1.0;
        const c2 = 2.0;
float eps,eps1;
    eps=c1;eps1:=c2;

```



```

while (eps1>1.0)
    { eps:=eps/2;eps1:=eps+1 }
eps=eps*2;
cout<<eps;
getch();
return 0;
}

```

Example 3

Find all prime numbers on a given interval m,n (use a loop with a parameter)
(program code in console mode)

```

//-----
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
    int m,n; // segment boundaries
    int i,j; // Cycle Options
    int kl ;
    cout<<" Enter Segment Boundary..." ;
    cin>>m,n;
        For( j=2 ;to int(Sqrt(i)) do
    {
    If (i % j)=0 then kl:=kl+1;
        If kl=0 then cout<<i," ")
    }
    Cout<<"Result: ",d) ;
}

```

Example 3

Implement the task of calculating a function in the interval x a,b with a step h. (in visual mode)

This program uses Memo ,Lable, Edit, Button components
The Edit number depends on how much data the user needs to enter manually.
Button1 - represents a button and is used to evaluate the function.

Memo1 is used to display all F values in the range [A,B] since component represents multiline text.

Program code:

```
-----  
#include "Unit1.h"  
#include <vcl.h>  
#pragma hdrstop  
  
//-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
  
//-----  
__fastcall TForm1::TForm1(TComponent* Owner)  
    : TForm(Owner)  
{  
}  
//-----  
void __fastcall TForm1::Button1Click(TObject *Sender)  
{  
    {  
float a,b,h,f,x;  
    a=strtof(Edit2->Text);  
    b=strtof(Edit3->Text);  
    h=strtof(Edit1->Text);  
    do  
    {  
        x=a;  
        f=x*x +sin(x) +exp(x);  
        Memo1->Text= Memo1->Text + “ “+floattostr(f) +” ”;  
        a=a+h;  
    }  
    while(a>b);  
    }  
}
```

Conclusions:

1. Three types of loop operators are used to describe cyclic algorithms in C++Builder 6.
2. Loop operators with postcondition and precondition are used for loops with an unknown number of repetitions. Moreover, the loop operator with a postcondition is executed at least once, while the loop with a precondition may not be executed even once.

3. In loops with a known number of repetitions, it is convenient to use the loop operator with a parameter, but taking into account the type of the loop variable.

4. All loop operators can be used for cyclic nested structure algorithms (loop within a loop when there are several loop variables).

5. Memo and ListBox components are used to display a set of results of the cyclic program.

Lecture 14

Functions and modules. Standard and user-defined functions in programming languages. Local, static, dynamic variables.

Plan:

1. Structured types of C++ Builder 6.
1. Arrays and their declaration in the program. Multidimensional arrays.
2. Processing array elements in the program.

Dynamic arrays

1. In C++Builder 6, new types can be introduced. These include an enumerated type. An enumerated type in C++Builder 6 is a collection of constants listed with their values in parentheses, separated by commas.

The new type input rule is defined:

```
enum<type name>=(constant 1, constant 2, constant3,... constant n).
```

Enumerated type constants are strictly ordered and the type of these values can be a scalar standard type, except for type float. (The real type has an infinite number of values, even in the range 0.1.)

Enumerated type examples:

```
enum week=(sun,mon,tue,wed,thi,fri,sat);
```

```
enum color=(red,blue,white,yellow,black);
```

```
enum subject=(math,phis,biol,eng);
```

```
enum season=(spring,summer,autmn,winter);
```

Declared types can be used to declare the following variables:

```
enum weekday;
```

```
enum color cvet;
```

```
enum subject lessson;
```

```
enum season yeartime season;
```

In this case, you can write the following assignment operators:

```
day=wed; cvet=red; lesson=eng; yeartime=winter;
```

```
cvet=mon; day=math; operators are incorrect, because do not match the description.
```

Values of an enumerated type cannot be entered and output by input and output statements.

To input and output these values, you must unambiguously map them to the corresponding possible values of the standard types (for example, string constants)

Input example:

Enumerated type examples:

```
enum week=(sun,mon,tue,wed,thi,fri,sat);
enum color=(red,blue,white,yellow,black);
enum subject=(math,phis,biol,eng);
enum season=(spring,summer,autmn,winter);
```

Declared types can be used to declare the following variables:

```
enum weekday;
enum color cvet;
enum subject lessson;
enum season yeartime season;
```

In this case, you can write the following assignment operators:

```
day=wed; cvet=red; lesson=eng; yeartime=winter;
```

```
switch( day)
{
case mon: writeln('Monday');break;
case tue: writeln('Tuesday'); break
case wed: writeln('Wednesday'); break
case thu: writeln('Thursday'); break
case fri: writeln('Friday'); break
case sat: writeln('Saturday'); break
case sun: writeln('Sunday'); }
```

It should be noted that the ordered values of an enumerated type have their ordinal number from 0 to 255. Therefore, the number of constants in an enumerated type is limited.

2. It is known that when implementing a wide class of tasks in programming languages, such a construction as arrays is used. For example, the coefficients of systems of linear equations. Moreover, the result is also an array of system roots.

For this, special structured array types are used. An array is a collection of ordered variables of the same type, provided with the same name. An array component is an indexed variable whose index indicates its number among the elements of the given array. For an array X, the elements are x_1, x_2, \dots, x_{10} . There are one-dimensional, two-dimensional, three-dimensional, etc. n-dimensional arrays.

For example:

A(10) ,D(4,4) - where A,D are the names of arrays, their dimensions are indicated in brackets

This is the mathematical notation for an array. The dimension of an array indicates the number of components or otherwise the number of elements of the given array.

Array A has 10 elements: A1, A2, A3 ... A10;

In array D-16 elements:

D11 D12 D13 D14

D21, . . . D24

. . . .

D41. . . D44

The number of elements in a multidimensional array is determined by the product of the boundaries of the indices, and the dimension is determined by the number of indices. C++Builder 6 uses a special type to declare an array. The type of an array component in C++Builder 6 can be any simple or structured language type, including the regular type. An index type is any bounded or enumerated type. Specific index values can be given by a corresponding bounded type expression, called an index expression. Thus, an element of an array (component) in a program is an indexed variable consisting of the array name and an index expression, which is specified in square brackets.

Array declaration examples:

```
float P [10];
```

```
int t[4][4];
```

The description of an array in a program differs from a simple description of a variable by the presence of square brackets after the array name, which indicate the number of array elements (its dimension):

```
int a[10]; // array of ten integers
```

```
float b[3]; // array of three real numbers.
```

The numbering of array elements always starts from zero. So for the previously declared array of ten integers, the first index will be 0 and the last 9.

Like variables, arrays can be initialized at the time of declaration with a list of constant values enclosed in curly braces.

```
int a[3] = { 1, 2, 3};
```

The dimension of an array, together with the type of its elements, determines the amount of memory required to store the array. This volume is already allocated at the compilation stage, so only positive constants or constant expressions can be used to specify the array dimension. If the dimension is not specified, the compiler tries to allocate memory based on the number of initialized values. Conversely, if the dimension does not match the number of initialized values, the remaining elements of the array are filled with zeros.

Array elements are accessed by specifying the element index in square brackets after the array name.

Example. A program that finds the sum of a given array:

```

#include <iostream.h>
int main(){
int i, sum;
int a [5]    {7, 6, 3, 2, 6 };
for (i = 0, sum = 0; i < 5; i++) sum += a[i];
cout << "Sum of elements: " << sum;
return 0;
}

```

You can set the dimension of an array using a named constant:

```

const int n = 5;
int a[n] = {1, 2, 3, 4, 5};

```

The const specifier indicates that the value declared with it cannot be changed in the program. With each element of the array, you can perform the same operations as with a simple variable of the same type: modify, use in expressions, assign to another variable, etc.

Multidimensional arrays.

Arrays of arrays are called multidimensional arrays. They can be represented as an array, each element of which is an array. For example, a two-dimensional array can be represented as a simple m-by-n-element table. In a program, a two-dimensional array is declared by specifying both of its dimensions:

```

int matrix[3][5];
int array2d[5][5];

```

A two-dimensional array, generally speaking, is an abstraction, that is, the logical arrangement of elements in it is determined by the programmer, and not by the C compiler. You can talk about a two-dimensional array m by n, like a table with m rows and n columns. In memory, such an array is located line by line, so when moving to the next element, the rightmost index grows fastest:

```

int a[2] [3] ;
a[0] [0]
a[0] [1]
a[0] [2]
a[1] [0]
a[1] [1]
a[1] [2]

```

In this example, a 2x3 array is given, and all its elements are listed in a row. As you can see, first the rightmost index (column) runs through all 3 values, then the first index (row) increases by one, and so on. When initializing a multidimensional array, it is represented by either as an array "of arrays, with each array enclosed in its own curly braces (in this case,

you can not specify the dimension), or a general list of elements is given in the ""th order in which they are located in memory, i.e. line by line. Of course, in this case, you will have to specify the dimension, otherwise the compiler will not be able to determine how many elements are in one line and how many lines are in total.

3. Input and output of array elements is performed both directly in the program and using input and output operators or an assignment operator using a cycle operator with a FOR parameter, as well as other cycle operators of the C++ Builder 6 language. The following program fragment can be used to input and output of elements of a one-dimensional array.:

```
int i;
for (i = 0; i < n; i++) cin << b[i];
    { input of array elements }
int i;
for (i = 0; i < n; i++) cin << b[i];
```

Matrix operations are provided in C++Builder 6 for processing arrays: for example, a=b if the matrices are of the same type and size. But you can't perform comparison operations in an if a=b conditional statement. This is done element by element in the program. Consider examples. The following program finds the row with the smallest sum of elements in a two-dimensional array:

```
#include <iostream.h>
int main(){
const int nstr = 3, nstb = 4;
int m[nstr][nstb] =
{ {1, 3, 3, 6}, {1, 1, 2, 2}, {3, 3, 2, 0} };
int i, j;
int sum = 0, min = 0, num;
for (i = 0; i < nstr; i++) {
sum = 0;
for (j = 0; j < nstb; j++)
sum += m[i][j];
if (sum < min) min = sum;
}
cout << "Least sum in line" << num << ": " << sum << "\n";
return 0;
}
```

Sorting arrays. Sorting an array is giving a sequence of array elements a certain order, such as arranging them in ascending or descending order. There are many different sorting methods. Below is the algorithm for sorting an array by selection method. The algorithm is

that the smallest element is selected and swapped with the first element of the array, then the elements are considered, starting with the second, and the smallest of them is swapped with the second element, and so on n-1 times.

```
#include <iostream.h>
int main ()
{
const int n = 20;
int b[n];
int i;
for (i = 0; i < n; i++) cin >> b[i];
for (i = 0; i < n-1; i++)
{
int imin = i;
for (int j = i + 1; j < n; j++)
if (b[j] < b[imin]) imin = j;
int a = b[i];
b [i] := b [imin];
b[imin] = a;
}
for (i = 0; i < n; i++) cout << b[i] << ' ';
return 0;
}
```

As a task, you can comment on this program. The only thing that may seem difficult is the exchange of two variables. It is carried out using a temporary variable a, which stores the previous value of the variable to be changed.

Another way to sort is bubble sort. Its principle is based on the fact that “lighter” elements “float to the beginning of the array, and “heavier” ones sink;

```
#include <iostream.h>
int main()
{
const int arraysize = 10;
int a [arraysize] {2,6,4,8,10,12,89,68, 45, 37};
inttemp;
cout ""Elements in source order \n";
for (int i = 0; i < arraysize; i++)
cout << a[i] << " ";
for (int pass = 1; pass<arraysize; pass++)
```



```

for (i = 0; i < arraysize - 1; i++)
if (a[i] > a[i-t-1]) {
temp = a[i];
a[i] = a[i+1];
a[i+1] = temp;
}
cout << "\n " "Elements ascending" << "\n";
for (i = 0; i < arraysize; i++)
    cout << a[i] << " ";
cout << "\n";
return 0;
}

```

Example 3. Given a square matrix. Calculate the sum of all elements of the array. (Visual mode). The program uses the Lable, Button, StringGrid components (the Cells property determines the index of the component). Button1(button) is used to populate an array (Stringgrid). Button2 - to calculate the sum of all array elements according to the program code.

Program code:

```

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
intsum;
int i,j, a[3][3];
randomize();
for(i=0;i<3;i++)
for(j=0;j<3;j++)

```

```

{ a[i][j]=random(10);
StringGrid1->Cells[i][j]=IntToStr(a[i][j]);
sum=sum+a[i][j];}
}
//-----
void __fastcall TForm1::Button2Click(TObject *Sender)
{
intsum;
    Label1->Caption="Sum of all elements=" +IntToStr(sum);
}
//-----

```

Conclusions:

Enumerated types in C++Builder 6 have their own specification and use.

In C++Builder 6, array construction is a structured type.

To describe multidimensional arrays, you can use your own specification and declaration syntax.

The type of array indexes can be any ordinal type. The C++Builder 6 language has the ability to process dynamic arrays, taking into account the peculiarities of their description and initialization. The input and output of the components of a variable of a regular type, and their processing in the program is carried out only with the help of loop statements. In the C++ language, the structure type is defined to describe such a data type. Unlike an array, all elements of which are of the same type, a structure can contain elements of different types. In C++, a struct is a kind of class and has all of its properties, but in many cases it is sufficient to use structs as they are defined in C:

```

struct [ typename ]
{type_1 element_1;
 type_2 element_2;
 type_n element_n;}
    [ descriptor-list ];

```

The elements of a structure are called fields of the structure and can be of any type, except for the type of the same structure, but can be pointers to it.

If there is no type name, a list of variable descriptors, pointers, or arrays must be provided. In this case, the structure declaration serves as the definition of the list elements:

```

struct
{
charfio[30];

```

```
int date, code;
float salary;
}stuff[100], *ps; /* defining an array of structures and a pointer to the structure */
```

When processing variables of type structure, the C++Builder 6 compiler provides the ability to directly access any field. To do this, you must specify the variable name and the selected field, separating them with a dot.

For example:

for the student variable, you can apply the following statements:
 stud.fio="Khamidova Zamira"; stud.date=1984; stud.pol:="F";

For the input-output of values in the program, the input and output operators of the C++ Builder 6 language are used. (if standard types) otherwise the values are input by assignment operators.

When processing fields of variables of type structure, all possible operations and standard functions provided by the syntax of the language for the type of the selected field are used and there were no conflicts with type descriptions.

Example: Each computer is characterized by its name and technical characteristics: operating speed in thousand op/sec, RAM size, machine word length, number of pixels in graphics mode. Sort computers by memory size (in ascending order).

We write the program code in visual mode in the following form:

Visual mode:

```

/-----
#include <vcl.h>
#pragma hdrstop

#include "Unitctruktura.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
typedef struct {
char name[10];
float speed;

```

```

int razr, pix, ram; } comp;
comp evm[10]; comp k; int i,j,n=3;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
    for(i=0;i<n;i++)
    { evm[i].name=Edit1->Text;
      evm[i].speed=StrToFloat(Edit2->Text);
      evm[i].razr=StrToInt(Edit3->Text);
      evm[i].pix=StrToInt(Edit4->Text);
      evm[i].ram=StrToInt(Edit5->Text);
    }
    Edit1->Text=" ";
    Edit2->Text=" ";
    Edit3->Text=" ";
    Edit4->Text=" ";
    Edit5->Text=" ";
}
//-----

void __fastcall TForm1::Button2Click(TObject *Sender)
{
    for (i=1;i<=n-1;i++) {
    for (j=1;j<=n-1;j++) {

    if (evm[j].ram>evm[j+1].ram)
    { k=evm[j]; evm[j]=evm[j+1]; evm[j+1]=k; } }

    for (i=1;i<=n;i++)
    Memo1->Lines->Add(IntToStr(i)+". "+evm[i].name+" / "+
      IntToStr(evm[i].ram)+" /"+FloatToStr(evm[i].speed)+
      " /"+IntToStr(evm[i].razr)+" /"+IntToStr(evm[i].pix));

    }
}
//-----

```

Multiple data type.

1. The concept of a set in C++Builder 6 has the same meaning as in mathematics. It is a set of all possible unordered elements. For example, a lot of integers, a lot of shapes, a lot of details, a lot of functions. This type in the C++ language can be called a union.(union).

Sets in C++Builder 6 are a set of different elements of the base type (int, char, enum).

2. The type declaration is carried out like all structured types in a C++Builder 6 program.

This type in the C++Builder 6 language allows, based on some base type, to introduce sets of types consisting of elements of this base type. A set is made up of individual elements, and each element of a set is an expression of a base type.

```
<Base set type>::=<restricted integer>!<restricted char>!<char>! <boolean>!  
<enumerable>!<restricted enumerable>
```

```
Set<type,minval,maxval>
```

Real types cannot be used. they are unlimited.

For example, if set descriptions are given as follows:

```
enum color(red,black,blue,white,yellow);
```

```
set <red, black, blue, white>s
```

```
set<int,1,20>s1,s2; { base type bounded integer }
```

```
set <char, "A","R">p; {base type restricted character }
```

then set type variables can take the following values:

```
s1=(2,4,7,9,13,19); s2=(1,5,10,15,20);
```

```
p=("D","F","G","K");
```

```
s=(red, blue, white);
```

Each element of the set in the computer is assigned a binary number. The maximum value of the number is limited and is 256. If the element of the set is an integer, then it is represented in the computer by its value in the binary number system. If it is a character or a letter, then its value is determined by its ordinal number in the ASCII I code. In an enumerated type, their ordinal numbers are written as values of the elements of the set.

Test questions:

1. How and in what files can you create your own class, dynamic object and call class methods?

2.What are component properties? Give examples of several form properties.

3.What are component events? Give examples.

4.How can I change the values of component properties?

Lecture 15

Application in systems of graphic and multimedia programming. Capabilities of the graphic module and their use. Object animation, animation options

Plan:

1.GDI drawing tools

2.Canvas. Using canvas.

3.Graphic files

4.Maintenance of palettes

Key terms: *Canvas, graphics, colors.*

The Windows system provides GDI (Graphics Device Interface) drawing tools for drawing graphics on a graphics context, regardless of the type of output device. When directly calling GDI functions, they need to pass a device context handle (HDC) that specifies the drawing tools. The graphics context represents the graphics device model. After you've finished working with images, you need to restore the device context to its original state and release it. C++Builder takes care of the GDI work of looking up image descriptors and memory resources. Applications can directly access Windows GDI functions. Consider an example:

```
void __fastcall TForm1::Button1Click(TObject *Sender) { HDC hdc =Canvas->Handle;
LineTo(hdc,100,100);
MessageBox(Form1->Handle,"Windows API call","",MB_OK);
}
```

C++Builder provides a simple interface through the Canvas property of graphical components. This property initializes and releases the device context. The Canvas property is a class that encapsulates properties and methods for working with graphics. In the C++Builder environment, there are three ways to work with graphics:

- Canva provides a bitmap of a surface for drawing on a form, graphical component, or other bitmap. In addition to the form, objects of the TImage and TPaintBox classes are used for drawing. Canvas is a property of these classes. Consider an example of drawing on a form:

```
void __fastcall TForm1::FormPaint(TObject *Sender) { Canvas->Pen->Color = clBlue;
// select color for outline
Canvas->Brush->Color = clYellow; // select fill color
Canvas->Ellipse(10, 20, 50, 50); // draw an ellipse
}
```

The FormPaint method here is called by the form's OnPaint event when the form is being drawn. Note that when drawing directly on the form, there are a number of problems associated with its redrawing. The following example discusses various ways to set the shape color and line color for drawing on a TImage component.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
65
if (ColorDialog1->Execute())
```

```

    { Form1->Color = ColorDialog1->Color;
    }
}
void __fastcall TForm1::Button2Click(TObject *Sender) { Image1->Canvas->Pen-
>Width=2;
    randomize();
    Image1->Canvas->Pen->Color =(Graphics::TColor)random(256) ; Image1->Canvas-
>LineTo(100,200);
}

```

When you click on the first button, the color of the form is selected, when you click on the second button, lines are drawn in a randomly selected color.

- **Graphics** (TGraphic) is an abstract base class for working with graphics. C++Builder defines graphic classes TBitmap, TClipboard, TImage, TIcon, and TMetafile derived from the base class Tgraphic. Objects of these classes use canvas drawing methods and have their own methods. Widely used class methods TGraphic LoadFromClipboardFormat(), LoadFromFile(), LoadFromStream(), SaveToClipboardFormat(), SaveToFile(), SaveToStream().

- **Figure** (TPicture) is a container for graphic objects and can contain any graphic objects. The properties of this class are: Bitmap, Graphic, Icon, Metafile, PictureAdapter.

A container TPicture class can contain a bitmap, icon, metafile, or some other graphic type, and the application will refer to all container objects through the TPicture class object. If you need to specify access to a particular graphic object, set it in the Graphic property of this drawing. Methods of the classes of this class are: LoadFromClipboardFormat(), LoadFromFile(), LoadFromStream(), SaveToClipboardFormat(), SaveToFile(), SaveToStream().

MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY



PRACTICAL LESSONS

on subject

“Information technologies in technical systems”

Tashkent 2022

Practical lesson - №1

Learning and applying the interface of CAD applications. A technique for creating mathematical models of engineering problems using applications (Mathematica, Maple, Matlab, MathCAD). Requirements for hardware and software when using CAD systems.

Purpose: To study modern platforms of computer systems and methods of using hardware and software. To get acquainted with the concept of architecture, with the hardware of a personal computer, the principles of von Neumann, the logical nodes of a computer.

Theoretical part

The process of developing software systems is closely related to the field of project management, because any software product is a unique result. The main characteristics of the implementation of a software project directly depend on the organization of this process - the deadlines, the planned budget, the quality of the product being produced. to live without such a device as a computer. It should be considered as a combination of **two components:**

- hardware part (hardware);
- software part (software, soft).

Computer architecture **Computer architecture** is its device and the principles of interaction of its main elements - logical nodes, among which the bases are processors, internal memory (basic and operational), external memory and devices for input-output of information (peripheral) (Fig. 1).

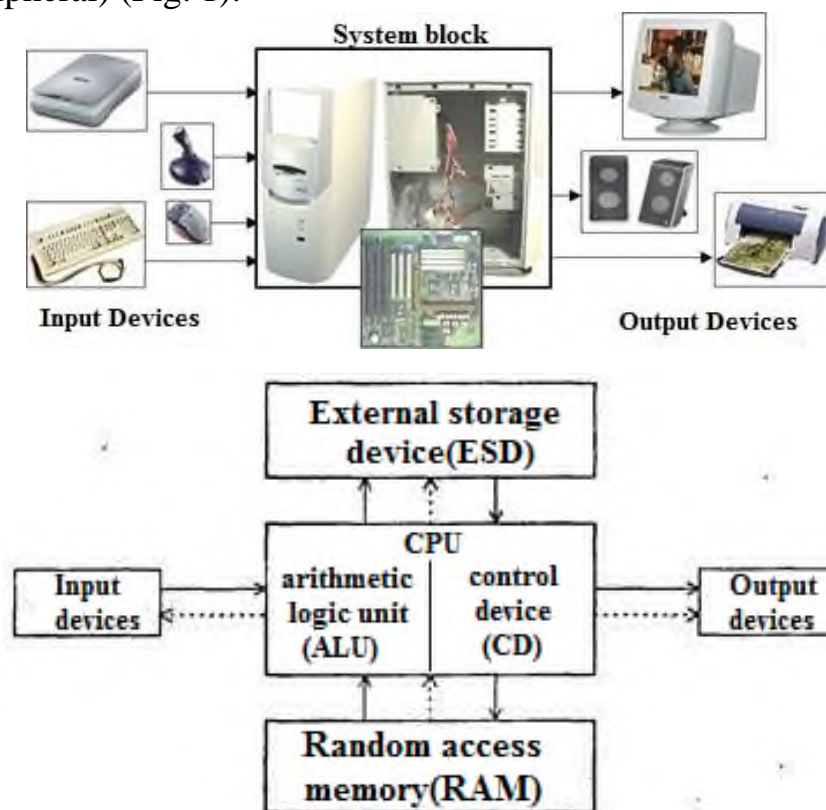


Fig. 1. Conditional model of the structure of the computer architecture

Von Neumann's principles

The principles underlying the architecture of computers were formed in 1945 by John von Neumann, who developed the ideas of Charles Babbage, presenting the computer work as a work with a set of devices: processing, control, memory, input-output.

1. The principle of the same kind of memory. You can perform the same actions on commands as you can on data.

2. The principle of memory addressability. The main memory is structurally composed of pre-numbered cells; process at a random moment of time access to any cell. From here it follows that it is possible to give names to memory areas, so that the values stored in them can be it would be possible to subsequently handle or change them in the process of executing the program using their own names.

3. The principle of subsequent program management. It assumes that the program consists of a set of commands that are executed by the process run automatically one after another in a certain sequence.

4. The principle of rigidity of architecture. Immutability in the process of work topology, architecture, command list.

Harvard architecture

Computers built on the principles of von Neumann have a classic architecture, but besides it, there are other types of architecture. For example, Harvard. Its distinguishing features are:

- instruction storage and data storage are different physical devices;
- the instruction channel and the data channel are also physically separate.

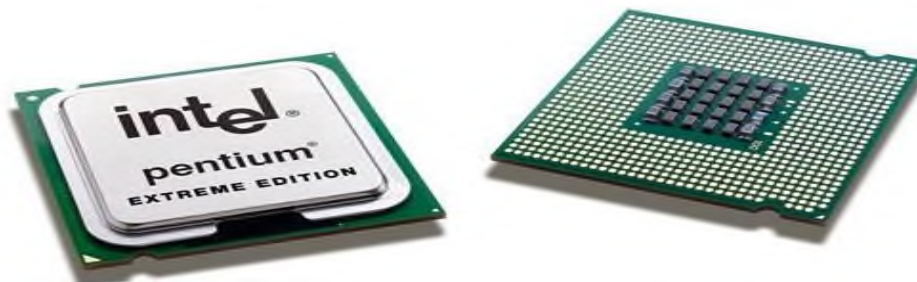
Among computers that are not classical, not the background of the Neumann architecture, you can name it like this my neurocomputers. In them, the work of the cells of the human brain, neurons, as well as some from parts of the nervous system capable of exchanging signals.

Functions of some computer components

Each logical node of the computer performs its functions.

Functions of the processor (Fig. 2):

- data processing (performing arithmetic and logical operations on them);
- management of all other devices of the computer.



View from the side of the radiator mounting View from the side of the contacts

Fig. 2. Computer CPU

The program consists of separate commands. The command includes the code of the operation, the address of the operands (values that participate in the operation) and result address. The execution of the command is divided into the following stages:

- selection of teams;
- formation of the address of the next command;
- de-co-di-ro-va-tion of the team;
- calculation of addresses of operators;
- selection of operas;
- execution of operations;
- the formation of the sign of the result;
- recording the result.

Not all of the stages are present when executing any command (depending on the type of command), however, the stages are high parsing, decoding, forming the address of the next command and executing the operation always take place . In certain situations, two more stages are possible:

- indirect addressing;
- reaction to pre-rying.

Operational memory (Fig. 3) is arranged in the following way:

- receiving information from other devices;
- memorization of information;
- transferring information on request to other computer devices.



Fig. 3. RAM (Random Access Device) computer

Trunk-modular principle

The architecture of modern computers is based on the main-modular principle (Fig. 4). The modular principle allows you to complete the necessary configuration and carry out the necessary modernization. It is based on the bus principle of information exchange between modules. The system bus or computer backbone includes several buses of various purposes. The highway includes three multi-bit tires:

- *data bus*;
- *address bus*;
- *control bus*.

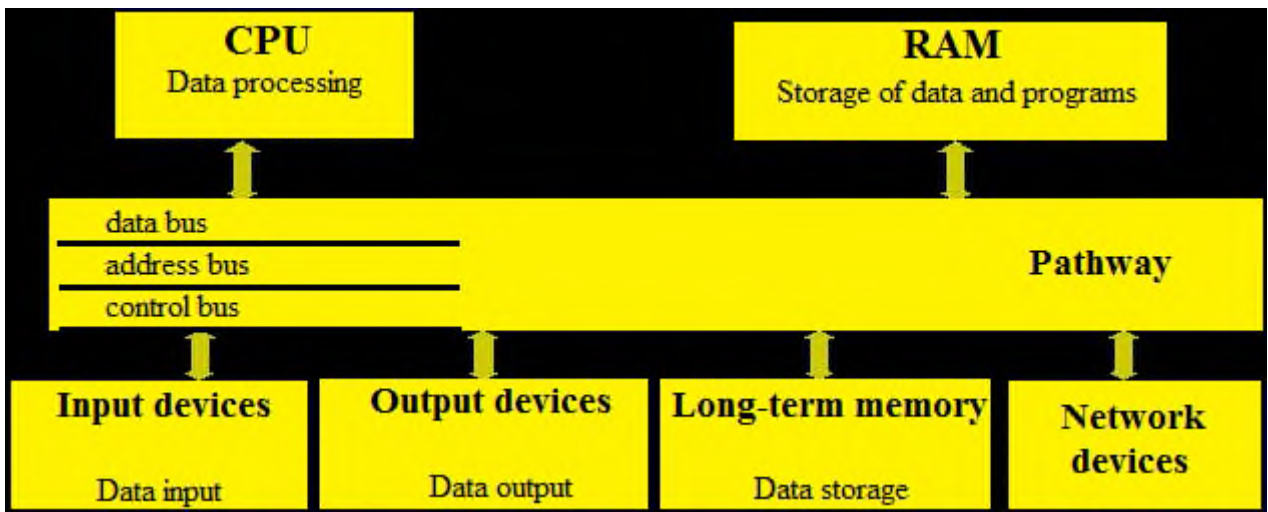


Fig. 4. Pathway-modular principle of building a PC

The data bus is used to transfer various data between computer devices; the address bus is used for addressing the fetched data, that is, for determining their one hundred positions in memory or in input / output devices; the control bus includes control signals that serve for temporal agreement operation of various computer devices, to determine the direction of data transmission, to determine the formats of the transmitted data, etc.

This principle is valid for various computers, which can be conditionally divided into three groups:

- *stationery;*
- *compact (notebooks, netbooks, etc.);*
- *pockets (smartphones, etc.).*

In the system unit of a stationary computer or in the case of a compact computer, there are basic logical nodes are a motherboard with a processor, a power supply unit, on an external memory, etc. d.

The start screen is easily recognizable by the text Start in the upper left corner.

Programs that are downloaded from the Windows Store and gets opened from the Start menu are called applications and programs that are downloaded from any other and get opened from the desktop menu as programs.

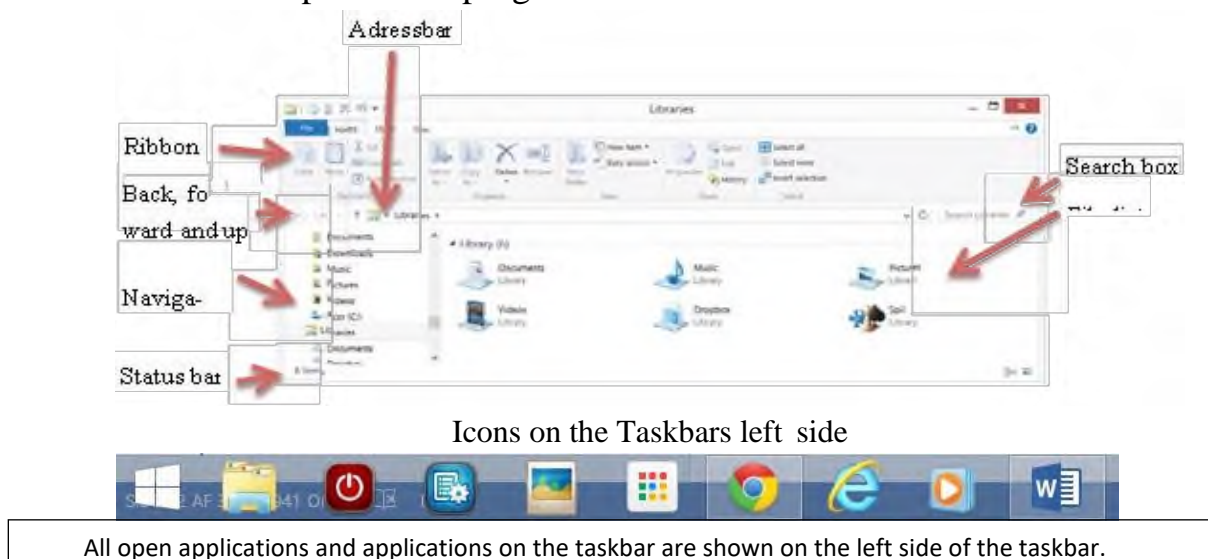


Fig.5.

Start screen where you have quick access to preferred applications, favorite websites, frequently used folders, current work files.

Example. Navigation between user interfaces in Windows 8.1 can be done using key combinations instead of, or, the mouse:

- On the Start screen, you can go to the desktop by clicking on the desktop tile
- From the beginning of the screen, you can go to the desktop by pressing the Windows keys on your keyboard.
- From the beginning of the screen, you can go to the desktop by holding the Windows key, press the D-key once, and release the Windows key again.
- From the beginning of the screen you can go to the desktop by placing the mouse cursor in the lower left corner of the screen until the purple window right-click on the window once.

Click on Desktop.

- From the beginning of the screen you can go to the desktop by placing the mouse pointer in the upper left corner of the screen until a thumbnail image is shown. Click on the image once.

Task 1. Creating a file archive

1. Select any bmp file from the files in the Личная\Графика folder on your disk.
2. Right-click to open the context menu.
3. Select a command (fig. 6).

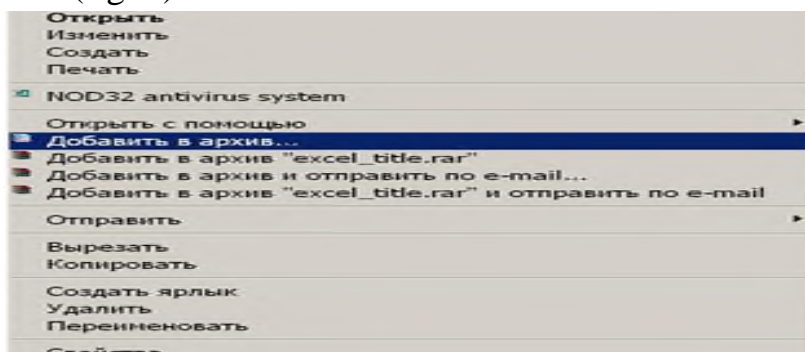


Fig. 6. Context menu of the file object

4. Set the following archiving parameters in the Параметры window (Fig. 7):

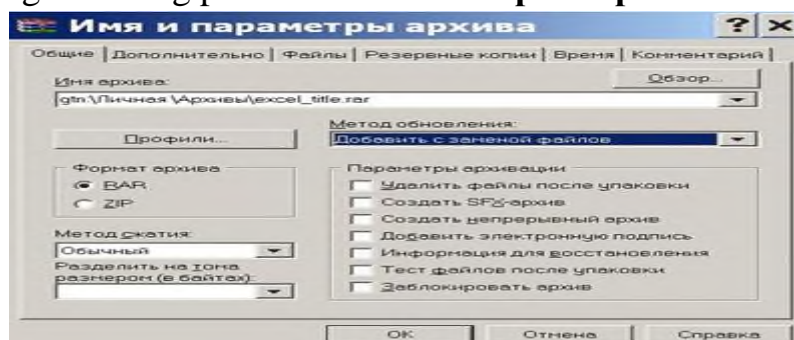


Fig.7. Window for setting archive parameters

- archive name – leave the **default** name (same as the source file);
- archive type - **RAR**.

5. Set the parameters for placing the archive file - **the Archives folder**. To do this, use the browse button to open the folder tree access window and select the **Archives** folder in the **Personal** folder. The path to this folder (address) will be displayed in the **Archive name** line.

6. Click **OK**.

7. Check if the file appears in the Archives folder.

Task 2. Opening an archive file

1. Open the archive file created in the previous task by double-clicking.

2. In the archiver window (Fig. 8) find the information:

- file size before archiving – Size column in the workspace of the archiver window;
- file size after compression – Compressed column;
- date and time of change;
- file type.

Estimate the difference in volume before compression and after compression.

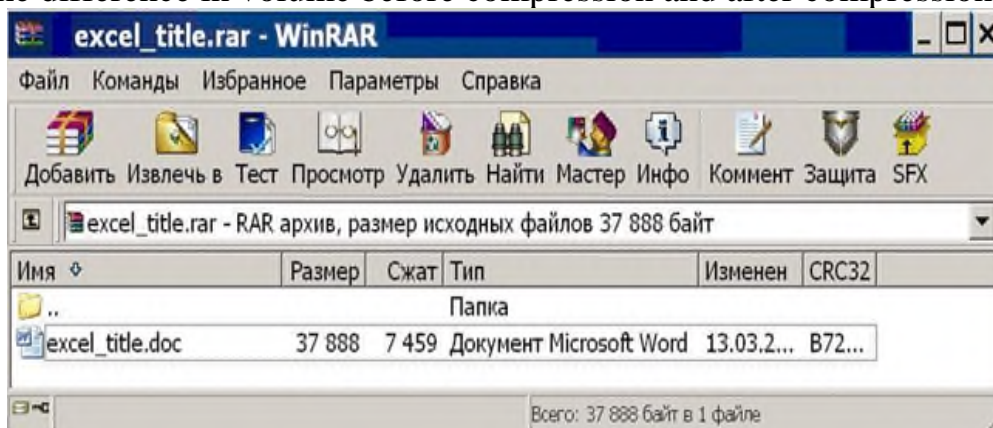


Fig.8. View file options

Task 3. Creating an archive from several files

1. Select a group of files containing Word documents (Personal\Documents folder) and perform archiving according to the algorithm specified in Task 1.

2. Estimate the volume of files before and after archiving.

Task 4. Creating a self-extracting archive

1. Select the same file for archiving as in the first task.

2. Create a self-extracting archive of this file (Fig. 8) according to the task algorithm 1. When performing steps 4 and 5, specify the name of the archive - **SFX**, place it in the Archives folder, check the **SFX** checkbox in the parameters window.

3. Find the file in the Archives folder and familiarize yourself with its properties (Properties context menu command). Determine the file format (extension). Compare the size of the file with the size of the **RAR** archive of the same file.

Task 5. Extracting files from the archive

1. Create an Extract folder on your desktop.

2. Select any of the archives available in the Personal\Archives folder from the folder.

3. Open it by double-clicking in the archiver window (Fig. 9).

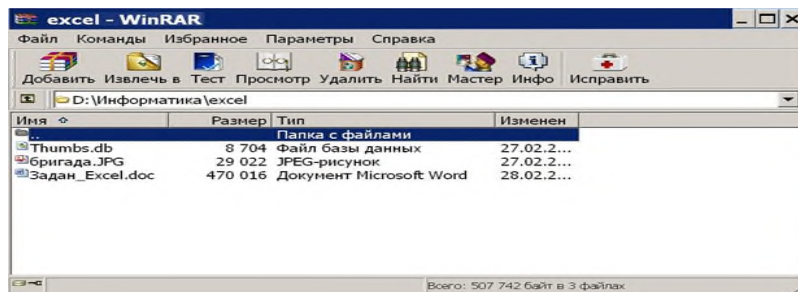


Fig.9. Unpacking the archive

4. Extract it to your desktop into the **Extract folder**. To do this, click the **Extract button** (Fig. 9) to open the window for setting the parameters for extracting from archive, in which specify the required folder in the folder tree (Fig. 10).

5. Click OK.

6. Open the Extract folder. Check out the **Extract** result.

7. Delete the folder from the Desktop.

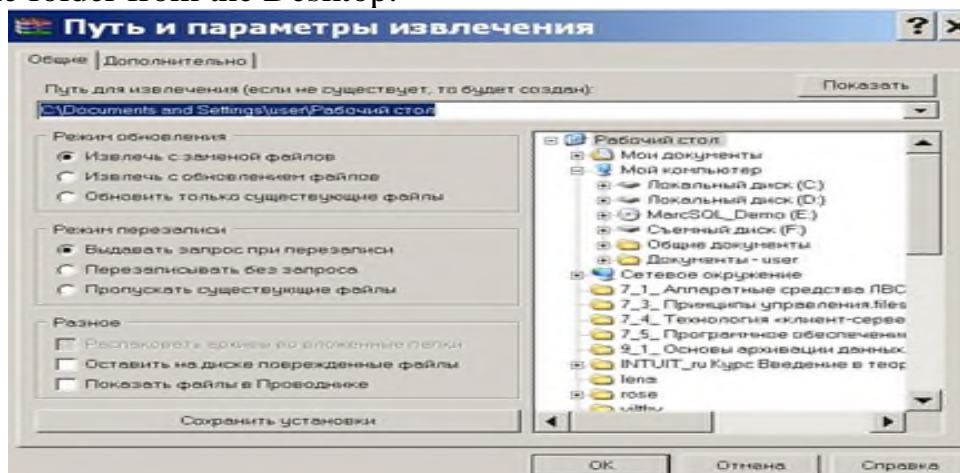


Fig.10. Setting archive extraction options

1. Set your own search criteria for the following files:

- All files on the d: drive created within the **last 2 months**, with the extension Doc. Select those of the found files that are less than 100 kb in size and save them in the Documents folder of your Personal Folder.

- Find all files in the **My Documents** folder that are of type **XLS**.

- Find all files that have Word in their names that are executable files (search scope drive C:). Present the result to the teacher.

- Look for files with .exe extension on drive C: as well as those with .gif extension created in **the last 3 days**. Copy the latest files to your **Graphics** folder.

Control questions

1. What are the main computer devices that make up a computer system?
2. List modern platforms of computer systems.
3. What are the main types and functions of processors do you know?
4. List the main types of computer system memory.
5. What file system is used in the current version of Windows?
6. What are the advantages of NTFS compared to the FAT32 file system?
7. What is the difference between SSD drives and traditional hard drives?
8. What file systems does Windows support for CD-ROMs and DVDs?
9. Specify some network types of APIs in the Windows operating system.

Practical lesson - №2

Working with vectors and matrices in MathCAD. Solution of systems of equations.

Purpose of work: Learn to perform various operations with vectors and matrices; solve systems of linear algebraic equations; learn to build tables of function values of one and two arguments.

Theoretical part

To work with matrices and vectors in Mathcad, use the Matrix panel (Fig. 1).



Fig.1 - Matrix toolbar

You can open the Matrix panel by clicking on the image of the matrix on the Math toolbar (Fig. 2):



Fig 2 - Math toolbar

In order to type a matrix of the required dimension, you need to click on the image of the matrix on the Matrix panel, and a dialog box will open (Fig. 3):

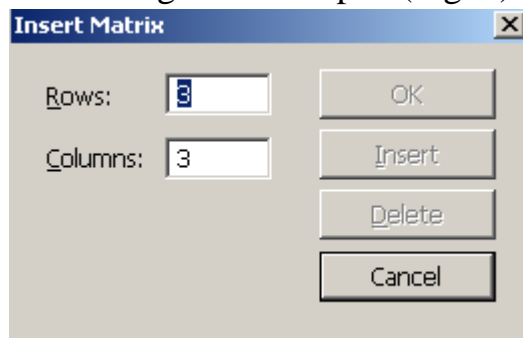


Fig.3 - Dialog box Insert Matrix

In the dialog box, specify the number of matrix rows (Rows) and the number of columns (Cols) of the matrix, then click OK. Next, in the field that opens, type the required numbers.

Basic operations on matrices and vectors

Let the matrix

$$M := \begin{bmatrix} 7 & 6 & 5 & 8 \\ 1 & 0 & 3 & 2 \\ 9 & 3 & 1 & 0 \\ 5 & 4 & 3 & 2 \end{bmatrix}$$

Inverse matrix M^{-1} , transpose matrix M^T , determinant $|M|$ are calculated using the appropriate tools on the Matrix panel.

$$M^{-1} = \begin{bmatrix} 0.026 & 0.071 & 0.179 & -0.173 \\ -0.026 & -0.321 & -0.179 & 0.423 \\ -0.154 & 0.327 & -0.077 & 0.288 \\ 0.218 & -0.026 & 0.026 & -0.346 \end{bmatrix}$$

$$M^T = \begin{bmatrix} 7 & 1 & 9 & 5 \\ 6 & 0 & 3 & 4 \\ 5 & 3 & 1 & 3 \\ 8 & 2 & 0 & 2 \end{bmatrix}$$

$$|M| = -312$$

Matrix rank

$$\text{rank}(M) = 4$$

The matrix size can be calculated as follows:

$$\text{Number of rows: rows}(M) = 4$$

$$\text{Number of columns: cols}(M) = 4$$

Matrix Merging

Let be

$$B := \begin{bmatrix} 6 & 8 & 1 \\ 7 & 6 & 4 \end{bmatrix} \quad C := \begin{bmatrix} 4 & 7 & 9 \\ 3 & 0 & 2 \end{bmatrix} \quad A := \begin{bmatrix} 1 & 4 & 5 \\ 5 & 0 & 3 \\ 5 & 7 & 8 \end{bmatrix}$$

Then the merging of matrices from left to right is performed as follows:

$$\text{augment}(C, B) = \begin{bmatrix} 4 & 7 & 9 & 6 & 8 & 1 \\ 3 & 0 & 2 & 7 & 6 & 4 \end{bmatrix}$$

Merging matrices from top to bottom:

$$\text{stack}(A, B) = \begin{bmatrix} 1 & 4 & 5 \\ 5 & 0 & 3 \\ 5 & 7 & 8 \\ 6 & 8 & 1 \\ 7 & 6 & 4 \end{bmatrix}$$

Selection of individual elements, rows, columns of a matrix

When performing operations, keep in mind that rows and columns are numbered starting from zero.

$$A_{1,1} = 0 \quad A_{0,1} = 4 \quad A^{(0)} = \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix} \quad (A^T)^{(0)} = \begin{bmatrix} 1 \\ 4 \\ 5 \end{bmatrix}$$

To select a submatrix, the `submatrix(A, ir, jr, ic, jc)` function is used, which returns the part of the matrix A located between rows ir, jr and columns ic, jc inclusive.

$$\text{submatrix}(A, 0, 1, 0, 1) = \begin{bmatrix} 1 & 4 \\ 5 & 0 \end{bmatrix}$$

Operations on vectors

The vector modulus is calculated using the `|x|` tool located on the Calculator panel (not to be confused with the similar designation on the Matrix panel, intended for calculating the matrix determinant).

Example.

$$r := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad r1 := \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}$$

$$|r| = 3.74165739$$

Vector dimension:

$$\text{length}(r) = 3$$

Dot and vector product

$$r \times r1 = \begin{bmatrix} 0 \\ 9 \\ -6 \end{bmatrix} \quad r \cdot r1 = 17$$

Creating a Table of Values for a Single Variable Function

In order to build a table of function values in Mathcad, you must do the following:

- set a function;
- set the boundaries of the interval [a, b] on which the function values will be calculated;

- set the number of points for splitting the interval [a, b];
- calculate the vector of function argument values at split points;
- calculate the function value vector corresponding to the argument value vector.

Example.

Function assignment: $f(x) := e^{-x^2}$.

Setting the range limits: a:= -10; b:= 10.

Set the number of split points, including boundary points (≥ 2): n:= 20.

Computing the vector of argument values:

$$r := a, a + \frac{b - a}{n - 1} .. b$$

r =

-10
-8.94736842
-7.89473684
-6.84210526
-5.78947368
-4.73684211
-3.68421053
-2.63157895
-1.57894737
-0.52631579
0.52631579
1.57894737
2.63157895
3.68421053
4.73684211
...

Calculation of the vector of function values:

f(r) =

0
0
0
0
2.77533332·10 ⁻¹⁵
1.80070418·10 ⁻¹⁰
1.27392585·10 ⁻⁶
9.82698935·10 ⁻⁴
0.08265543
0.7580482
0.7580482
0.08265543
9.82698935·10 ⁻⁴
1.27392585·10 ⁻⁶
1.80070418·10 ⁻¹⁰
...

The matrix of values of a function of two variables is constructed in a similar way.

Example.

Function assignment: $f(x,y) := \sin(x,y)$

Setting the boundaries of the rectangle along the x-axis:

$$x_a := -2\pi; x_b := 2\pi.$$

Setting the boundaries of the rectangle along the y-axis:

$$y_a := -\pi; y_b := \pi.$$

Setting the number of split points along the x-axis, including boundary points (≥ 2): $x_n := 40$.

Specifying the number of split points along the y-axis, including boundary points (≥ 2): $y_n := 20$.

Calculation of index values i, j : $i := 0 .. x_n - 1$; $j := 0 .. y_n - 1$.

Calculation of $x(i)$ and $y(j)$ values:

$$x_{i_i} := x_a + i \cdot \frac{x_b - x_a}{x_n - 1};$$

$$y_{i_j} := y_a + j \cdot \frac{y_b - y_a}{y_n - 1}.$$

Function value matrix: $M_{i,j} := f(x_{i_i}, y_{i_j})$

	0	1	2	3	4	5	6	7
0	0.777	-0.928	0.124	0.807	-0.908	0.075	0.836	-0.886
1	-0.122	-0.866	0.798	0.244	-0.988	0.526	0.578	-0.977
2	-0.906	-0.142	0.988	-0.431	-0.739	0.859	0.241	-0.999
3	-0.838	0.692	0.581	-0.908	-0.243	0.998	-0.129	-0.95
4	0.018	0.995	-0.178	-0.966	0.334	0.912	-0.482	-0.835
5	0.857	0.537	-0.83	-0.579	0.8	0.621	-0.768	-0.66
6	0.891	-0.333	-0.979	0.075	0.998	0.189	-0.949	-0.439
7	0.087	-0.947	-0.535	0.694	0.864	-0.285	-0.999	-0.187
8	-0.799	-0.837	0.232	0.994	0.441	-0.695	-0.912	0.078
9	-0.933	-0.085	0.859	0.836	-0.129	-0.949	-0.699	0.338
10	-0.191	0.731	0.966	0.292	-0.656	-0.988	-0.391	0.574
11	0.731	0.988	0.488	-0.386	-0.964	-0.804	-0.028	0.769
12	0.966	0.488	-0.285	-0.886	-0.95	-0.439	0.338	0.91
13	0.292	-0.386	-0.886	-0.978	-0.619	0.025	0.657	0.987
14	-0.656	-0.964	-0.95	-0.619	-0.082	0.483	0.887	0.994
15	-0.988	-0.804	-0.439	0.025	0.483	0.833	0.994	...

Solving systems of linear algebraic equations

Let us present several methods for solving systems of linear algebraic equations (SLAE) in Mathcad. Inverse matrix method (for square systems with a nonsingular matrix).

Let SLAE be given $A \cdot X = B$

Then the solution vector is found by the formula: $X = A^{-1} \cdot B$

Least square method.

Let the system of equations be given: $\hat{A}_{m \times n} \cdot X_{n \times 1} = B_{m \times 1}$, where $m > n$, i.e. the number of equations is greater than the number of unknowns.

We multiply both parts of the matrix equation by the transposed matrix of the system.
 $\hat{A}^T \cdot A \cdot X = A^T B$

From here $X = (A^T \cdot A)^{-1} \cdot A^T \cdot B$

Symbolic solution method using the Given - Find block (the solution of the system will be found if it exists).

Example.

Given

$$2x + 3y + 5z + 4 = 0$$

$$4x + 5y + 7z - 5 = 0$$

$$3x + 8y - 4z - 1 = 0$$

Note: The equal signs \leq between the left and right parts of the equations must be set using the Boolean panel (Fig. 4).

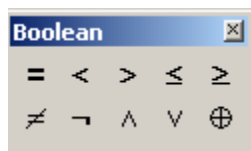


Figure 4 - Boolean toolbar

Any "other" equal signs taken from other panels or typed from the keyboard will result in an error.

The task

Calculate the determinants of matrices, find the matrix inverse to the given one, find the ranks of the matrices, extract the given rows and columns from the matrices.

Solve the SLAE by the inverse matrix method and using the Given - Find block; solve SLAE by the least squares method.

Task variants

Exercise 1.

- calculate determinants;
- find the matrix inverse to the given one, transpose the matrix;
- find the ranks of matrices; select the second row and third column from the matrices.

Variant №	Task 1 a	Task 1 b	Task 1 c
1	$\begin{vmatrix} 1 & 2 & 2 & 4 \\ 1 & 6 & 0 & 5 \\ 1 & 2 & -1 & 3 \\ 1 & 2 & -1 & 2 \end{vmatrix}$ $\begin{vmatrix} -2 & 2 & -2 & 1 \\ -1 & 2 & 2 & 4 \\ -1 & 1 & 0 & 1 \\ -3 & 3 & 0 & 4 \end{vmatrix}$	$\begin{vmatrix} 1 & -1 & 0 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix}$	$\begin{vmatrix} 3 & 0 & 1 \\ 2 & 5 & 3 \\ 1 & 4 & -1 \end{vmatrix}$ $\begin{vmatrix} 2 & 7 & 9 & 7 \\ 6 & 9 & 5 & 4 \\ 8 & 16 & 14 & 11 \\ -4 & -2 & 4 & 3 \end{vmatrix}$
2	$\begin{vmatrix} -2 & 2 & -2 & 1 \\ -1 & 2 & 2 & 4 \\ -1 & 1 & 0 & 1 \\ -3 & 3 & 0 & 4 \end{vmatrix}$ $\begin{vmatrix} 4 & 8 & -3 & 6 \\ 2 & 5 & 0 & 3 \\ 2 & 4 & -3 & 1 \\ 6 & 12 & -9 & 2 \end{vmatrix}$	$\begin{vmatrix} 2 & 3 & 1 \\ 3 & 4 & 2 \\ 1 & 1 & 2 \end{vmatrix}$	$\begin{vmatrix} 4 & 3 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$ $\begin{vmatrix} 2 & 8 & 4 & 0 \\ 8 & 9 & 5 & 1 \\ 10 & 17 & 9 & 1 \\ -6 & -1 & -1 & -1 \end{vmatrix}$

3	$\begin{array}{cccc c} 4 & 8 & -3 & 6 & \\ 2 & 5 & 0 & 3 & \\ 2 & 4 & -3 & 1 & \\ 6 & 12 & -9 & 2 & \\ \hline 6 & -2 & 9 & 7 & \\ 3 & 0 & 8 & -1 & \\ 3 & -1 & 4 & 2 & \\ 9 & -3 & 12 & 7 & \end{array}$	$\begin{array}{ccc c} 4 & 3 & 0 & \\ 1 & 1 & 0 & \\ 0 & 0 & 1 & \end{array}$	$\begin{array}{ccc c} 0 & 0 & 1 & \\ 1 & 5 & 1 & \\ 1 & 4 & 2 & \\ \hline 2 & 0 & 8 & 6 \\ 6 & 9 & 4 & 3 \\ 8 & 9 & 12 & 9 \\ -4 & -9 & 4 & 3 \end{array}$
4	$\begin{array}{cccc c} 6 & -2 & 9 & 7 & \\ 3 & 0 & 8 & -1 & \\ 3 & -1 & 4 & 2 & \\ 9 & -3 & 12 & 7 & \\ \hline 4 & -4 & -1 & 12 & \\ 2 & -3 & 2 & 6 & \\ 2 & -2 & 1 & 4 & \\ 6 & -6 & 3 & 13 & \end{array}$	$\begin{array}{ccc c} 0 & 0 & 1 & \\ 1 & 5 & 1 & \\ 1 & 4 & 2 & \end{array}$	$\begin{array}{ccc c} 1 & 2 & 1 & \\ 0 & 3 & 1 & \\ 0 & 2 & 1 & \\ \hline 2 & 3 & 7 & 4 \\ 4 & 5 & 7 & 2 \\ 6 & 8 & 14 & 6 \\ -2 & -2 & 0 & 2 \end{array}$
5	$\begin{array}{cccc c} 4 & -4 & -1 & 12 & \\ 2 & -3 & 2 & 6 & \\ 2 & -2 & 1 & 4 & \\ 6 & -6 & 3 & 13 & \\ \hline -4 & 2 & 1 & -4 & \\ -2 & 2 & 3 & 3 & \\ -2 & 1 & 0 & -1 & \\ -6 & 3 & 0 & -2 & \end{array}$	$\begin{array}{ccc c} 1 & 2 & 1 & \\ 0 & 3 & 1 & \\ 0 & 2 & 1 & \end{array}$	$\begin{array}{ccc c} 1 & 1 & 2 & \\ 0 & 2 & 3 & \\ 0 & 1 & 2 & \\ \hline 2 & 7 & 4 & 9 \\ 3 & 5 & 2 & 8 \\ 5 & 12 & 6 & 17 \\ -1 & 2 & 2 & 1 \end{array}$
6	$\begin{array}{cccc c} -4 & 2 & 1 & -4 & \\ -2 & 2 & 3 & 3 & \\ -2 & 1 & 0 & -1 & \\ -6 & 3 & 0 & -2 & \\ \hline -6 & 2 & 3 & 10 & \\ -3 & -1 & 1 & 4 & \\ -3 & 1 & 0 & 3 & \\ -9 & 3 & 0 & 8 & \end{array}$	$\begin{array}{ccc c} 1 & 1 & 2 & \\ 0 & 2 & 3 & \\ 0 & 1 & 2 & \end{array}$	$\begin{array}{ccc c} -2 & 0 & 5 & \\ 3 & 1 & 5 & \\ 1 & 0 & 2 & \\ \hline 2 & 5 & 9 & 6 \\ 5 & 7 & 6 & 3 \\ 7 & 12 & 15 & 9 \\ -3 & -2 & -3 & 3 \end{array}$
7	$\begin{array}{cccc c} -6 & 2 & 3 & 10 & \\ -3 & -1 & 1 & 4 & \\ -3 & 1 & 0 & 3 & \\ -9 & 3 & 0 & 8 & \\ \hline 2 & 4 & -5 & 3 & \\ 1 & 4 & -4 & 4 & \\ 1 & 2 & -3 & 1 & \\ 3 & 6 & -9 & 5 & \end{array}$	$\begin{array}{ccc c} 2 & 0 & 5 & \\ 3 & 1 & 5 & \\ 1 & 0 & 2 & \end{array}$	$\begin{array}{ccc c} 1 & 0 & 1 & \\ 2 & 2 & 3 & \\ 3 & 0 & 4 & \\ \hline 2 & 6 & 5 & 3 \\ 8 & 6 & 4 & 3 \\ 10 & 12 & 9 & 6 \\ -6 & 0 & 1 & 0 \end{array}$
8	$\begin{array}{cccc c} 2 & 4 & -5 & 3 & \\ 1 & 4 & -4 & 4 & \\ 1 & 2 & -3 & 1 & \\ 3 & 6 & -9 & 5 & \\ \hline 2 & 4 & 7 & 11 & \\ 1 & -3 & 7 & 12 & \\ 1 & 2 & 4 & 5 & \\ 3 & 6 & 12 & 17 & \end{array}$	$\begin{array}{ccc c} 1 & 0 & 1 & \\ 2 & 2 & 3 & \\ 3 & 0 & 4 & \end{array}$	$\begin{array}{ccc c} -2 & 1 & 0 & \\ 3 & 2 & 0 & \\ 3 & 2 & 1 & \\ \hline 2 & 6 & 9 & 1 \\ 4 & 5 & 3 & 9 \\ 6 & 11 & 12 & 10 \\ -2 & 1 & 6 & -8 \end{array}$
9	$\begin{array}{cccc c} 2 & 4 & 7 & 11 & \\ 1 & -3 & 7 & 12 & \\ 1 & 2 & 4 & 5 & \\ 3 & 6 & 12 & 17 & \\ \hline 6 & 2 & 1 & -4 & \\ 3 & 3 & -4 & 2 & \\ 3 & 1 & 0 & -1 & \\ 9 & 3 & 0 & -2 & \end{array}$	$\begin{array}{ccc c} 2 & 1 & 0 & \\ 3 & 2 & 0 & \\ 3 & 2 & 1 & \end{array}$	$\begin{array}{ccc c} 1 & 3 & 0 & \\ 1 & 2 & 0 & \\ 2 & 1 & 1 & \\ \hline 2 & 8 & 5 & 9 \\ 3 & 1 & 5 & 8 \\ 5 & 9 & 10 & 17 \\ -1 & 7 & 0 & 1 \end{array}$

10	$\begin{vmatrix} 6 & 2 & 1 & -4 \\ 3 & 3 & -4 & 2 \\ 3 & 1 & 0 & -1 \\ 9 & 3 & 0 & -2 \end{vmatrix}$ $\begin{vmatrix} 2 & -4 & 4 & 11 \\ 1 & -9 & 2 & 6 \\ 1 & -2 & 1 & 4 \\ 3 & -6 & 3 & 11 \end{vmatrix}$	$\begin{vmatrix} 1 & 3 & 0 \\ 1 & 2 & 0 \\ 2 & 1 & 1 \end{vmatrix}$	$\begin{vmatrix} -6 & 5 & -2 \\ 3 & -3 & 1 \\ 1 & 1 & 2 \end{vmatrix}$ $\begin{vmatrix} 2 & 9 & 5 & 1 \\ 1 & 3 & 4 & 1 \\ 3 & 12 & 9 & 2 \\ 1 & 6 & 1 & 0 \end{vmatrix}$
11	$\begin{vmatrix} 2 & -4 & 4 & 11 \\ 1 & -9 & 2 & 6 \\ 1 & -2 & 1 & 4 \\ 3 & -6 & 3 & 11 \end{vmatrix}$ $\begin{vmatrix} 2 & -2 & 2 & 4 \\ 1 & 7 & 1 & -1 \\ 1 & -1 & -1 & 3 \\ 8 & -3 & -3 & 10 \end{vmatrix}$	$\begin{vmatrix} -6 & 5 & -2 \\ 3 & -3 & 1 \\ 1 & 1 & 2 \end{vmatrix}$	$\begin{vmatrix} 3 & 0 & 1 \\ 2 & 5 & 3 \\ 1 & 4 & -1 \end{vmatrix}$ $\begin{vmatrix} 2 & 9 & 0 & 5 \\ 4 & 3 & 2 & 7 \\ 6 & 12 & 2 & 12 \\ -2 & 6 & -2 & -2 \end{vmatrix}$
12	$\begin{vmatrix} 2 & -2 & 2 & 4 \\ 1 & 7 & 1 & -1 \\ 1 & -1 & -1 & 3 \\ 8 & -3 & -3 & 10 \end{vmatrix}$ $\begin{vmatrix} 8 & 6 & 5 & 9 \\ 4 & 5 & -2 & 4 \\ 4 & 3 & 2 & 3 \\ 12 & 9 & 6 & 11 \end{vmatrix}$	$\begin{vmatrix} 1 & -3 & 0 \\ -1 & 4 & 0 \\ 0 & 0 & 1 \end{vmatrix}$	$\begin{vmatrix} -6 & -4 & -5 \\ 1 & 1 & -1 \\ 1 & 0 & 0 \end{vmatrix}$ $\begin{vmatrix} 2 & 6 & 9 & 1 \\ 2 & 2 & 4 & 3 \\ 4 & 8 & 13 & 4 \\ 0 & 4 & 5 & -2 \end{vmatrix}$
13	$\begin{vmatrix} 8 & 6 & 5 & 9 \\ 4 & 5 & -2 & 4 \\ 4 & 3 & 2 & 3 \\ 12 & 9 & 6 & 11 \end{vmatrix}$ $\begin{vmatrix} 6 & 4 & -4 & 5 \\ 3 & 3 & -2 & 6 \\ 3 & 2 & -3 & 4 \\ 9 & 6 & -9 & 11 \end{vmatrix}$	$\begin{vmatrix} -6 & -4 & -5 \\ 1 & 1 & -1 \\ 1 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & -2 & 3 \end{vmatrix}$ $\begin{vmatrix} 2 & 6 & 3 & 3 \\ 4 & 5 & 5 & 3 \\ 6 & 11 & 8 & 6 \\ -2 & 1 & -2 & 0 \end{vmatrix}$
14	$\begin{vmatrix} 6 & 4 & -4 & 5 \\ 3 & 3 & -2 & 6 \\ 3 & 2 & -3 & 4 \\ 9 & 6 & -9 & 11 \end{vmatrix}$ $\begin{vmatrix} 4 & 14 & 5 & 12 \\ 2 & 10 & 0 & 6 \\ 2 & 7 & -1 & 4 \\ 6 & 21 & -3 & 13 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & -2 & 3 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & -1 \\ 0 & 2 & -3 \\ 0 & -1 & 2 \end{vmatrix}$ $\begin{vmatrix} 2 & 5 & 5 & 6 \\ 7 & 4 & 3 & 2 \\ 9 & 9 & 8 & 8 \\ -5 & 1 & 2 & 4 \end{vmatrix}$
15	$\begin{vmatrix} 4 & 14 & 5 & 12 \\ 2 & 10 & 0 & 6 \\ 2 & 7 & -1 & 4 \\ 6 & 21 & -3 & 13 \end{vmatrix}$ $\begin{vmatrix} -2 & 2 & -1 & 1 \\ -1 & 3 & 4 & 3 \\ -1 & 1 & 3 & 0 \\ -3 & 3 & 9 & 2 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & -1 \\ 0 & 2 & -3 \\ 0 & -1 & 2 \end{vmatrix}$	$\begin{vmatrix} -2 & 0 & 5 \\ 1 & 1 & -5 \\ 1 & 0 & -2 \end{vmatrix}$ $\begin{vmatrix} 2 & 9 & 9 & 8 \\ 7 & 2 & 3 & 4 \\ 9 & 11 & 12 & 12 \\ -5 & 7 & 6 & 4 \end{vmatrix}$
16	$\begin{vmatrix} 1 & 2 & 2 & 4 \\ 1 & 6 & 0 & 5 \\ 1 & 2 & -1 & 3 \\ 1 & 2 & -1 & 2 \end{vmatrix}$ $\begin{vmatrix} -2 & 2 & -2 & 1 \\ -1 & 2 & 2 & 4 \\ -1 & 1 & 0 & 1 \\ -3 & 3 & 0 & 4 \end{vmatrix}$	$\begin{vmatrix} 1 & -1 & 0 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix}$	$\begin{vmatrix} 3 & 0 & 1 \\ 2 & 5 & 3 \\ 1 & 4 & -1 \end{vmatrix}$ $\begin{vmatrix} 2 & 7 & 9 & 7 \\ 6 & 9 & 5 & 4 \\ 8 & 16 & 14 & 11 \\ -4 & -2 & 4 & 3 \end{vmatrix}$

17	$\begin{array}{cccc c} -2 & 2 & -2 & 1 & \\ -1 & 2 & 2 & 4 & \\ -1 & 1 & 0 & 1 & \\ -3 & 3 & 0 & 4 & \\ \hline 4 & 8 & -3 & 6 & \\ 2 & 5 & 0 & 3 & \\ 2 & 4 & -3 & 1 & \\ 6 & 12 & -9 & 2 & \end{array}$	$\begin{array}{ccc c} 2 & 3 & 1 & \\ 3 & 4 & 2 & \\ 1 & 1 & 2 & \end{array}$	$\begin{array}{ccc c} 4 & 3 & 0 & \\ 1 & 1 & 0 & \\ 0 & 0 & 1 & \\ \hline 2 & 8 & 4 & 0 \\ 8 & 9 & 5 & 1 \\ 10 & 17 & 9 & 1 \\ -6 & -1 & -1 & -1 \end{array}$
18	$\begin{array}{cccc c} 4 & 8 & -3 & 6 & \\ 2 & 5 & 0 & 3 & \\ 2 & 4 & -3 & 1 & \\ 6 & 12 & -9 & 2 & \\ \hline 6 & -2 & 9 & 7 & \\ 3 & 0 & 8 & -1 & \\ 3 & -1 & 4 & 2 & \\ 9 & -3 & 12 & 7 & \end{array}$	$\begin{array}{ccc c} 4 & 3 & 0 & \\ 1 & 1 & 0 & \\ 0 & 0 & 1 & \end{array}$	$\begin{array}{ccc c} 0 & 0 & 1 & \\ 1 & 5 & 1 & \\ 1 & 4 & 2 & \\ \hline 2 & 0 & 8 & 6 \\ 6 & 9 & 4 & 3 \\ 8 & 9 & 12 & 9 \\ -4 & -9 & 4 & 3 \end{array}$
19	$\begin{array}{cccc c} 6 & -2 & 9 & 7 & \\ 3 & 0 & 8 & -1 & \\ 3 & -1 & 4 & 2 & \\ 9 & -3 & 12 & 7 & \\ \hline 4 & -4 & -1 & 12 & \\ 2 & -3 & 2 & 6 & \\ 2 & -2 & 1 & 4 & \\ 6 & -6 & 3 & 13 & \end{array}$	$\begin{array}{ccc c} 0 & 0 & 1 & \\ 1 & 5 & 1 & \\ 1 & 4 & 2 & \end{array}$	$\begin{array}{ccc c} 1 & 2 & 1 & \\ 0 & 3 & 1 & \\ 0 & 2 & 1 & \\ \hline 2 & 3 & 7 & 4 \\ 4 & 5 & 7 & 2 \\ 6 & 8 & 14 & 6 \\ -2 & -2 & 0 & 2 \end{array}$
20	$\begin{array}{cccc c} 4 & -4 & -1 & 12 & \\ 2 & -3 & 2 & 6 & \\ 2 & -2 & 1 & 4 & \\ 6 & -6 & 3 & 13 & \\ \hline -4 & 2 & 1 & -4 & \\ -2 & 2 & 3 & 3 & \\ -2 & 1 & 0 & -1 & \\ -6 & 3 & 0 & -2 & \end{array}$	$\begin{array}{ccc c} 1 & 2 & 1 & \\ 0 & 3 & 1 & \\ 0 & 2 & 1 & \end{array}$	$\begin{array}{ccc c} 1 & 1 & 2 & \\ 0 & 2 & 3 & \\ 0 & 1 & 2 & \\ \hline 2 & 7 & 4 & 9 \\ 3 & 5 & 2 & 8 \\ 5 & 12 & 6 & 17 \\ -1 & 2 & 2 & 1 \end{array}$
21	$\begin{array}{cccc c} -4 & 2 & 1 & -4 & \\ -2 & 2 & 3 & 3 & \\ -2 & 1 & 0 & -1 & \\ -6 & 3 & 0 & -2 & \\ \hline -6 & 2 & 3 & 10 & \\ -3 & -1 & 1 & 4 & \\ -3 & 1 & 0 & 3 & \\ -9 & 3 & 0 & 8 & \end{array}$	$\begin{array}{ccc c} 1 & 1 & 2 & \\ 0 & 2 & 3 & \\ 0 & 1 & 2 & \end{array}$	$\begin{array}{ccc c} -2 & 0 & 5 & \\ 3 & 1 & 5 & \\ 1 & 0 & 2 & \\ \hline 2 & 5 & 9 & 6 \\ 5 & 7 & 6 & 3 \\ 7 & 12 & 15 & 9 \\ -3 & -2 & -3 & 3 \end{array}$
22	$\begin{array}{cccc c} -6 & 2 & 3 & 10 & \\ -3 & -1 & 1 & 4 & \\ -3 & 1 & 0 & 3 & \\ -9 & 3 & 0 & 8 & \\ \hline 2 & 4 & -5 & 3 & \\ 1 & 4 & -4 & 4 & \\ 1 & 2 & -3 & 1 & \\ 3 & 6 & -9 & 5 & \end{array}$	$\begin{array}{ccc c} 2 & 0 & 5 & \\ 3 & 1 & 5 & \\ 1 & 0 & 2 & \end{array}$	$\begin{array}{ccc c} 1 & 0 & 1 & \\ 2 & 2 & 3 & \\ 3 & 0 & 4 & \\ \hline 2 & 6 & 5 & 3 \\ 8 & 6 & 4 & 3 \\ 10 & 12 & 9 & 6 \\ -6 & 0 & 1 & 0 \end{array}$
23	$\begin{array}{cccc c} 2 & 4 & -5 & 3 & \\ 1 & 4 & -4 & 4 & \\ 1 & 2 & -3 & 1 & \\ 3 & 6 & -9 & 5 & \\ \hline 2 & 4 & 7 & 11 & \\ 1 & -3 & 7 & 12 & \\ 1 & 2 & 4 & 5 & \\ 3 & 6 & 12 & 17 & \end{array}$	$\begin{array}{ccc c} 1 & 0 & 1 & \\ 2 & 2 & 3 & \\ 3 & 0 & 4 & \end{array}$	$\begin{array}{ccc c} -2 & 1 & 0 & \\ 3 & 2 & 0 & \\ 3 & 2 & 1 & \\ \hline 2 & 6 & 9 & 1 \\ 4 & 5 & 3 & 9 \\ 6 & 11 & 12 & 10 \\ -2 & 1 & 6 & -8 \end{array}$

24	$\begin{vmatrix} 2 & 4 & 7 & 11 \\ 1 & -3 & 7 & 12 \\ 1 & 2 & 4 & 5 \\ 3 & 6 & 12 & 17 \\ \hline 6 & 2 & 1 & -4 \\ 3 & 3 & -4 & 2 \\ 3 & 1 & 0 & -1 \\ 9 & 3 & 0 & -2 \end{vmatrix}$	$\begin{pmatrix} 2 & 1 & 0 \\ 3 & 2 & 0 \\ 3 & 2 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 3 & 0 \\ 1 & 2 & 0 \\ 2 & 1 & 1 \end{pmatrix}$ $\begin{pmatrix} 2 & 8 & 5 & 9 \\ 3 & 1 & 5 & 8 \\ 5 & 9 & 10 & 17 \\ -1 & 7 & 0 & 1 \end{pmatrix}$
25	$\begin{vmatrix} 6 & 2 & 1 & -4 \\ 3 & 3 & -4 & 2 \\ 3 & 1 & 0 & -1 \\ 9 & 3 & 0 & -2 \\ \hline 2 & -4 & 4 & 11 \\ 1 & -9 & 2 & 6 \\ 1 & -2 & 1 & 4 \\ 3 & -6 & 3 & 11 \end{vmatrix}$	$\begin{pmatrix} 1 & 3 & 0 \\ 1 & 2 & 0 \\ 2 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} -6 & 5 & -2 \\ 3 & -3 & 1 \\ 1 & 1 & 2 \end{pmatrix}$ $\begin{pmatrix} 2 & 9 & 5 & 1 \\ 1 & 3 & 4 & 1 \\ 3 & 12 & 9 & 2 \\ 1 & 6 & 1 & 0 \end{pmatrix}$
26	$\begin{vmatrix} 2 & -4 & 4 & 11 \\ 1 & -9 & 2 & 6 \\ 1 & -2 & 1 & 4 \\ 3 & -6 & 3 & 11 \\ \hline 2 & -2 & 2 & 4 \\ 1 & 7 & 1 & -1 \\ 1 & -1 & -1 & 3 \\ 8 & -3 & -3 & 10 \end{vmatrix}$	$\begin{pmatrix} -6 & 5 & -2 \\ 3 & -3 & 1 \\ 1 & 1 & 2 \end{pmatrix}$	$\begin{pmatrix} 3 & 0 & 1 \\ 2 & 5 & 3 \\ 1 & 4 & -1 \end{pmatrix}$ $\begin{pmatrix} 2 & 9 & 0 & 5 \\ 4 & 3 & 2 & 7 \\ 6 & 12 & 2 & 12 \\ -2 & 6 & -2 & -2 \end{pmatrix}$
27	$\begin{vmatrix} 2 & -2 & 2 & 4 \\ 1 & 7 & 1 & -1 \\ 1 & -1 & -1 & 3 \\ 8 & -3 & -3 & 10 \\ \hline 8 & 6 & 5 & 9 \\ 4 & 5 & -2 & 4 \\ 4 & 3 & 2 & 3 \\ 12 & 9 & 6 & 11 \end{vmatrix}$	$\begin{pmatrix} 1 & -3 & 0 \\ -1 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -6 & -4 & -5 \\ 1 & 1 & -1 \\ 1 & 0 & 0 \end{pmatrix}$ $\begin{pmatrix} 2 & 6 & 9 & 1 \\ 2 & 2 & 4 & 3 \\ 4 & 8 & 13 & 4 \\ 0 & 4 & 5 & -2 \end{pmatrix}$
28	$\begin{vmatrix} 8 & 6 & 5 & 9 \\ 4 & 5 & -2 & 4 \\ 4 & 3 & 2 & 3 \\ 12 & 9 & 6 & 11 \\ \hline 6 & 4 & -4 & 5 \\ 3 & 3 & -2 & 6 \\ 3 & 2 & -3 & 4 \\ 9 & 6 & -9 & 11 \end{vmatrix}$	$\begin{pmatrix} -6 & -4 & -5 \\ 1 & 1 & -1 \\ 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & -2 & 3 \end{pmatrix}$ $\begin{pmatrix} 2 & 6 & 3 & 3 \\ 4 & 5 & 5 & 3 \\ 6 & 11 & 8 & 6 \\ -2 & 1 & -2 & 0 \end{pmatrix}$
29	$\begin{vmatrix} 6 & 4 & -4 & 5 \\ 3 & 3 & -2 & 6 \\ 3 & 2 & -3 & 4 \\ 9 & 6 & -9 & 11 \\ \hline 4 & 14 & 5 & 12 \\ 2 & 10 & 0 & 6 \\ 2 & 7 & -1 & 4 \\ 6 & 21 & -3 & 13 \end{vmatrix}$	$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & -2 & 3 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & -3 \\ 0 & -1 & 2 \end{pmatrix}$ $\begin{pmatrix} 2 & 5 & 5 & 6 \\ 7 & 4 & 3 & 2 \\ 9 & 9 & 8 & 8 \\ -5 & 1 & 2 & 4 \end{pmatrix}$

30	$\begin{vmatrix} 4 & 14 & 5 & 12 \\ 2 & 10 & 0 & 6 \\ 2 & 7 & -1 & 4 \\ 6 & 21 & -3 & 13 \end{vmatrix}$ $\begin{vmatrix} -2 & 2 & -1 & 1 \\ -1 & 3 & 4 & 3 \\ -1 & 1 & 3 & 0 \\ -3 & 3 & 9 & 2 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & -1 \\ 0 & 2 & -3 \\ 0 & -1 & 2 \end{vmatrix}$	$\begin{vmatrix} -2 & 0 & 5 \\ 1 & 1 & -5 \\ 1 & 0 & -2 \end{vmatrix}$ $\begin{vmatrix} 2 & 9 & 9 & 8 \\ 7 & 2 & 3 & 4 \\ 9 & 11 & 12 & 12 \\ -5 & 7 & 6 & 4 \end{vmatrix}$
----	---	--	---

Task 2. Solve the SLAE by the inverse matrix method and using the Given–Find block;

Variant №	Task 2 a	Task 2 b
1	$\begin{cases} 3x + 6y + 5z + t = 0 \\ 3x + 4y + z + 2t = 1 \\ 5x + 4y + 7z + 8t = 0 \\ 6x + 10y + 6z + 5t = 1 \end{cases}$	$\begin{cases} 3x + 6y + 5z = 1 \\ 3x + 4y + z = 2 \\ 5x + 4y + 7z = 8 \\ 6x + 10y + 6z = 3 \end{cases}$
2	$\begin{cases} 3x + 5y + 6z + 3t = 1 \\ x + y + 3z + 4t = 0 \\ 3x + 2y + 4z + 4t = 1 \\ 4x + y + 9z + t = 0 \end{cases}$	$\begin{cases} 3x + 5y + 6z = 3 \\ x + y + 3z = 4 \\ 3x + 2y + 4z = 4 \\ 4x + 6y + 9z = 7 \end{cases}$
3	$\begin{cases} 3x + 4y + 7z + 8t = 1 \\ 9x + 5y + 4z + 5t = 0 \\ x + 2y + z + t = 1 \\ 12x + y + 11z + t = 13 \end{cases}$	$\begin{cases} 3x + 4y + 7z = 8 \\ 9x + 5y + 4z = 5 \\ x + 2y + z = 0 \\ 12x + 9y + 11z = 13 \end{cases}$
4	$\begin{cases} 3x + 6y + 5z + 4t = 0 \\ 4x + 6y + 4z + t = 1 \\ 2x + 3y + z + 3t = 0 \\ 7x + y + 9z + t = 15 \end{cases}$	$\begin{cases} 3x + 6y + 5z = 4 \\ 4x + 6y + 4z = 1 \\ 2x + 3y + z = 3 \\ 7x + 12y + 9z = 5 \end{cases}$
5	$\begin{cases} 3x + 2y + 4t = 0 \\ 2x + 3y + 2z + t = 1 \\ x + 5y + 7z + 3t = 0 \\ 5x + y + 2z + t = 15 \end{cases}$	$\begin{cases} 3x + 2y = 4 \\ 2x + 3y + 2z = 1 \\ x + 5y + 7z = 3 \\ 5x + 5y + 2z = 5 \end{cases}$
6	$\begin{cases} 3x + 9y + 8z + 5t = 0 \\ x + 3y + 3z + t = 1 \\ 2x + y + 3z + t = 0 \\ 4x + y + 11z + t = 16 \end{cases}$	$\begin{cases} 3x + 9y + 8z = 5 \\ x + 3y + 3z = 1 \\ 2x + y + 3z = 1 \\ 4x + 12y + 11z = 6 \end{cases}$
7	$\begin{cases} 3x + 5y + 3z + 4t = 0 \\ x + 7y + 6z + 6t = 1 \\ 3x + y + 8z + 6t = 0 \\ 4x + y + 9z + t = 10 \end{cases}$	$\begin{cases} 3x + 5y + 3z = 4 \\ x + 7y + 6z = 6 \\ 3x + y + 8z = 6 \\ 4x + 12y + 9z = 10 \end{cases}$
8	$\begin{cases} 3x + 7y + 5z + t = 0 \\ x + 2y + 3z + t = 1 \\ 3x + 2y + 6z + 7t = 0 \\ 4x + y + 8z + t = 10 \end{cases}$	$\begin{cases} 3x + 7y + 5z = 1 \\ x + 2y + 3z = 1 \\ 3x + 2y + 6z = 7 \\ 4x + 9y + 8z = 2 \end{cases}$
9	$\begin{cases} 3x + y + 5z + t = 1 \\ x + 2y + z + 4t = 0 \\ 3x + 2y + 6z + 5t = 1 \\ 4x + 3y + z + t = 15 \end{cases}$	$\begin{cases} 3x + y + 5z = 1 \\ x + 2y + z = 4 \\ 3x + 2y + 6z = 5 \\ 4x + 3y + 6z = 5 \end{cases}$

10	$\begin{cases} 3x + 6y + 9z + t = 1 \\ 5x + 6y + 6z + 7t = 0 \\ 4x + 4y + 6z + t = 10 \\ 8x + 12y + z + 2t = 18 \end{cases}$	$\begin{cases} 3x + 6y + 9z = 1 \\ 5x + 6y + 6z = 7 \\ 4x + 4y + 6z = 1 \\ 8x + 12y + 15z = 8 \end{cases}$
11	$\begin{cases} 3x + 5y + 4z + 6t = 16 \\ 5x + 4y + 6z + t = 1 \\ 3x + y + 2z + t = 0 \\ 8x + y + 10z + 2t = 17 \end{cases}$	$\begin{cases} 3x + 5y + 4z = 6 \\ 5x + 4y + 6z = 1 \\ 3x + y + 2z = 1 \\ 8x + 9y + 10z = 7 \end{cases}$
12	$\begin{cases} 3x + 5y + z + 8t = 0 \\ 4x + 3y + 6z + 6t = 16 \\ 3x + y + 2z + 5t = 1 \\ 7x + y + 7z + t = 1 \end{cases}$	$\begin{cases} 3x + 5y + z = 8 \\ 4x + 3y + 6z = 6 \\ 3x + y + 2z = 5 \\ 7x + 8y + 7z = 14 \end{cases}$
13	$\begin{cases} 3x + 6y + 5z + 4t = 0 \\ x + 3y + 4z + 5t = 15 \\ 4x + 3y + 5z + 2t = 1 \\ 4x + y + 9z + 9t = 0 \end{cases}$	$\begin{cases} 3x + 6y + 5z = 4 \\ x + 3y + 4z = 5 \\ 4x + 3y + 5z = 2 \\ 4x + 9y + 9z = 9 \end{cases}$
14	$\begin{cases} 3x + 5y + 3z + 6t = 16 \\ 3x + 6y + 7z + t = 0 \\ 5x + 9y + 7z + 8t = 18 \\ 6x + y + 10z + t = 1 \end{cases}$	$\begin{cases} 3x + 5y + 3z = 6 \\ 3x + 6y + 7z = 1 \\ 5x + 9y + 7z = 8 \\ 6x + 11y + 10z = 7 \end{cases}$
15	$\begin{cases} 3x + 5y + 3z + 9t = 19 \\ x + 2y + 3z + 5t = 0 \\ 7x + 7y + 6z + 5t = 1 \\ 4x + y + 6z + t = 1 \end{cases}$	$\begin{cases} 3x + 5y + 3z = 9 \\ x + 2y + 3z = 5 \\ 7x + 7y + 6z = 5 \\ 4x + 7y + 6z = 14 \end{cases}$
16	$\begin{cases} 3x + 6y + 5z + t = 0 \\ 3x + 4y + z + 2t = 1 \\ 5x + 4y + 7z + 8t = 0 \\ 6x + 10y + 6z + 5t = 1 \end{cases}$	$\begin{cases} 3x + 6y + 5z = 1 \\ 3x + 4y + z = 2 \\ 5x + 4y + 7z = 8 \\ 6x + 10y + 6z = 3 \end{cases}$
17	$\begin{cases} 3x + 5y + 6z + 3t = 1 \\ x + y + 3z + 4t = 0 \\ 3x + 2y + 4z + 4t = 1 \\ 4x + y + 9z + t = 0 \end{cases}$	$\begin{cases} 3x + 5y + 6z = 3 \\ x + y + 3z = 4 \\ 3x + 2y + 4z = 4 \\ 4x + 6y + 9z = 7 \end{cases}$
18	$\begin{cases} 3x + 4y + 7z + 8t = 1 \\ 9x + 5y + 4z + 5t = 0 \\ x + 2y + z + t = 1 \\ 12x + y + 11z + t = 13 \end{cases}$	$\begin{cases} 3x + 4y + 7z = 8 \\ 9x + 5y + 4z = 5 \\ x + 2y + z = 0 \\ 12x + 9y + 11z = 13 \end{cases}$
19	$\begin{cases} 3x + 6y + 5z + 4t = 0 \\ 4x + 6y + 4z + t = 1 \\ 2x + 3y + z + 3t = 0 \\ 7x + y + 9z + t = 15 \end{cases}$	$\begin{cases} 3x + 6y + 5z = 4 \\ 4x + 6y + 4z = 1 \\ 2x + 3y + z = 3 \\ 7x + 12y + 9z = 5 \end{cases}$
20	$\begin{cases} 3x + 2y + 4t = 0 \\ 2x + 3y + 2z + t = 1 \\ x + 5y + 7z + 3t = 0 \\ 5x + y + 2z + t = 15 \end{cases}$	$\begin{cases} 3x + 2y = 4 \\ 2x + 3y + 2z = 1 \\ x + 5y + 7z = 3 \\ 5x + 5y + 2z = 5 \end{cases}$
21	$\begin{cases} 3x + 9y + 8z + 5t = 0 \\ x + 3y + 3z + t = 1 \\ 2x + y + 3z + t = 0 \\ 4x + y + 11z + t = 16 \end{cases}$	$\begin{cases} 3x + 9y + 8z = 5 \\ x + 3y + 3z = 1 \\ 2x + y + 3z = 1 \\ 4x + 12y + 11z = 6 \end{cases}$

22	$\begin{cases} 3x + 5y + 3z + 4t = 0 \\ x + 7y + 6z + 6t = 1 \\ 3x + y + 8z + 6t = 0 \\ 4x + y + 9z + t = 10 \end{cases}$	$\begin{cases} 3x + 5y + 3z = 4 \\ x + 7y + 6z = 6 \\ 3x + y + 8z = 6 \\ 4x + 12y + 9z = 10 \end{cases}$
23	$\begin{cases} 3x + 7y + 5z + t = 0 \\ x + 2y + 3z + t = 1 \\ 3x + 2y + 6z + 7t = 0 \\ 4x + y + 8z + t = 10 \end{cases}$	$\begin{cases} 3x + 7y + 5z = 1 \\ x + 2y + 3z = 1 \\ 3x + 2y + 6z = 7 \\ 4x + 9y + 8z = 2 \end{cases}$
24	$\begin{cases} 3x + y + 5z + t = 1 \\ x + 2y + z + 4t = 0 \\ 3x + 2y + 6z + 5t = 1 \\ 4x + 3y + z + t = 15 \end{cases}$	$\begin{cases} 3x + y + 5z = 1 \\ x + 2y + z = 4 \\ 3x + 2y + 6z = 5 \\ 4x + 3y + 6z = 5 \end{cases}$
25	$\begin{cases} 3x + 6y + 9z + t = 1 \\ 5x + 6y + 6z + 7t = 0 \\ 4x + 4y + 6z + t = 10 \\ 8x + 12y + z + 2t = 18 \end{cases}$	$\begin{cases} 3x + 6y + 9z = 1 \\ 5x + 6y + 6z = 7 \\ 4x + 4y + 6z = 1 \\ 8x + 12y + 15z = 8 \end{cases}$
26	$\begin{cases} 3x + 5y + 4z + 6t = 16 \\ 5x + 4y + 6z + t = 1 \\ 3x + y + 2z + t = 0 \\ 8x + y + 10z + 2t = 17 \end{cases}$	$\begin{cases} 3x + 5y + 4z = 6 \\ 5x + 4y + 6z = 1 \\ 3x + y + 2z = 1 \\ 8x + 9y + 10z = 7 \end{cases}$
27	$\begin{cases} 3x + 5y + z + 8t = 0 \\ 4x + 3y + 6z + 6t = 16 \\ 3x + y + 2z + 5t = 1 \\ 7x + y + 7z + t = 1 \end{cases}$	$\begin{cases} 3x + 5y + z = 8 \\ 4x + 3y + 6z = 6 \\ 3x + y + 2z = 5 \\ 7x + 8y + 7z = 14 \end{cases}$
28	$\begin{cases} 3x + 6y + 5z + 4t = 0 \\ x + 3y + 4z + 5t = 15 \\ 4x + 3y + 5z + 2t = 1 \\ 4x + y + 9z + 9t = 0 \end{cases}$	$\begin{cases} 3x + 6y + 5z = 4 \\ x + 3y + 4z = 5 \\ 4x + 3y + 5z = 2 \\ 4x + 9y + 9z = 9 \end{cases}$
29	$\begin{cases} 3x + 5y + 3z + 6t = 16 \\ 3x + 6y + 7z + t = 0 \\ 5x + 9y + 7z + 8t = 18 \\ 6x + y + 10z + t = 1 \end{cases}$	$\begin{cases} 3x + 5y + 3z = 6 \\ 3x + 6y + 7z = 1 \\ 5x + 9y + 7z = 8 \\ 6x + 11y + 10z = 7 \end{cases}$
30	$\begin{cases} 3x + 5y + 3z + 9t = 19 \\ x + 2y + 3z + 5t = 0 \\ 7x + 7y + 6z + 5t = 1 \\ 4x + y + 6z + t = 1 \end{cases}$	$\begin{cases} 3x + 5y + 3z = 9 \\ x + 2y + 3z = 5 \\ 7x + 7y + 6z = 5 \\ 4x + 7y + 6z = 14 \end{cases}$

Work performance technology

In this work, it is necessary to perform actions with vectors and matrices using the Matrix panel, solve the SLAE using the methods of the inverse matrix, least squares, using the Given - Find block.

Test questions:

1. How to calculate the modulus of a vector in Mathcad?
2. How to solve a system of equations using a symbolic processor?
3. How to highlight a row of a given matrix in Mathcad?
4. How to build a matrix of values of a function of two variables?

Practical lesson No3

Fundamentals of work in the MatLab system. System capabilities. Program interface. Creation of MatLab-document and their application in technical systems.

The purpose of the lesson: to study the user interface of the MATLAB system and the basics of working with the system in the direct calculation mode.

Theoretical part

MATLAB includes a command interpreter, a graphical shell, a debugger editor, command libraries, a compiler, the symbolic core of the Maple package for analytical calculations, MATLAB mathematical libraries in C / C ++, a report generator and wide toolboxes.

The MATLAB interface is quite consistent with modern canons (see Figure 1). It is multi-windowed and has a number of means of direct access to various system components. Pay attention to the following toolbar buttons:

- **New M-file** - displays an empty m-file editor window;
- **Open file** - opens a window for loading Matlab files;
- **Simulink** - Opens the Simulink Library Browser window.
- **Help** - opens a help window.

These functions are duplicated in a very simple MATLAB system menu.

Windows with **Launch Pad/Workspace** tabs for access to system components and tabs for the current directory **Current Directory** and session history **History** appeared in the left part of the system window. They provide operational control over the state of the system. Displayed MATLAB interface windows can be enabled or disabled from the View menu item.

All work is organized through the command window (**Command Window**), which appears when you start the program. In the course of work, the data is located in memory (**Workspace**) in the form of matrices.

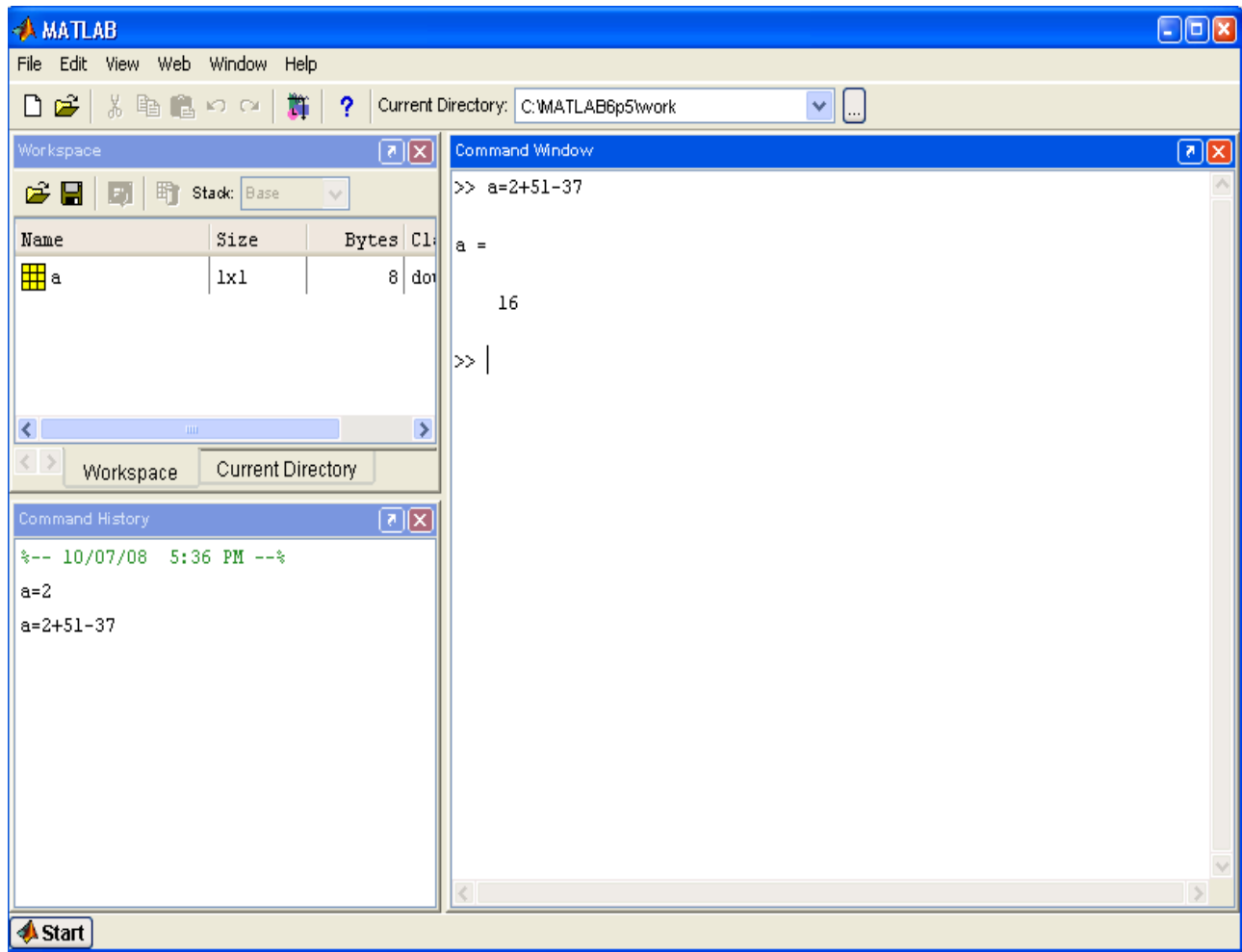


Figure 1 - MATLAB program interface

Variables in MATLAB do not need to be pre-declared by specifying their type. All data is stored as arrays: numeric variables (internal type numeric), text strings (char), cells (cell), and structures (struct). A two-dimensional array is a matrix, a one-dimensional array is a vector, and a scalar is a 1x1 matrix. A variable name must begin with a letter, followed by letters, numbers, and underscores. Names of any length are allowed, but MATLAB identifies them by the first 31 characters and distinguishes between uppercase and lowercase letters. MATLAB has a number of constants (Table 1)

Table 1.

Reserved constant names.

Name	Description
ans	The result of the last operation
i, j	Imaginary unit
pi	Number π
eps	Machine Precision
realmax	Maximum real number

realmin	Minimum real number
inf	Infinity
NaN	Non-numeric variable
end	The largest value of the array dimension index

Note that the name NaN (Not-a-Number) is reserved for the result of operations $0/0$, $0*\text{inf}$, $\text{inf}-\text{inf}$, and so on.

Calculations are performed in the command window in dialog mode. The user enters commands or launches files with texts in the MATLAB language. The interpreter processes the entered value and returns the results: numeric and string data, warnings and error messages. The input line is marked with `>>`.

Variable names must start with a letter. The `=` sign corresponds to the assignment operator. Pressing the Enter key causes the system to evaluate the expression and display the result. If the operator record does not end with the symbol `;`, then the result is displayed in the command window, otherwise it is not displayed. If the statement does not contain an assignment sign `=`, then the value of the result is assigned to the `ans` system variable. All values of variables calculated during the current session of work are stored in a specially reserved area of computer memory called the workspace of the MATLAB system (**Workspace**). To view the value of any variable from the current workspace of the system, just type its name and press the Enter key.

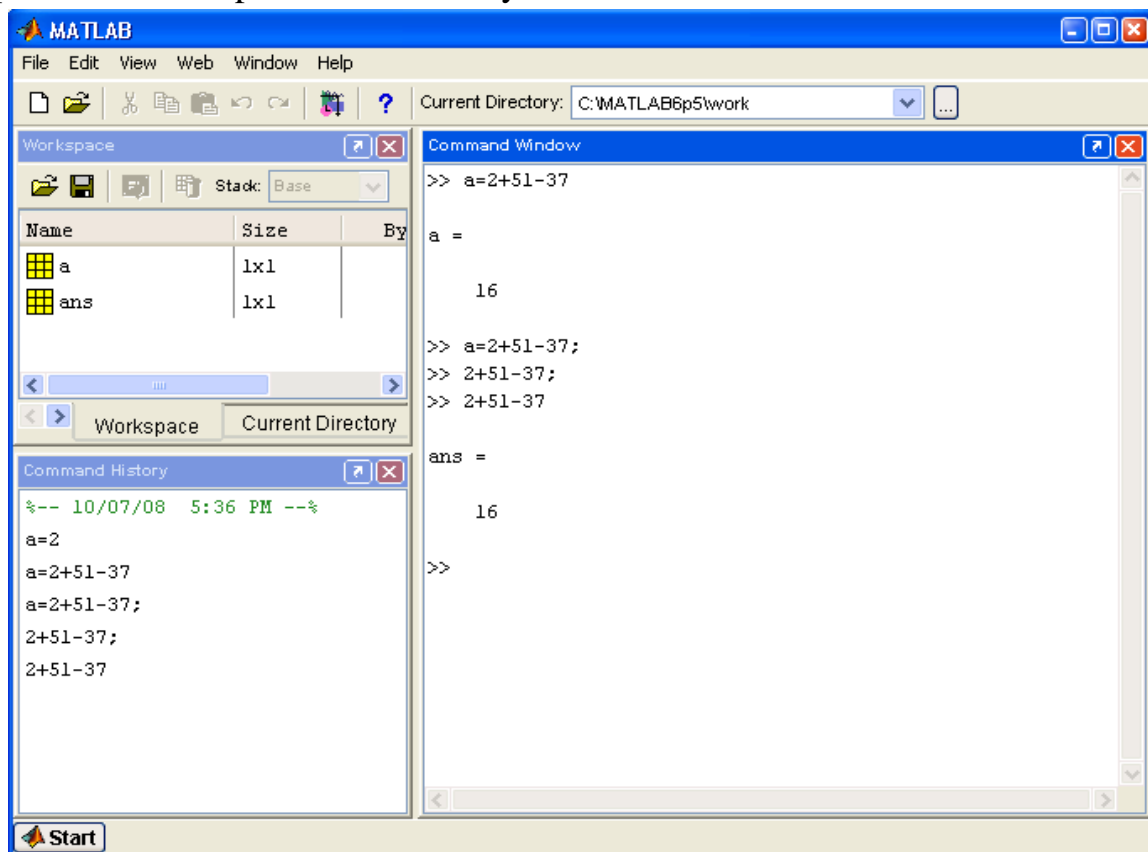


Fig.2. - Demonstration of the execution of the assignment command

After the end of the session with the MATLAB system, all previously calculated variables are lost. To save the contents of the workspace of the MATLAB system to a file on the computer disk, you need to execute the menu command **File \Save Workspace As** By default, the file name extension is mat, so such files are usually called MAT-files.

The MATLAB system works with both real and complex numbers. Before using complex number operations, you must define the variable $i = \text{sqrt}(-1)$ or $j = \text{sqrt}(-1)$. The following operation signs are used in arithmetic expressions:

- + , - – addition, subtraction,
- *- multiplication,
- / - division from left to right;
- \ - division from right to left;
- ^ - exponentiation.

The MATLAB system allows you to calculate various mathematical functions. The following elementary algebraic functions take one or two real (x, y) or one complex (z) number as an argument (Table 2).

Table 2

Elementary algebraic functions

Function	Description
abs(z), abs(x),	Compute the modulus of a complex number z or the absolute value of a real number x.
angle(z)	Calculation of the argument z.
sqrt(z), sqrt(x)	Calculating the square root of numbers z and x
real(z)	Calculation of the real part of the complex number z.
imag(z)	Calculation of the imaginary part of the complex number z.
round(x)	Rounding up to an integer.
fix(x)	Rounding to the nearest integer towards zero.
rem(x, y)	Calculation of the remainder after dividing x by y.
exp(z)	Calculate e to the power of x.
log(z)	Calculation of the natural logarithm of a number x.
log10(z)	Calculation of the decimal logarithm of the number x.

The MATLAB system provides the ability to calculate the following trigonometric and inverse trigonometric functions of the x variable (Table 3).

Trigonometric functions

Function	Description
sin (x)	Sine Calculation
cos (x)	Cosine calculation
tan (x)	Tangent calculation
asin (x)	Arcsine calculation
acos (x)	Arc cosine calculation
atan (x)	Arc tangent calculation
atan2 (y, x)	Calculation of the arc tangent from the coordinates of a point

WORKING WITH GRAPHICS IN MATLAB

3.1 Basic theoretical information

One of the advantages of the MATLAB system is the abundance of graphics tools, ranging from commands for constructing simple graphs of functions of one variable in the Cartesian coordinate system to combined and presentation graphs with animation elements, as well as graphical user interface (GUI) design tools. Particular attention in the system is paid to three-dimensional graphics with functional coloring of the displayed figures and imitation of various lighting effects.

To display the functions of one variable $y(x)$, graphs are used in the Cartesian (rectangular) coordinate system. In this case, two axes are usually built - horizontal X and vertical Y, and coordinates x and y are given, which determine the nodal points of the function $y(x)$. Since MATLAB is a matrix system, the set of points $y(x)$ is given by X and Y vectors of the same size.

The plot (X, Y) command is used to plot function graphs in the Cartesian coordinate system, the coordinates of points (x, y) are taken from vectors of the same size Y and X. If X or Y is a matrix, then a family of graphs is built according to the data contained in the columns matrices.

The plot(X, Y, S) command is similar to the plot(X, Y) command, but the plot line type can be specified using the string constant S.

The values of the constant S can be the following characters, which are presented in Table 4.

Setting the line type

Line type marker	
Marker	Line type
-	Continuous
--	Dashed
:	Punctured (dotted)
-.	Dash-dot
Graph Color Marker	
Marker	Graph color
C	Blue
M	Violet
Y	Yellow
R	Red
G	Green
B	Blue
W	White
K	Black
Point type	
Marker	Point type
.	Dot
+	Plus
*	Asterisk
O	Circle (the Latin letter o is indicated)
X	Cross (indicates the Latin letter x)

Thus, using the string constant S, you can change the color of the line, represent node points with different marks (point, circle, cross, triangle with different vertex orientations, etc.) and change the line type of the graph.

The **plot(X1, Y1, S1, X2, Y2, S2, X3, Y3, S3,...)** command builds a series of lines on one chart, represented by data of the form (X, Y, S), where X and Y are vectors or matrices, and S are rows. Using this construction, it is possible to construct, for example, a graph of a function with a line whose color differs from the color of the nodal points. If there is no indication of the color of lines and points, it is selected automatically from the color table (white is excluded). If there are more than six lines, then the choice of colors is repeated.

Sometimes you want to compare the behavior of two functions whose values are very different from each other. The graph of a function with small values practically merges with the abscissa axis, and its appearance cannot be determined. In this situation, the **plot(y)** function helps, which displays graphs in a window with two vertical axes that have a suitable scale.

Three-dimensional surfaces are usually described by a function of two variables $z(x, y)$. The specificity of building three-dimensional graphs requires not just setting a series of x and y values, that is, x and y vectors. It requires the definition for X and Y of two-dimensional arrays - matrices.

The **meshgrid** function is used to create such arrays. It is primarily used in conjunction with 3D surface plotting functions. The **meshgrid** function is written in the following forms:

– $[X,Y,Z] = \text{meshgrid}(x, y, z)$ — returns three-dimensional arrays used to calculate functions of three variables and build three-dimensional graphs;

– $[X,Y] = \text{meshgrid}(x, y)$ — converts the area specified by the x and y vectors into X and Y arrays, which can be used to calculate a function of two variables and build three-dimensional graphs. The rows of the output array X are copies of the vector x , and the columns of Y are copies of the vector y .

The **plot3(...)** command is similar to the **plot(...)** command, but refers to a function of two variables $z(x, y)$. It builds an axonometric image of three-dimensional surfaces and is represented by the following forms:

– **plot3**(x, y, z) – builds an array of points represented by x, y and z vectors, connecting them with line segments. This command is of limited use;

– **plot3**(X, Y, Z, S)—provides plotting with line and point style specification;

– **plot3**($x1, y1, z1, s1, x2, y2, z2, s2, \dots$) – draws graphs of several functions $z1(x1,y1), z2(x2,y2)$, etc. on one figure with specification of lines and markers of each of them.

The most representative and visual are **mesh** graphs of surfaces with a given or functional coloring. The name of their commands contains the word mesh. There are three groups of such commands:

– **mesh**(X, Y, Z, C) — displays a mesh surface $Z(X,Y)$ in the graphics window with the colors of the surface nodes specified by the C array;

– **mesh**(X, Y, Z) — analogue of the previous command for $C=Z$.

In this case, functional coloring is used, in which the color is given by the height of the surface. The **mesh** function returns a handle to an object of class surface. The following is an example of using the **mesh** command:

```
>> [X, Y]=meshgrid([-3:0.15:3]);  
>> Z=X.^2+Y.^2;  
>> mesh(X, Y, Z)
```

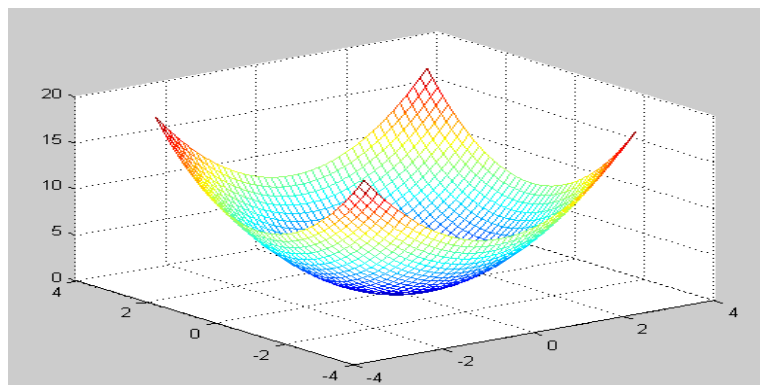


Figure 3 - Surface plot created by the mesh(X,Y,Z) command

After the graph has already been built, MATLAB allows you to format or design it in the desired form. Thus, to set a title inscription above a chart, use the following command **title('string')** — setting a title inscription on 2D and 3D charts, specified by the string constant 'string'.

The following commands are used to set labels near x, y and z axes: **xlabel('String')**, **ylabel('String')**, **zlabel('String')**.

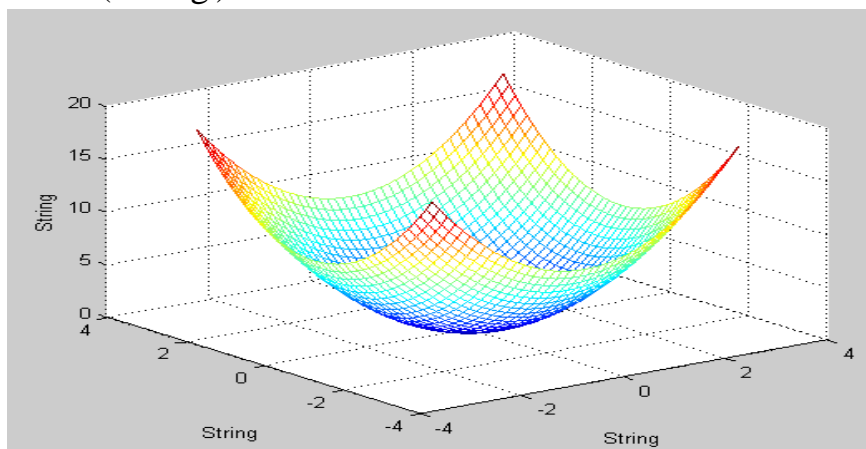


Figure 4 - Installing labels using the command:
`xlabel('String'), ylabel('String'), zlabel('String')`

Often there is a need to add text to a specific place in the graph, for example, to designate one or another curve of the graph. The `text` command is used for this:

– `text(X,Y, 'string')` — adds the text specified by the string constant 'string' to the two-dimensional chart, so that the beginning of the text is located at the point with coordinates (X, Y). If X and Y are given as one-dimensional arrays, then the label is placed in all positions [x(i), y(i)];

– `text(X,Y, Z, 'string')` — adds the text specified by the string constant 'string' to the 3D plot, so that the beginning of the text is located at the position specified by the X, Y and Z coordinates.

The `gtext` command provides a very convenient way to enter text:

– `gtext('string')` — sets the text displayed on the chart in the form of a string constant 'string' and displays a marker moved by the mouse in the form of a cross on the chart. Having set the marker in the right place, it is enough to click any mouse button to display the text.

An explanation in the form of line segments with reference labels placed inside the graph or near it is called a legend. To create a legend, various variants of the **legend** command are used:

`legend(string1, string2,..., strings)` — adds a legend to the current chart in the form of strings specified in the list of parameters;

`>> legend('chart')`

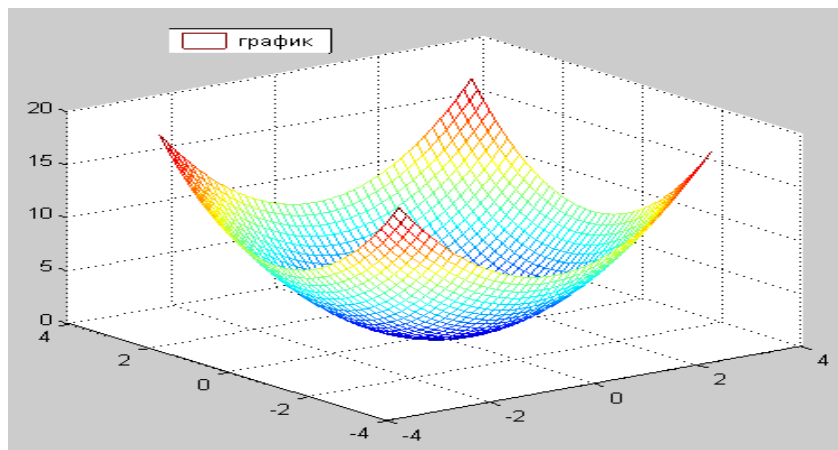


Figure 5 - Graph with explanations

legend (Pos) - places the legend at the exact location specified by the Pos parameter:

Pos=0 - the best place selected automatically;

Pos=1 — upper right corner;

Pos=2 - upper left corner;

Pos=3 — lower left corner;

Pos=4 — lower right corner;

Pos=-1 — to the right of the chart.

When adding a legend, it should be noted that the order and number of arguments of the **legend** command must correspond to the order in which graphs are displayed and their number.

Typically, graphs are displayed in auto-scaling mode. The following **axis** class commands change this situation:

- **axis**([XMIN XMAX YMIN YMAX]) — setting the ranges of coordinates along the x and y axes for the current two-dimensional chart;

- **axis**([XMIN XMAX YMIN YMAX ZMIN ZMAX]) – setting the coordinate ranges along the x, y and z axes of the current 3D chart;

- **axis** auto — setting the default axis parameters;

In mathematical, physical and other literature, when plotting graphs, in addition to marking the axes, a scale grid is often used. The grid commands allow you to set the gridding or cancel this construction:

- **grid** on — adds a grid to the current chart;

- **grid** off — turns off the grid.

In many cases, it is desirable to build many superimposed graphs in the same window. To do this, use the command to continue graphical constructions **hold**. It is used in the following forms:

- **hold on**—provides continuation of displaying charts in the current window, which allows you to add subsequent charts to existing ones;

- **hold off** — cancels the graphic plot continuation mode;

It happens that in one window it is necessary to place several coordinate axes with different graphs without overlapping them. For this, the **subplot** commands used before plotting are used:

– **subplot(m, n, p)**—divides the graphics window into $m \times n$ subwindows, where m is the number of subwindows horizontally, n —the number of subwindows vertically, and p —the number of the subwindow into which the current chart will be displayed (subwindows are counted line by line).

Let's illustrate the work of the subplot function (see Fig. 6):

```
>> subplot(3, 2, 1); plot (x,y);
>> subplot(3, 2, 4); plot (x,y);
>> subplot(3, 2, 5); plot (x,y);
```

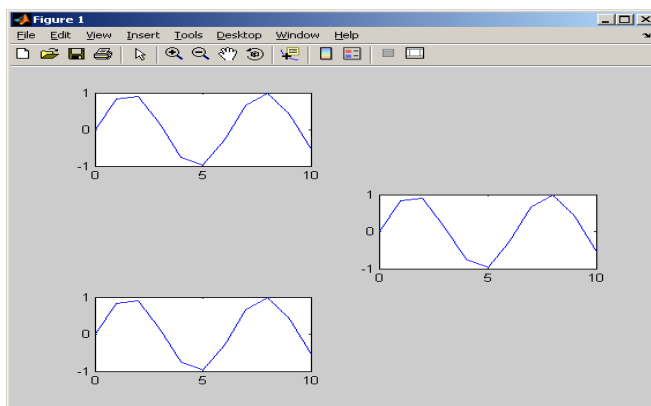


Figure 6 - Function operation

3 rows and two columns of fields for displaying graphs were formed. Each specific field is accessed with its number. Numbering is from left to right and from bottom to top.

Order of execution

1. Compilation and debugging of a program for displaying graphs of functions f_1, f_2, f_3 based on the task from table 5 . The output of graphs must be carried out in one window, the graphs must be signed, scaled.

Table 5

Task variants

Variant №	f_1	f_2	f_3	f_4
1	2	3	4	5
1	$\sin(x)$	$\cos(x)$	x^2	$\cos(r)/r$
2	e^x	x^2	x	$\cos^2(r)/r$
3	$\sin(x) + \cos(x)$	$\cos(x) + x^2$	$x^2 + \lg(x)$	$\cos(r^2)/r$
4	$\sin(x) + e^x$	$\sin(x) + x^2$	$\sin(x) + x$	$\cos(r)/r^2$

5	$x * \sin(x)$	$x * \cos(x)$	x^2	$(\cos(r)/r)^2$
6	xe^x	$\sin(x) + x^2$	$\sin(x) + x$	$\sin^2(r)/r$
7	$\sin(x) * \cos(x)$	$\cos(x) * x^2$	$x^2 \lg(x)$	$\sin(r^2)/r$
8	$\sin(x)e^x$	$\sin(x) * x^2$	$\sin(x) * x$	$\sin(r)/r^2$
9	$\sin^2(x)$	$\cos^2(x)$	x	$(\sin(r)/r)^2$
10	$\sin(x) * e^x$	$\sin(x) * x^2$	$\sin(x) * x$	$r + \cos(r)/r$
11	$\sin^2(x) + \cos^2(x)$	$\cos(x) + x^2$	$x^2 + \lg(x)$	$r + \cos^2(r)/r$
12	$\sin(x) + e^x$	$\sin^2(x) + x^2$	$\sin^2(x) + x$	$r + \cos(r^2)/r$
13	$x * \sin(x)$	$\sin(x) + x^2$	$\sin(x) + x$	$r + \cos(r)/r^2$
14	$\sin(x) + \cos(x)$	$\cos(x) * x^2$	$x^2 \lg(x)$	$r + (\cos(r)/r)^2$
15	$\sin^2(x)$	$\sin(x) + x^2$	$\sin(x) * x$	$\sin^2(r)/r$

Practical lesson - №4

Using the programming mode in the MatLab system to solve problems in mechanics.

The purpose of the work: Learning programming in the MATLAB system.

In MATLAB, two types of files are of particular importance - with the extensions .mat and .m. The former are binary files that can store the values of variables, the latter are text files containing external programs, definitions of commands and system functions. It is to them that most of the commands and functions belong, including those set by the user to solve their specific tasks.

A multi-window editor-debugger with an empty m-file editing window can be called with the Edit command from the command line or with the **File > New > M-file** menu command (Figure 1).

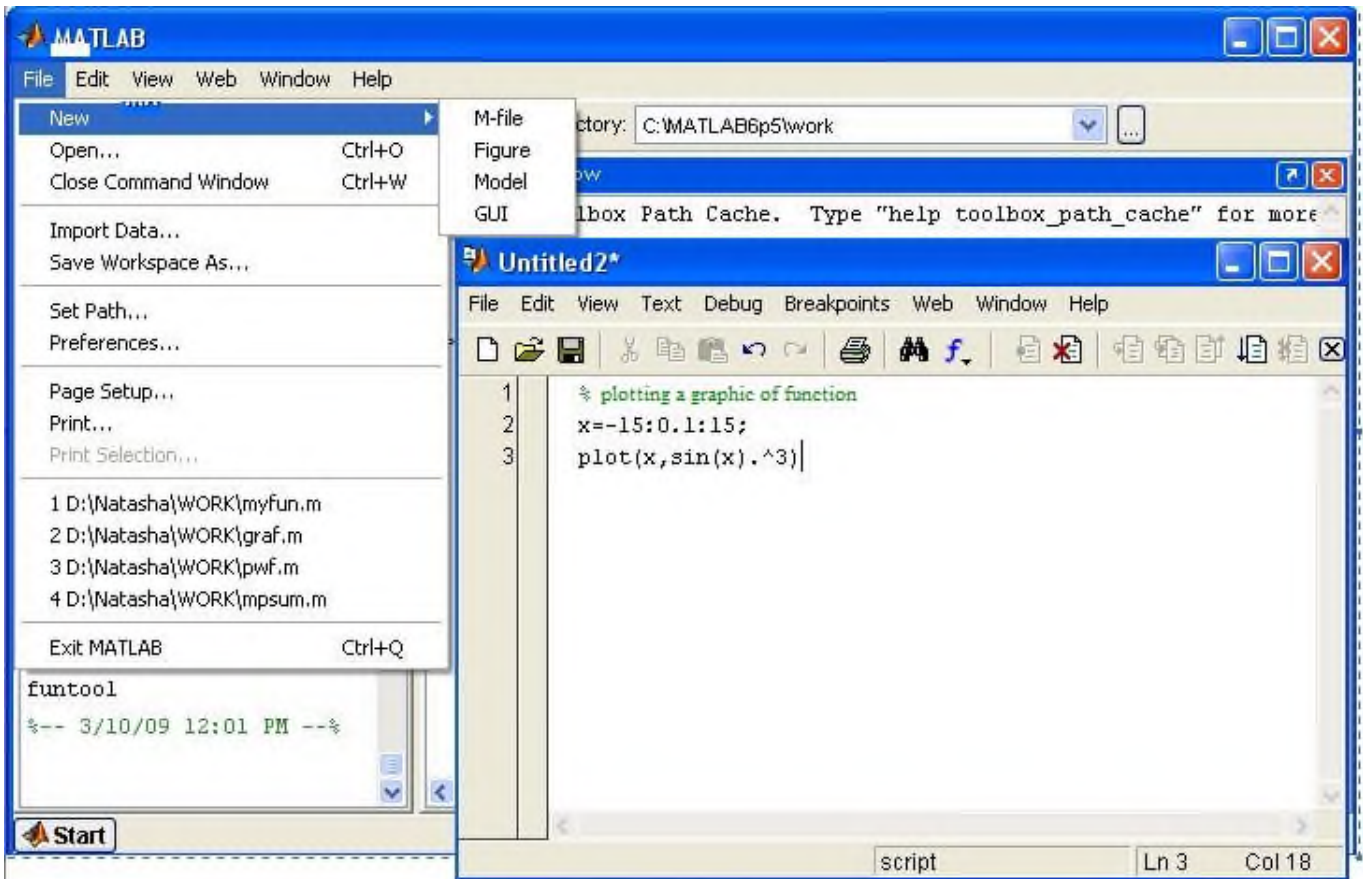


Figure 1 - Multi-window editor-debugger

After that, you can create your own file in the editor window, as well as use the tools for debugging and launching it. To run a file, it must be written to disk using the **Save as** command in the editor's **File** menu. The m-file editor-debugger performs syntactic checking of the program code as you enter text. It uses the following color highlighting:

- programming language keywords — blue color;
- operators, constants and variables — black color;
- comments after the % sign - green color;
- symbolic variables (in apostrophes) — brown color;
- syntax errors - red.

Thanks to color highlighting, the likelihood of syntax errors is drastically reduced.

M-files created by the debugger editor are divided into two classes: script files that do not have input parameters and function files that have input parameters. A **script file**, also called a script file, is simply a record of a series of commands with no input or output parameters. It has the following structure:

%Main Comment

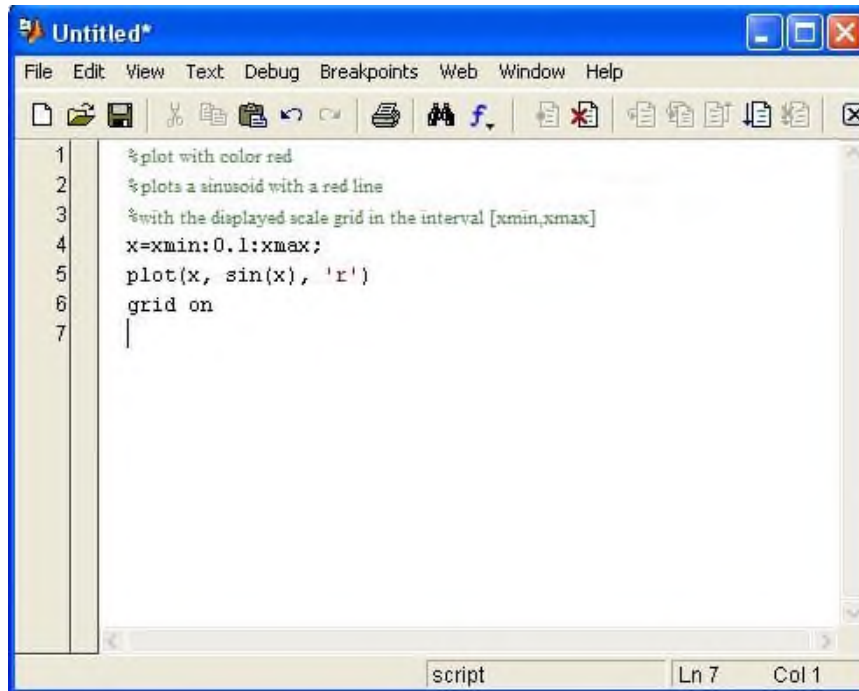
%Additional comment

File body with any expressions

The following properties of script files are important:

- a. they do not have input and output arguments;
- b. work with data from the workspace;
- c. are not compiled during execution;
- d. are a sequence of operations recorded in a file, completely similar to the one used in the session.

Consider the following script file (Figure 2):



```
1 %plot with color red
2 %plots a sinusoid with a red line
3 %with the displayed scale grid in the interval [xmin,xmax]
4 x=xmin:0.1:xmax;
5 plot(x, sin(x), 'r')
6 grid on
7 |
```

Figure 2 - Creating a script file in MATLAB

The first three lines here are the comment, the rest are the body of the file. Pay attention to the possibility of setting a comment in Russian. The % sign in comments must start at the first position of the line. It should be noted that such a file cannot be launched without preliminary preparation, which boils down to setting the values of the **xmin** and **xmax** variables used in the file body. This is a consequence of the first property of script files - they work with data from the workspace. Script file names cannot be used as function parameters because script files do not return values. We can say that a script file is the simplest program in the MATLAB programming language.

The M-file-function is a typical object of the programming language of the MATLAB system. At the same time, it is a full-fledged module from the point of view of structured programming, since it contains input and output parameters and uses the apparatus of local variables. The structure of such a module with one output parameter is as follows:

```
function var = f_name(List_parameters)
```

```
%Main Comment
```

```
%Additional comment
```

```
File body with any expressions
```

var=expression

The M-file function has the following properties:

- 1) it starts with a function declaration, after which the name of the variable var is indicated - the output parameter, the name of the function itself f_name and the list of its input parameters;
- 2) the function returns its value and can be used in mathematical expressions;
- 3) all variables present in the body of the function file are local, i.e., they act only within the body of the function;
- 4) a function file is an independent program module that communicates with other modules through its input and output parameters;
- 5) the rules for outputting comments are the same as for script files;
- 6) when a function file is found, it is compiled and then executed, and the generated machine codes are stored in the workspace of the MATLAB system.

The last construction var = expression is introduced if you want the function to return the result of a calculation. The above form of the function file is typical for a function with one output parameter. If there are more output parameters, then they are indicated in square brackets after the word function. The structure of the module looks like this:

```
function [var1,var2....] = f_name(ParameterList)
```

```
%Main Comment
```

```
%Additional comment
```

```
File body with any expressions
```

```
var1=expression
```

```
var2=expression
```

If a function is used as having a single output parameter, but has multiple output parameters, then the first one will be used to return the value. This often leads to errors in mathematical calculations. Therefore, as noted, this function is used as a separate element of programs of the form [var1, var2] = f_name (List_of_parameters). After its application, the output variables var1, var2 become defined and can be used in subsequent mathematical expressions and other program segments.

To organize dialog input and output, the following operators are used, presented in Table 1.

Table 1

Dialog I/O Operators

Operator	Syntax	Appointment
INPUT	$x = \text{input}(\text{'<invitation >'})$	To enter data from the keyboard
DISP	disp (<variable or text in apostrophes >)	To display

Let us give a simple example of a dialog program that serves to repeatedly calculate the circumference of a circle by a user-entered value of the radius r (Fig.3).

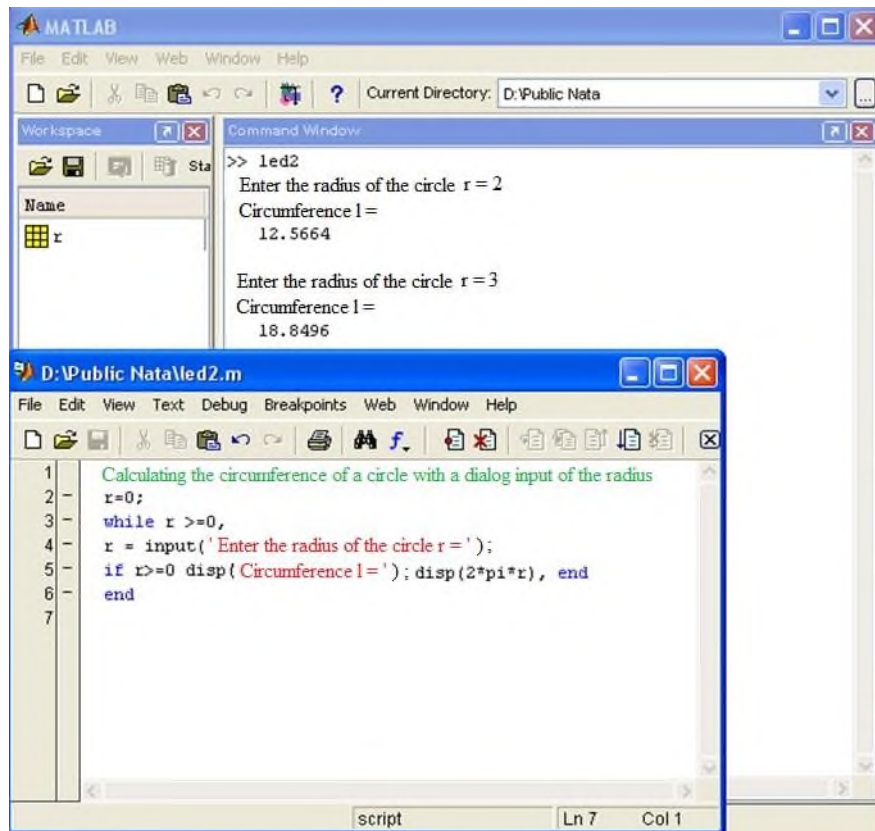


Figure 3 - An example of a dialogue program

Conditional statements are used to organize branches.

Conditional constructs:

1)

if <condition>

<operators>

end

Operators (expression body) are executed only if the condition is true, if the condition is false, then the expression body is not executed.

2)

if <condition>

<statements 1>

else

<statements 2>

end

If the program flow must change depending on several conditions, then the full **if-elseif-else** construct should be used. Each of the **elseif** branches in this case must contain a

condition for executing the block of statements placed after it. It is important to understand that the conditions are checked in a row, the first condition that is met leads to the operation of the corresponding block, exiting **the if-elseif-else** construct and moving to the operator following the **end**. The last else branch should not have any condition. The statements between **else** and **end** work if all conditions are not met. For example, it is required to write a function file for calculating a piecewise given function:

$$f(x) = \begin{cases} 1 - e^{-1-x}, & x < -1; \\ x^2 - x - 2, & -1 \leq x \leq 2; \\ 2 - x, & x > 2. \end{cases}$$

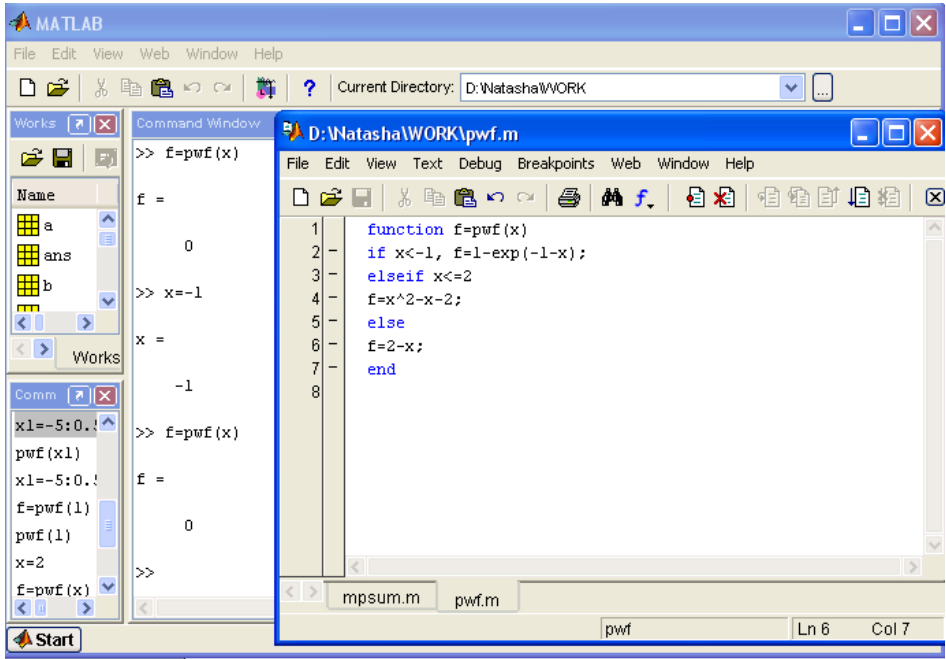


Figure 4 - Listing of the program for calculating the value of the function

The MATLAB system can use the following comparison operators, shown in Table 2.

Table 2

Comparison Operators

Symbol	Appointment	Name of function
<	Less	lt
>=	More or equal	ge
>	More	gt
<=	Less than or equal to	le
==	Equals	eq
~=	Not equal	ne

The operations (`==`, `~=`) compare the real and imaginary parts of complex numbers, and the operations (`>`, `<`, `>=`, `<=`) compare only the real parts.

Logical operations can be written as functions (Table 3).

Boolean Operations

Symbol	Appointment	Name of function
&	Logical "and"	and
	Logical "or"	or
~	Negation	not

The result of logical operations are the numbers 0 (false) and 1 (true).

There are two kinds of loop statements in MATLAB - conditional and arithmetic. To repeat statements an unfixed number of times, use the loop statement with a precondition:

while <condition>

<operators>

end

The statements are executed if the variable <condition> has non-zero elements.

The arithmetic loop operator has the following form:

for <name> = <IV>: <Step>: <FV>

<operators>

end,

where <name> is the name of the cycle control variable,

<IV> – initial value of the control variable,

<FV> – the final value of the control variable,

<Step> – increment of the values of the variable <name> in the course of its change from the value <IV> to the value <FV>. If the <Step> parameter is not specified, its default value is one.

When working with a for loop, it is permissible to use the break loop operator. When this statement is executed, the loop is terminated, and control is transferred to the next statement after the end of the loop.

The progress of the program can be determined by the value of some variable (switch). Such an alternative method of program branching is based on the use of the **switch** statement. The **switch** statement contains blocks beginning with the word **case**, after each **case**, the value of the switch at which this block is executed is written separated by a space. The last block begins with the word **otherwise**, and its statements work when none of the case blocks has been executed. If at least one of the **case** blocks is executed, then the **switch** statement exits and goes to the statement following the end.

Let's assume that you want to find the number of ones and minus ones in a given array and, in addition, find the sum of all elements that are different from one and minus one. The program listing contains a function file that, given an array, returns the number minus ones

in the first output argument, the number of ones in the second, and the sum in the third (Fig.5).

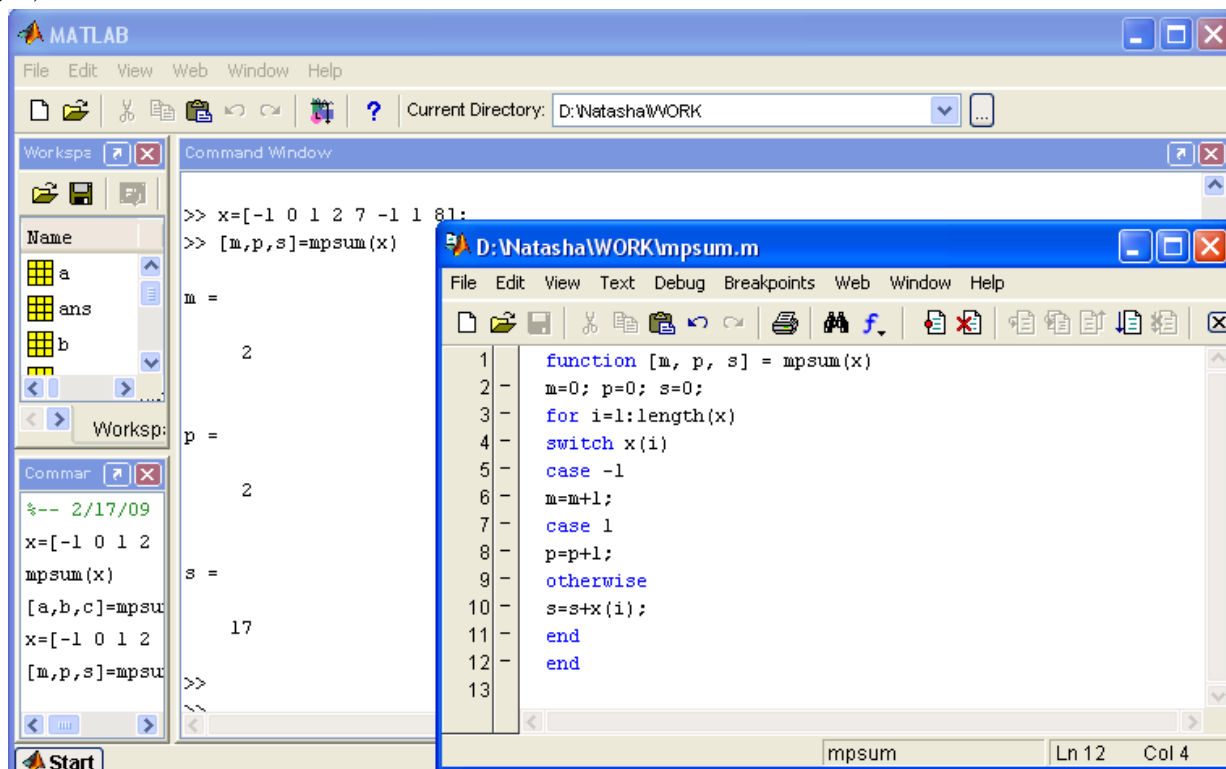


Figure 5 - Listing of the program

The pause statement is used to stop the program. It is used in the following forms:

- a) `pause` - stops calculations until any key is pressed;
- b) `pause(N)` - stops calculations for N seconds;
- c) `pause on` - turns on the pause processing mode;
- d) `pause off` - turns off the pause processing mode.

Order of execution

1. From the script file, use the dialog input function to enter all the necessary data from the keyboard. Perform the calculation using conditional operators and display the results in the command window (Table 4).

Table 4

Task variants

Variant №	Task
1	2
1	Find the sum of the positive of the four given variables.
2	Find the maximum value of the four given variables and output it.
3	Four variables are set. Replace the smallest of them with the sum of the rest.
4	Four variables are set. Count the number of negative ones and the number of zero ones.
5	Find the product of the negatives of four given variables.

6	Two figures are given: the square is given by the length of the side, and the circle is given by the length of the radius. Determine which of them has a large area and how much.
7	Four variables are set. Replace all negative ones with absolute values and increase by 2 times.
8	Given four variables, count the number of zeros, positive and negative.
9	Four variables are given. Find among them the variables closest in value to .
10	Four variables are set. Replace all positive ones with negative values multiplied by 5.
11	Find the minimum and maximum values of the four given variables.
12	Four values are given. Determine which of them are integers.
13	Four variables are set. Calculate the number and product of values in the interval [1 5].
14	Four variables are set. Replace all positive ones with negative values multiplied by 5.
15	Four variables are set. Count the number of negative ones and the number of zero ones.

2. Write a function file using branch and loop operators, based on the task options presented in Table 5.

Table 5

Task variants

№	Input array	Formed array	Task
1	2	3	4
1	$A_{3 \times 3}$	$B_{3 \times 3}, b_{ij} = \begin{cases} a_{ij}, i < j \\ a_{ji}^2, i \geq j \end{cases}$	Form an array A1 from the minimum elements of the rows of the matrix A and an array B1 from the minimum elements of the rows of the matrix B. Among the elements of A1 and B1 find the maximum
2	A_3	$B_3, b_i = \sin(i^2), i = 1 \dots 3$	Form an array C - the sum of the elements of arrays A and B. Find the maximum value of arrays A, B, C.
3	$A_{3 \times 3}$	$B_{3 \times 3}, b_{ij} = \sin(i) * \sin(j)$ $i = 1 \dots 3, j = 1 \dots 3.$	Determine the minimum elements in the matrices A and B (mA and mB). Calculate $C = A * B * mA * mB.$

4	$A_{3 \times 3}$	$B_3, b_i = \log(2i + \cos(i)),$ $i = 1 \dots 3.$	Form an array A1 from the average values of the elements of the rows of matrix A.
5	$A_{3 \times 3}$	$B_3, b_i = \sin(i) + \cos(i)$ $i = 1 \dots 3.$	Determine the maximum elements in matrix A and array B (mA and mB). Calculate $C = A * B * mA * mB.$
6	$A_{3 \times 3}$	$B_3, b_i = \log(2i + \cos(i)),$ $i = 1 \dots 3.$	Form an array A1 from the average values of the elements of the rows of matrix A.
7	$A_{3 \times 3}$	$B_3, b_i = \sin(\ln(i) + \cos(i)),$ $i = 1 \dots 3.$	Sort array A1 in ascending order and B in descending order. Perform element-wise multiplication of A1 and B.
8	$A_{3 \times 3}$	$B_3,$ $b_i = i * \sin(i) + j * \cos(i)$ $i = 1 \dots 3, j = 1 \dots 3.$	Calculate the product of the elements of matrix A (pA) and the sum of the elements of matrix B (cB). Calculate the matrix $C = pA * cB * A * B'.$
9	A_3	$B_3, b_i = i * \log(i^2) + \sin(i),$ $i = 1 \dots 3.$	Sort array A in ascending order and replace the last row of matrix B with it.
10	$A_{3 \times 3}$	$B_3, b_i = i * \sin(j) * \log(i)$ $i = 1 \dots 3$	Sort arrays A and B in ascending order. Perform element-wise division of ordered arrays. Determine the product of the elements of the resulting array.
11	$A_{3 \times 3}$	$B_3,$ $b_i = i * \sin(i) + j * \cos(i)$ $i = 1 \dots 3, j = 1 \dots 3.$	Calculate the product of the elements of matrix A (pA) and the sum of the elements of matrix B (cB). Calculate the matrix $C = pA * cB * A * B'.$
12	$A_{3 \times 3}$	$B_3,$ $b_i = \sin(2i) + \cos(3i)$ $i = 1 \dots 3.$	Determine the minimum elements in the matrices A and B (mA and mB). Calculate $C = A * B * mA * mB.$
13	$A_{3 \times 3}$	$B_3,$ $b_i = \sin(\ln(i) + i * \cos(i))$ $i = 1 \dots 3.$	Form an array A1 from the maximum elements of the rows of the matrix A. Perform element-wise multiplication $A1 * B.$ Sort array A1 in ascending order.

14	$A_{3 \times 3}$	$B_3, b_i = \sin(\ln(i) + \cos(i)),$ $i = 1 \dots 3.$	Sort array A1 in ascending order and B in descending order. Perform element-wise multiplication of A1 and B.
15	A_3	$B_3,$ $b_i = \cos(i^2), i = 1 \dots 3.$	Form an array C - the product of elements of arrays A and B. Find the maximum and minimum values of arrays A, B, C.

Content of the report

1. The purpose of the lesson.
2. Listing of programs and results of program execution.

Control questions

1. How is dialog input and output carried out?
2. What are conditional statements used for?
3. What is the difference between script files and function files?

Practical lesson - №5

Cryptographic methods of information protection.

The purpose of the lesson: to study the methods of cryptographic protection, to get acquainted with cryptographic algorithms.

Theoretical part

An information message ready for transmission, initially open and unprotected, is encrypted and thereby converted into a ciphergram, i.e. into closed text or a graphic image of a document. In this form, the message is transmitted over a communication channel, even if it is not secure. The authorized user, after receiving the message, decrypts it (i.e., reveals it) by inverse transformation of the cryptogram, as a result of which the original, open form of the message is obtained, accessible to the perception of authorized users. Modern cryptography knows two types of cryptographic algorithms: classical algorithms based on the use of private, secret keys, and new public key algorithms that use one public and one private key (these algorithms are also called asymmetric). In addition, it is possible to encrypt information in a simpler way - using a pseudo-random number generator. The use of a pseudo-random number generator consists in generating a cipher gamma using a pseudo-random number generator with a certain key and applying the resulting gamma to the open data in a reversible way.

The reliability of encryption using a pseudo-random number generator depends both on the characteristics of the generator and, to a greater extent, on the gamma generation algorithm.

This method of cryptographic protection is implemented quite easily and provides a fairly high encryption speed, however, it is not sufficiently resistant to decryption and therefore is not applicable to such serious information systems as, for example, banking systems.

Classical cryptography is characterized by the use of one secret unit - the key, which allows the sender to encrypt the message, and the recipient to decrypt it. In the case of encrypting data stored on magnetic or other storage media, the key allows you to encrypt information when writing to the media and decrypt when reading from it. When encrypting the source text using this method, each letter is replaced by another letter of the same alphabet by shifting it in the used alphabet by a number of positions equal to K . When the end of the alphabet is reached, a cyclic transition to its beginning is performed.

Caesar's cipher

The general formula for a Caesar cipher is:

$$C = P + K \pmod{M},$$

where P is the number of the plaintext character, C is the corresponding number of the ciphertext character, K is the encryption key (shift factor), M is the size of the alphabet (for the Russian language $M = 32$)

For a given substitution cipher, a fixed substitution table can be specified containing the corresponding pairs of plaintext and ciphertext letters.

Example:

The table of substitutions for characters of the English text with the key $K=3$ is presented in table.1. This table corresponds to the formula:

$$C = P + K \pmod{M}, \quad (2)$$

Table 1.

Substitutions of the Caesar cipher for the key $K=3$

A		D		N		Q
B		E		O		R
C		F		P		S
D		G		Q		T
E		H		R		U
F		I		S		V
G		J		T		W
H		K		U		X
I		L		V		Y
J		M		W		Z
K		N		X		A
L		O		Y		B
M		P		Z		C

According to formula (2), the plaintext "LUGGAGE" will be converted into the ciphertext "OXJJDJH".

The decryption of a private text encrypted by the Caesar method according to (5.1) is carried out according to the formula:

$$P = C - K \pmod{M} \quad (3)$$

Simple monoalphabetic substitution

The simple monoalphabetic substitution cipher is a generalization of the Caesar cipher and performs encryption according to the following scheme:

$$C = a * P + K \pmod{M} \quad (4)$$

where $0 \leq a, K < M$ is the encryption key, P is the place of the character in the alphabet $\text{GCD}(a, M) = 1$.

A transformation according to scheme (1) is a one-to-one mapping only if a and M are coprime. In this case, to decrypt the private text, the inverse transformation is performed according to the formula:

$$P = a^{-1} \cdot (C - K) \pmod{M} \quad (5)$$

Example 4.2.

Let $M=26$, $a=3$, $K=6$, $\text{GCD}(3, 26) = 1$. Then we obtain the following substitution table for the cipher of a simple monoalphabetic substitution.

Simple permutation method

When encrypting by the simple permutation method, the plain text is divided into blocks of the same length, equal to the length of the key. The key of length n is a sequence of non-repeating numbers from 1 to n. The plaintext characters within each of the blocks are rearranged to match the key characters within the block from right to left. The key element K_i at a given position in the block indicates that the plaintext character numbered K_i from the corresponding block will be placed in that position.

Example:

Let's encrypt the plaintext "COMING DAY" by the permutation method with the key $K=3142$.

C O M I N G D A Y
C M O I N D G A Y

Hamilton's algorithm

A very high encryption strength can be achieved by complicating permutations along Hamiltonian-type routes. At the same time, the vertices of some hypercube are used to write the characters of the ciphertext, and the characters of the ciphertext are read along Hamiltonian routes, and eight different routes are used. The size of the permutation key in this case is equal to eight by the number of cube vertices. For example, two of Hamilton's

routes are shown in Fig. 2. The first route corresponds to the permutation 4-0-2-3-1-5-7-6, the second 4-6-2-0-1-5-7-3 (characters in the block are numbered from zero).

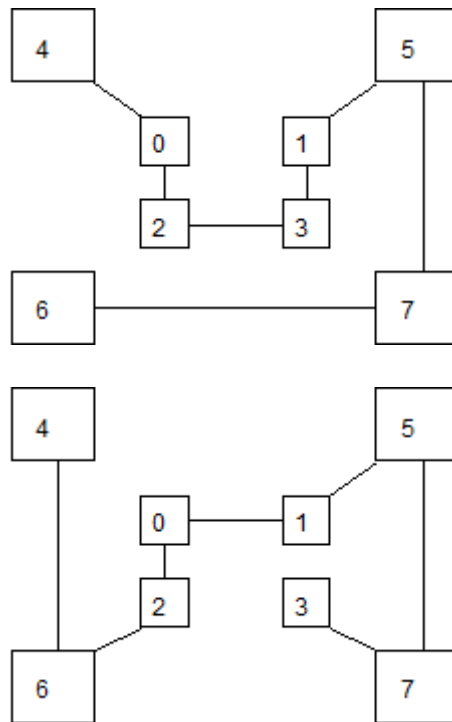


Fig.2. Example of Hamilton routes

Control questions:

- 1 Describe the direction of "cryptography". What is a cryptographic key?
2. Classify traditional encryption algorithms. Briefly describe these classes.
3. Describe Caesar encryption methods, simple monoalphabetic substitution, simple permutation, Hamilton permutations.

Practical lesson - №6

Mathematical modeling of the problem. Algorithmization of tasks. Using Borland C++ Builder 6 constructs to solve engineering problems. Working in C++ builder6 integrated environment. Programming technical tasks in the C++ builder environment.

The purpose of the lesson: To study the basic properties, structures of various algorithms and the principles of their description when preparing problems for solving on a computer.

Theoretical part

An algorithm is a set of certain rules, precise actions and instructions, the implementation of which in a given sequence leads to the solution of the task. Ways to describe the algorithm:

- 1) verbal description;
- 2) graphic description;

3) description of the algorithm in the programming language.

With the **verbal** method, the algorithm is given in an arbitrary presentation in natural language. The disadvantage of this method is that the algorithm is not strictly formalizable, verbose, and ambiguous. However, this way of presenting the algorithm does not require special knowledge and can be used by end users. It is in this language, as a rule, that the informal statement of the problem is reported at the formalization stage, and it can also be used to present the result of the first stage.

The **language of graphic symbols** involves the correlation of each type of action with a geometric figure, represented as a block symbol. Actions (blocks) are connected by flow lines. A collection of such related blocks is called a block diagram.

The algorithm has the following properties (they follow from the definition):

1) **certainty (determinacy)** - each command (or prescription) is understandable to the performer (human or computer) and eliminates the ambiguity of execution;

2) **effectiveness** - the implementation of the computational process provided by the algorithm must, after a certain number of steps, lead to a result or a message about the impossibility of obtaining it;

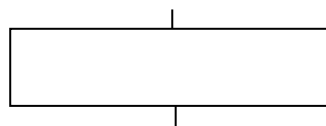
3) **mass character** - if the algorithm is designed to solve a specific problem, it must be applicable to solve problems of this type for all valid values of the initial data;

4) **discreteness** - the step-by-step nature of the process of obtaining the result, consisting in the sequential execution of a finite number of actions specified by the algorithm

Algorithm types: *linear algorithm, branching algorithm, cyclic algorithm.*

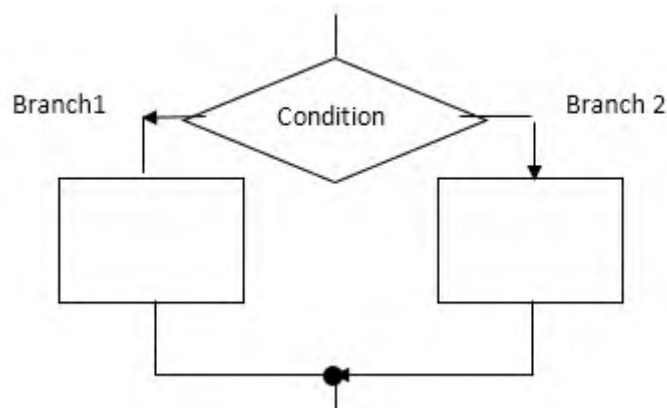
Linear algorithm - a sequence of actions is carried out sequentially in strict order. It is described by the language of graphic symbols as follows:

Arithmetic block:

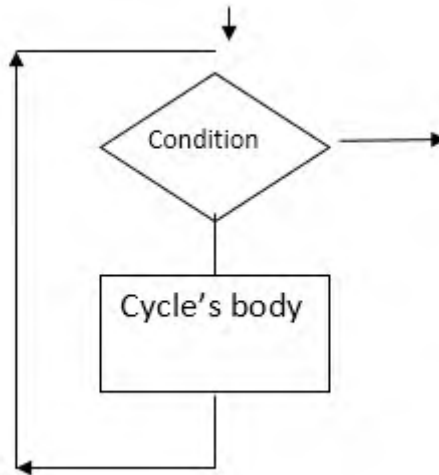


A **branching algorithm** is a problem in which the choice of a solution to the problem is carried out depending on the given conditions. It is described by the language of graphic symbols as follows:

Logic block



Cyclic algorithm - a sequence of steps is repeated many times with a change in some parameter. Described by the graphic symbol language in the following way



Exercise 1. Find h , which reaches the body thrown up. It is known from the course of physics that a body thrown vertically upwards with a speed V_0 reaches a height calculated by the formula:

$$H = V_0 T - gT^2 / 2$$

Therefore, the algorithm will be simple and linear, consisting of arithmetic blocks executed in strict sequence one after another.

Recording the algorithm in the form of a block diagram (Fig.1):

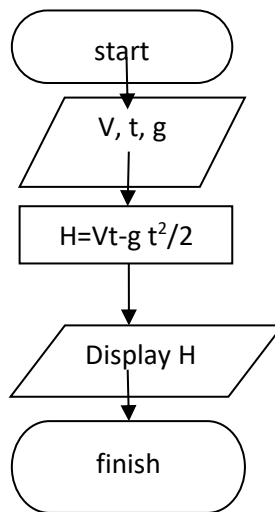


Fig.1. Block diagram of the algorithm of exercise 1.

Exercise 2. Determine if it is possible to build a triangle whose vertices are points on the plane with coordinates $x_1, y_1, x_2, y_2, x_3, y_3$

The algorithm for solving this problem can be represented as follows (Fig. 2), where R_1, R_2, R_3 are the distances between points, K -condition (logical expression for constructing a triangle),

$$k = (r_1 + r_2) > r_3 \ \& \ (r_2 + r_3) > r_1 \ \& \ (r_1 + r_3) > r_2$$

Recording the algorithm in the form of a block diagram (Fig. 2):

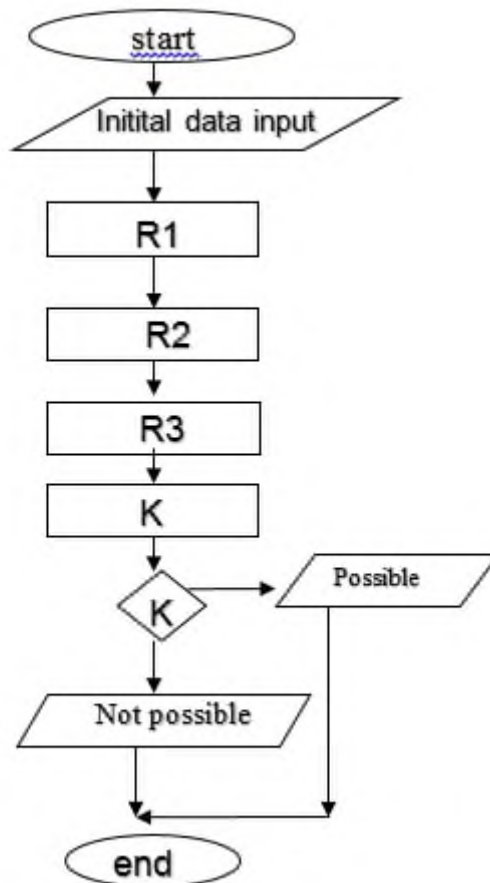


Fig.2. Block diagram of the algorithm of exercise 2.

Exercise 3. Calculate $y=(y*x)^n$ if the degree of n is given.

Recording the algorithm in the form of a block diagram (Fig. 3):

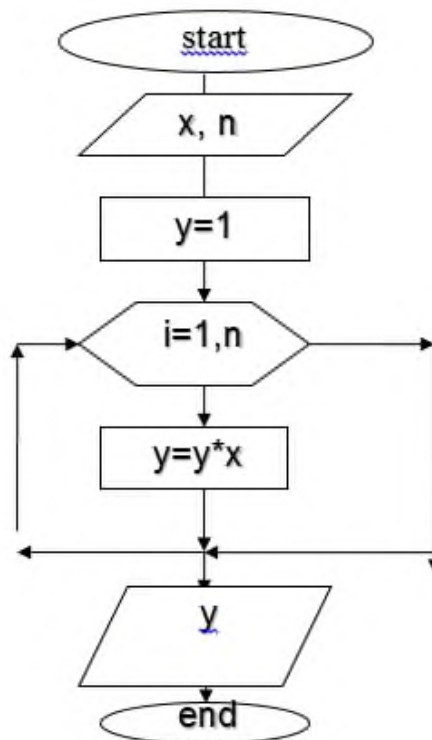


Fig.4. Block diagram of the algorithm of exercise 4.

Exercise 5. Determine the arithmetic mean of two numbers if **a** is positive and the quotient (a/b) otherwise.

Recording the algorithm in the form of a block diagram (Fig. 5):

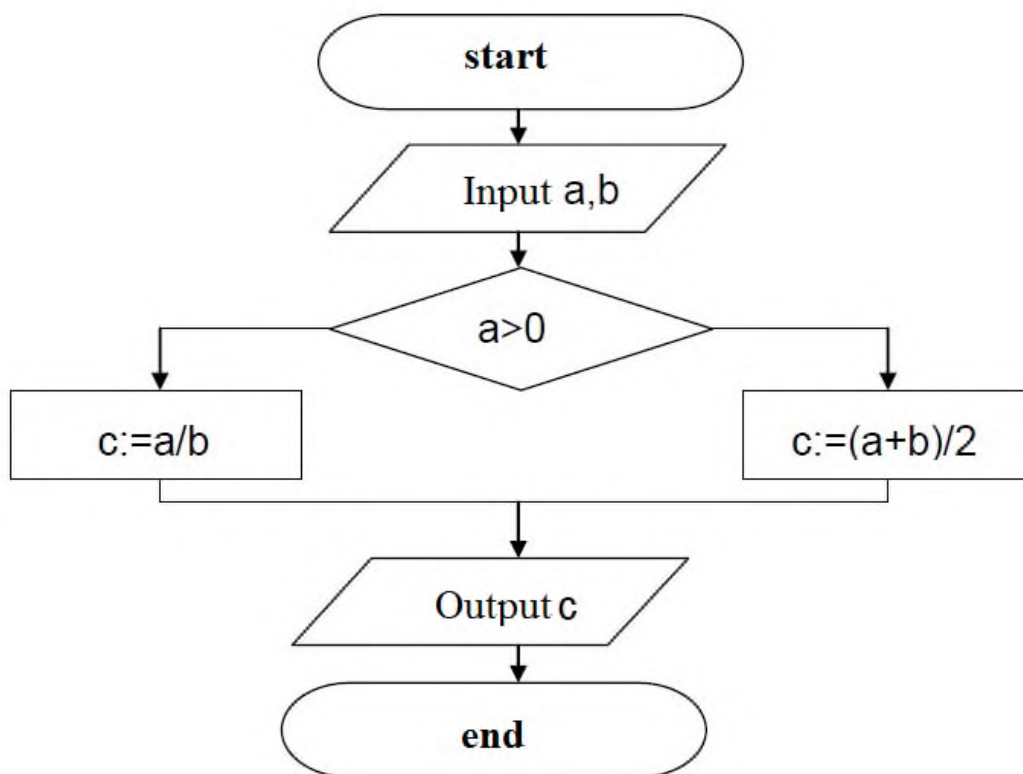


Fig.5 Block diagram of the algorithm of exercise 5.

Exercise 6. Write an algorithm for finding the sum of integers in the range from 1 to 10. Recording the algorithm in the form of a block diagram (Fig. 6):

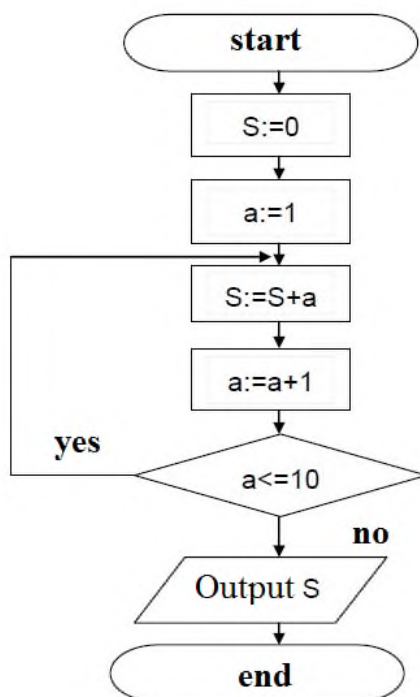


Fig.6 Block diagram of the algorithm of exercise 6.

Task variants

Task 1. Create an algorithm for solving the problem using block diagrams using the construction of a linear algorithm.

1. Calculate the surface area and volume of a truncated cone using the following formulas:

$$S = \pi (R + r) l + \pi R^2 + \pi r^2 ;$$

$$V = (1/3) \pi (R^2 + r^2 + Rr) h .$$

2. Calculate the coordinates of the center of gravity of three material points with masses m_1 , m_2 , m_3 and coordinates (x_1, y_1) , (x_2, y_2) , (x_3, y_3) using the formulas:

$$x_c = (m_1 x_1 + m_2 x_2 + m_3 x_3) / (m_1 + m_2 + m_3) ;$$

$$y_c = (m_1 y_1 + m_2 y_2 + m_3 y_3) / (m_1 + m_2 + m_3) .$$

3. Calculate the area of a triangle with sides a , b , c using Heron's formula:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

where p is the semiperimeter calculated by the formula $(a+b+c)/2$

Task 2. Create an algorithm for solving the problem using flowcharts, using the construction of an algorithm with branching.

1. Write a program to solve the quadratic equation $ax^2 + bx + c = 0$.

2. Determine the maximum even number of the two entered.

3. Determine whether it is possible to build a triangle from segments with lengths x , y and z .

Task 3. Create an algorithm for solving the problem using block diagrams using the design of a cyclic algorithm.

1. Find the sum of numbers that are multiples of three, in the range from 0 to 50.

2. Find the sum of the first ten numbers that are multiples of five.

3. Find the product of even numbers in the range from 2 to 30.

Questions for self-control.

1. What is an algorithm and what structures of algorithms do you know?

2. What forms of representation of the algorithm do you know?

3. Name the main blocks, representations of the algorithm in graphical form.

4. What is a linear algorithm?

5. What is a branching algorithm?

6. What types of cyclic algorithms do you know?

Practical lesson - №7

Visual programming. Components used in visual programming. Loop statements in mechanical programming

The purpose of the lesson: To master the skills of programming branching structures of algorithms.

Theoretical part

Loop operators are used to organize repetitive calculations. Any loop consists of a loop body, that is, those statements that are executed several times, initial settings, a block for modifying the loop parameter and checking the exit condition from the loop, which can be placed either before the loop body (then they talk about a loop with a precondition) or after the body loop (loop with postcondition). One pass of the loop is called an iteration. Variables that are forced to change in a loop and are used when checking the exit condition from it are called loop parameters. Integer loop parameters that change by an integer at each iteration are called loop counters. It is impossible to transfer control from the outside to the inside of the loop. Exit from the loop is possible both when the exit condition is met, and by the break, return or unconditional jump statements.

A loop with a precondition (while) looks like this:

```
while ( expression ) statement;  
or while ( expression ) { compound statement };
```

The expression defines the condition for repeating the loop body represented by a simple or compound statement. If the expression is not 0 (true), the loop statement is executed, after which the expression is evaluated again. If the expression is 0 on the first test, the loop will never execute. The type of the expression must be arithmetic or castable.

A loop with a postcondition (do while) looks like this:

```
do statement while expression;  
or do { compound statement } while expression;
```

First, the simple or compound statement that makes up the body of the loop is executed, and then the expression is evaluated. If it is not equal to 0 (true), the body of the loop is executed again, and so on, until the expression is equal to zero or any control transfer statement is executed in the body of the loop. The type of the expression must be arithmetic or castable.

A loop with a (for) parameter has the following format:

```
for (initialization; expression; modifications) statement;
```

Initialization is used to declare and assign initial values to the values used in the loop. In this part, you can write several statements, separated by a comma. The expression defines

the loop execution condition: if it is not equal to 0 (true), the loop is executed. Modifications are performed after each iteration of the loop and usually serve to change the parameters of the loop. In the part of modifications, you can write several operators separated by commas.

A simple or compound statement is the body of the loop. Any part of the for statement can be omitted (but the semicolons must be left in place!). In any part, you can use the "comma" operation (sequential calculation), for example:

```
for (int i = 1, s = 0; i<=100; i++) s += i; // sum of numbers from 1 to 100
```

Exercise 1. Calculate $y = \sum_{i=1}^{50} i^3 + 1$, where i is a natural number.

```
//-----
#include <math.h>
#include <vcl.h>
#include<iostream.h>
#include<conio.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
    float y;
    y=0;
    for (int i=1;i<=50;i++) //
// a variable of integer type i is declared and an initial value is assigned i=1; the expression
i<=50 defines the condition for executing the loop; i++ means that with each iteration of the
loop, the value of the variable is incremented by one.
    y=y+pow(i,3)+1;
    cout<<"y="<<y<<endl;
        getch();    return 0;
}
```

Exercise 2. Calculate $S = 1 + \frac{x}{1} + \frac{x^2}{2} + \dots + \frac{x^n}{n}$

```
//-----
#include <math.h>
#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
```

```

float s,x; int i,n;
cout<<"n="; cin>>n;
cout<<"x="; cin>>x;
s=1;
while (i<=n) {s=s+pow(x,i)/i; i+=i;}
cout<<"s= "<<s<<endl;
getch(); return 0;
}
//-----

```

Exercise 3: Login Program.

```

#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])

string password;
cout << " Vvedite parol: ";
cin >> password;
while (password != " xyzzy" )
{
cout << " Wrong password, try again: " ;
cin >> password;
}

```

Exercise 4: Write a program to display numbers between 0 and 9 using a FOR loop.

```

#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
    {for ( int i = 0; i < 10; i++ )
    {
cout << i << '\n';
}
}

```

Exercise 5. Write a program to display the squares of numbers in the range from 0 to 9 using the FOR loop operator.

```

#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{ // Cikl dlitsa while i < 10, //
for ( int i = 0; i < 10; i++)
{
    cout<< i << " in square " << i * i << endl;
}
    getch(); return 0;
}

```

Exercise 6: Write a password authentication program using **the do while** loop statement

```

#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
    string password;
    do
    {   cout << "Please enter your password: ";   cin >> password;
    } while ( password != "miracle" );
    cout << "Welcome, you got the password right";
}

```

Exercise 7. Write and debug a program to display all the values of the function $S(x)$ for the argument x , which varies in the range from a to b with a step h and a given n .

An example of creating a console application

The text of the program of the proposed task may look like:

```

#include <vcl.h>
#include <stdio.h>
#include <conio.h>
#pragma hdrstop
#pragma argsused
int main(int argc, char* argv[])
{
    double a, b, x, h, r, s;

```

```

int n, zn = -1, k;
puts("Input a,b,h,n");
scanf("%lf%lf%lf%d", &a, &b, &h, &n);
for(x = a; x<=b; x+=h) {
int n, zn = -1, k;
    a = StrToFloat(Edit1->Text);
    b = StrToFloat(Edit2->Text);
    n = StrToInt(Edit3->Text);
    h = StrToFloat(Edit4->Text);
for(x = a; x<=b; x+=h) {
    r = s = 1;
    for(k = 1; k<=n; k++) {
        r = zn*r*x/k;
        s+=r;
    }
    Memo1->Lines->Add("when x= "+FloatToStrF(x,ffFixed,8,2)
        +" sum= "+FloatToStrF(s,ffFixed,8,5));
}
}
//-----
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Memo1->Clear();
}

```

The task

1. Write a program for calculating $s = (1 + \frac{1}{1^2})(1 + \frac{1}{2^2}) + \dots (1 + \frac{1}{n^2})$
2. Write a program to calculate $Y = \sin x + \sin^2 x + \sin^3 x + \dots \sin^n x$
3. Write a program to calculate $F = n!$;

$$Y = \begin{cases} \sum_{i=1}^{50} \frac{x_i}{2^i} & x \geq 0 \\ \prod_{i=1}^{50} \frac{i^2 x^2}{i+2} & x < 0 \end{cases}$$

4. Write a program to calculate

5. Write a program to calculate the sum of the serial numbers of lowercase letters from 'A' to 'Z'.
6. Print out all odd natural numbers from 1 to 100.
7. Find a natural number from the interval from a to b, which has the maximum number of divisors
8. Write a program for finding the digital root. The digital root of a given number is obtained by adding all the digits of this number, then all the digits of the found sum and repeating this process until the result is a single digit.

Questions for self-control

1. What loop statements are used in C++ programs?
2. When is a loop statement with a parameter used?
3. What types are used for the loop parameter in the For..
4. Peculiarities of cycles with precondition and postcondition.
5. What is a loop within a loop?

Practice lesson - №8

Using classes and methods to work with graphical objects in programming.

The purpose of the work: to master the technique of creating graphic images using the Borland C ++ Builder6 tools and to study some of the possibilities of plotting functions using the Chart and Image components; learn how to work with graphic objects; write and debug a program using graphic information display functions.

Tasks:

1. Study the theoretical part.
2. Write a program for constructing a graphic image.
3. Implement the program in visual mode.
4. Prepare a progress report.

As in any programming system, Borland C++ Builder6 has the ability to work with graphics. To work with graphics, there are two special classes TGraphic and TPicture. The TGraphic class provides the creation of 3 types of files: icons, metafiles and bitmaps. There are special classes TFont styles and pen sizes, TPen pen, pen styles, TBrush brush styles.

Theoretical part

The surface on which the program can display graphics corresponds to the Canvas property. In turn, the Canvas property is an object of type TCanvas. Methods of this type provide the output of graphic primitives (points, lines, circles, rectangles, etc.), and properties allow you to set the characteristics of the displayed graphic primitives: color, thickness and style of lines; color and type of filling areas; font characteristics when displaying text information.

The rendering methods of drawing primitives consider the Canvas property as some kind of abstract canvas on which they can draw (canvas is translated as "surface", "canvas for drawing"). The canvas is made up of individual dots - pixels. The position of a pixel is characterized by its horizontal (X) and vertical (Y) coordinates. The top left pixel has coordinates (0, 0). Coordinates increase from top to bottom and from left to right. The coordinate values of the lower right point of the canvas depend on the size of the canvas

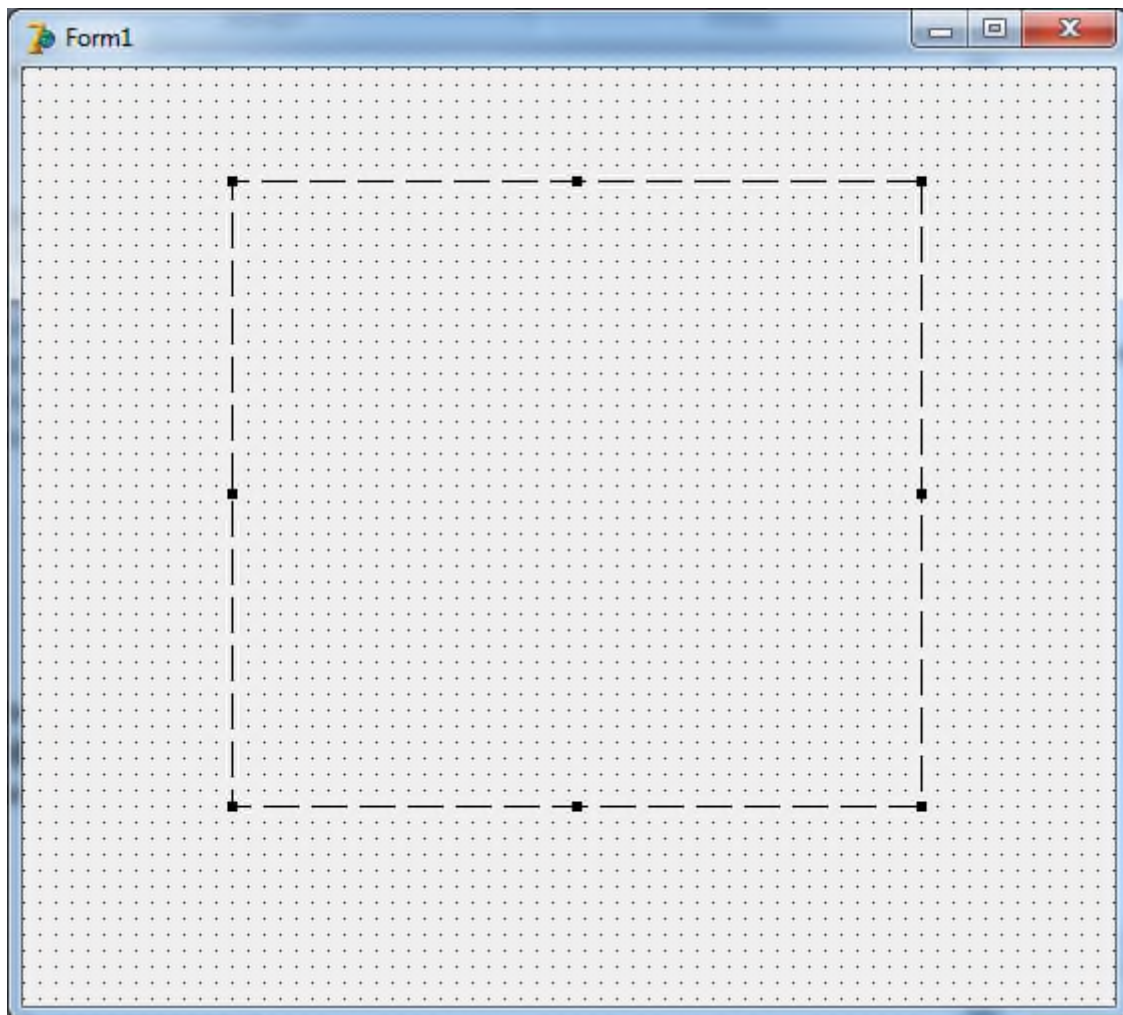


Fig.1. Canvas position for graphics.

The Color property specifies the color of a line drawn with a pencil.

Exercise 1. Using the Paint Box visual component, build a multi-colored square (Fig. 2).

Program code in visual mode:

```

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
}
//-----
int i,x,y;
void __fastcall TForm1::Button1Click(TObject *Sender)
{
randomize;
for (i=1;i<=300;i++)
{ x=random(250);
y=random(250);
}
}

```



```

PaintBox1->Canvas->Pixels[x][y]=RGB(255,0,0); }
}
//-----

```

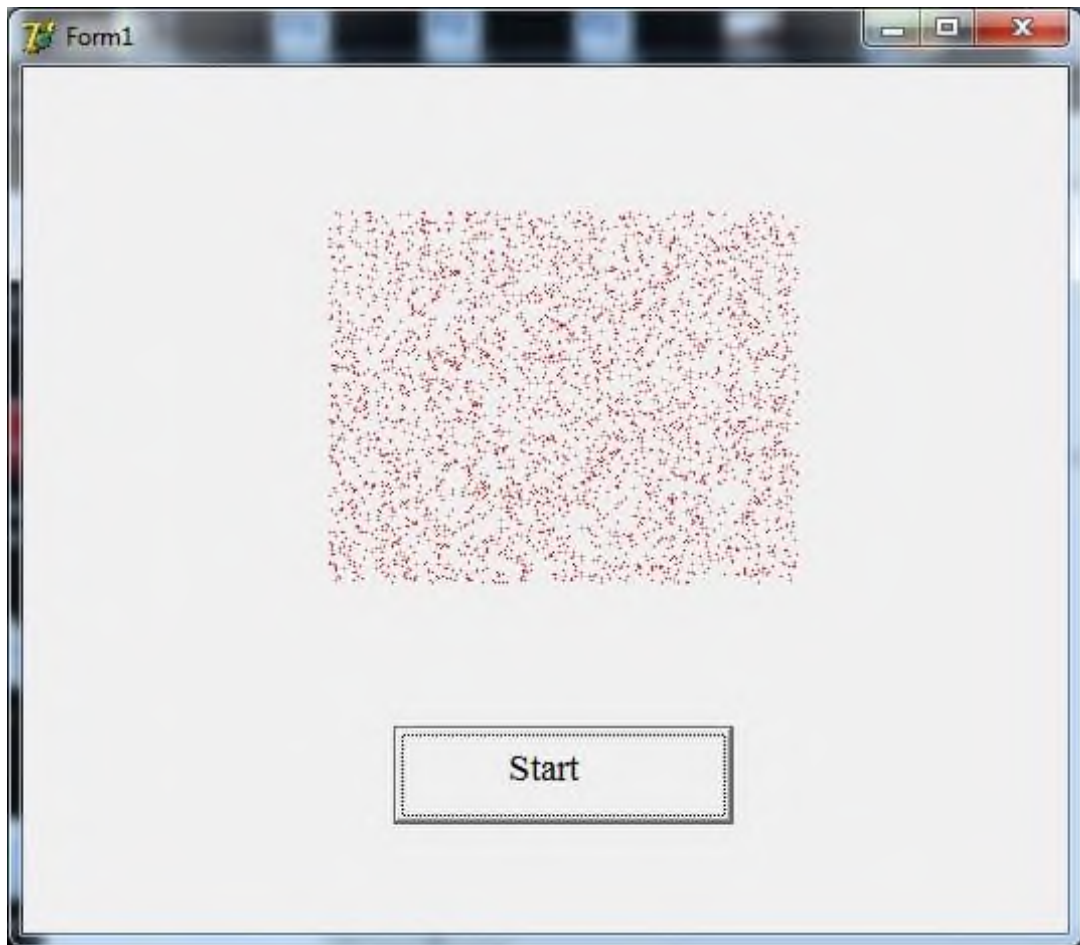


Fig.2 Received image

Exercise 2: Fill the Canvas with Random Colored Lines of Different Thickness

Program code in visual mode:

```

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
int i,x,y;
void __fastcall TForm1::Button1Click(TObject *Sender)

```

```

{
PaintBox1->Canvas->Pen->Color=RGB(random(255),random(255),random(255));
PaintBox1->Canvas->Pen->Width=random(3)+1;
x=random(150);
y=random(150);
PaintBox1->Canvas->MoveTo(x,y);
x=random(150);
y=random(150);
PaintBox1->Canvas->LineTo(x,y);
}

```



Fig.3 Received image

Exercise 3. Fill the canvas with ellipses randomly (Fig. 4).

```

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
//-----

int i,x,y;
void __fastcall TForm1::Button1Click(TObject *Sender)
{int i,x,y,x2,y2;

```

```

for (i=1;i<=300;i++)
{
PaintBox1->Canvas->Pen->Color=RGB(random(255),random(255),
random(255));
PaintBox1->Canvas->Pen->Width=random(3)+1;
PaintBox1->Canvas->Brush->Color=RGB(random(255),random(255),
random(255));
x=random(150);
y=random(150);
x2=random(150);
y2=random(150); PaintBox1->Canvas->Ellipse(x,y,x2,y2);
}
}
//-----

```



Fig.4 Received image.

Table 1

The value of the Color property determines the color of the line

	Constant	Color	Constant	Color		
	clBlack	Black	clSilver	Silver		
	clMaroon	Chestnut	clRed	Red		
	clGreen	Green	clLime	Salad		
	clOlive	Olive	clBlue	Blue		
	clNavy	Navy blue	clFuchsia	hot pink		
	clPurple	Pink	clAqua	Turquoise		
	clTeal	green-blue	clWhite	White		
	clGray	Grey				

The values of the Brush, style property determine the type of painting

Constant	Area fill type	
bsSolid	Solid fill	
bsClear	The area is not painted over	
bsHorizontal	Horizontal hatching	
bsVertical	Vertical hatching	
bsFDiagonal	Forward diagonal hatching	
bsBDiagonal	Diagonal hatching with lines sloping back	
bsCross	Horizontal-vertical hatching, checkered	
bsDiagCross	Diagonal hatching, checkered	

Plotting with the Chart Component

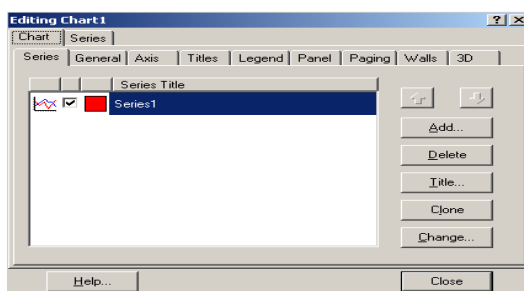
Usually the results of calculations are presented in the form of graphs and diagrams. The Builder system has a powerful package of standard programs for displaying and editing graphical information, which is implemented using the Chart component located on the

Additional - component panel. 

The construction of a graph (diagram) is carried out according to the calculated values of the coordinates of the points x and $y = f(x)$, which are transferred using the AddXY method to a special two-dimensional array Series[k] of the Chart component ($k = 0, 1, 2, \dots$ – number chart used).

The Chart component builds and lays out axes, draws a coordinate grid, signs the names of the axes and the chart itself, and displays the transferred points in the form of graphs or charts.

Having installed the Chart1 component on the form, to change its parameters, double-click to call the EditingChat1 editing window. To create Series1, click the Add button on the Series page.



In the TeeChart Gallery window that appears after this, select the icon labeled Line (the graph is displayed as lines). If there is no need to present the chart in 3D, the 3D check box

is disabled. To change the title, press the Title button. The title of the chart is entered on the Titles page.

Data along the X-axis is automatically sorted, so if you need to draw, for example, a circle, sorting is disabled by the Order function: Chart1->Series[0]->XValues->Order = loNone.

Explore other features of the EditingChat editor by clicking on the various menu buttons.

Using the canvas class

For drawing, a class of type TCanvas is used, which is not an independent component, but a property of many components, such as Image, PaintBox, and is a canvas (GDI context in Windows) with a set of tools for drawing. Each point on the canvas has its own coordinates. The origin of the coordinate axes is located in the upper left corner of the canvas. Data on the x-axis increases from left to right, and on the y-axis from top to bottom. The Image component is on the Additional page, and the PaintBox is on the System.

The main properties of the Canvas class:

Pen - pen (defines line parameters),

Brush - brush (defines the background and filling of closed shapes),

Font - font (defines font parameters).

Some methods of the Canvas class:

Ellipse (x1, y1, x2, y2) - draws an ellipse in the enclosing rectangle (x1, y1), (x2, y2) and fills the interior of the ellipse with the current brush;

MoveTo (x, y) - moves the pencil to position (x, y);

LineTo (x, y) - draws a line from the current pen position to the point (x, y);

Rectangle (x1, y1, x2, y2) - draws and fills a rectangle (x1, y1), (x2, y2). To draw without filling, use FrameRect or Polyline;

Polygon(const TPoint* Points, const int Points_Size) - draws a polygon using the points specified in the Points array of size Points_Size. The end point is connected to the start point and the polygon is filled with the current brush. For drawing without filling, the Polyline method is used.

TextOut(x, y, const AnsiString Text) - outputs the string Text so that the upper left corner of the rectangle enclosing the text is located at the point (x, y).

Example of creating a windowed application

Task.1. Write a program for displaying a graph of the selected function using the Chart and Image components.

Form customization

The program dialog panel with the results obtained is shown in fig. 5.

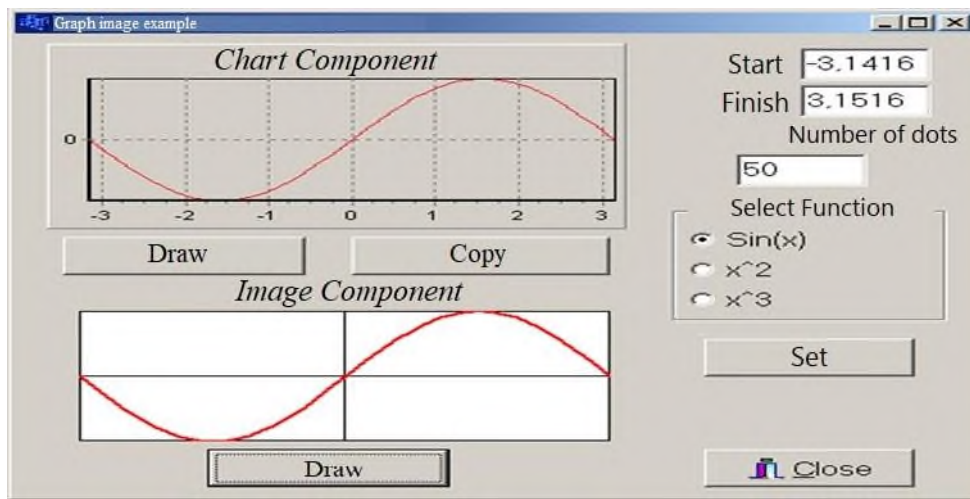


Fig.5.

The text of the program that implements the task may have the following form:

```
//-----
double a,b,h,y_min,y_max;
int n;
typedef double (*Tfun)(double);
Tfun f;
double fun0(double);
double fun1(double);
double fun2(double);
//-----
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Edit1->Text="-3,1416"; // a
    Edit2->Text="3,1416"; // b
    Edit3->Text="50"; // n
    RadioGroup1->ItemIndex = 0; }
//----- Give initial value -----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    double x, r;
    a=StrToFloat(Edit1->Text);
    b=StrToFloat(Edit2->Text);
    n=StrToInt(Edit3->Text);
    h = (b-a)/n;
    switch(RadioGroup1->ItemIndex) {
        case 0: f = fun0; break;
        case 1: f = fun1; break;
        case 2: f = fun2; break; }
    y_min = y_max = f(a);
    for (x = a+h; x<=b; x+=h)
    {
        r = f(x);
        if(y_min>r) y_min = r;
    }
}
```

```

        if(y_max<r) y_max = r;
    } }
//----- Plot in Chart -----
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Chart1->Series[0]->Clear();           // Clear graphic
for(double x=a; x<=b; x+=h)
    Chart1->Series[0]->AddXY(x,f(x));
}
//----- Copy to clipboard -----
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Chart1->CopyToClipboardMetafile(True);
}
//----- Plot in Image -----
void __fastcall TForm1::Button4Click(TObject *Sender)
{
int xmax, ymax, xt, yt, y0, x0;
double hx,hy,x;
Image1->Canvas->Pen->Color=clBlack; // Setup the color

// Finding the coordinates of the lower right corner of the canvas Image
xmax = Image1->Width;           ymax = Image1->Height;
// Painting the canvas Image with the current white brush
Image1->Canvas->Rectangle(0,0,xmax,ymax);
//Finding the middle of the canvas
y0=ymax/2;    x0=xmax/2;
// Drawing coordinate lines
Image1->Canvas->MoveTo(0,y0);
Image1->Canvas->LineTo(xmax,y0);
Image1->Canvas->MoveTo(x0,0);
Image1->Canvas->LineTo(x0,ymax);
Image1->Canvas->Pen->Color=clRed;           // Setting pen color
Image1->Canvas->Pen->Width=2;             // Setting the pen width
/

/ Finding steps in x and y with scaling
hx=(b-a)/xmax;    hy=(y_max-y_min)/ymax;
Image1->Canvas->MoveTo(ceil(x0+a/hx),ceil(y0-f(a)/hy));
for(x=a; x<=b; x+=h)
    Image1->Canvas->LineTo(ceil(x0+x/hx),ceil(y0-f(x)/hy));
}
//-----
double fun0(double r) {
    return sin(r);
}
double fun1(double r) {

```

```

    return r*r;
}
double fun2(double r) {
    return r*r*r;
}

```

Task 2: Draw the national flag of the Republic of Uzbekistan.

Let's install one component Button1 on the form and rename it to "Draw" and the second button Button2 and rename it to "Exit".

We write the program code for these buttons in the following:

```

form:#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//-----
#pragma package (smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Canvas->Pen->Color = clBlue;
Canvas->Brush->Color = clBlue;
Canvas->Rectangle(70,50,600,150);
// drawing moon
Canvas->Pen->Color = clBlue;
Canvas->Brush->Color = clWhite;
Canvas->Ellipse(80,55,170,145);
Canvas->Pen->Color = clBlue;
Canvas->Brush->Color = clBlue;
Canvas->Ellipse(105,55,195,145);
//yulduz chizish
Canvas->Font->Name = "Monotype Corsiva";
Canvas->Font->Color = clWhite;
Canvas->Font->Size = 28;
Canvas->TextOut(180,60," ***");
Canvas->TextOut(180,85,"*****");
Canvas->TextOut(180,110,"*****");
    Canvas->Pen->Color = clRed;
Canvas->Brush->Color = clRed;
Canvas->Rectangle(70,150,600,160);
Canvas->Pen->Color = clWhite;

```



```

Canvas->Brush->Color = clWhite;
Canvas->Rectangle(70,160,600,260);
Canvas->Pen->Color = clRed;
Canvas->Brush->Color = clRed;
Canvas->Rectangle(70,260,600,270);
Canvas->Pen->Color = clGreen;
Canvas->Brush->Color = clGreen;
Canvas->Rectangle(70,270,600,370);
}
//-----

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Form1->Close(); }

```

As a result, we get the following image:

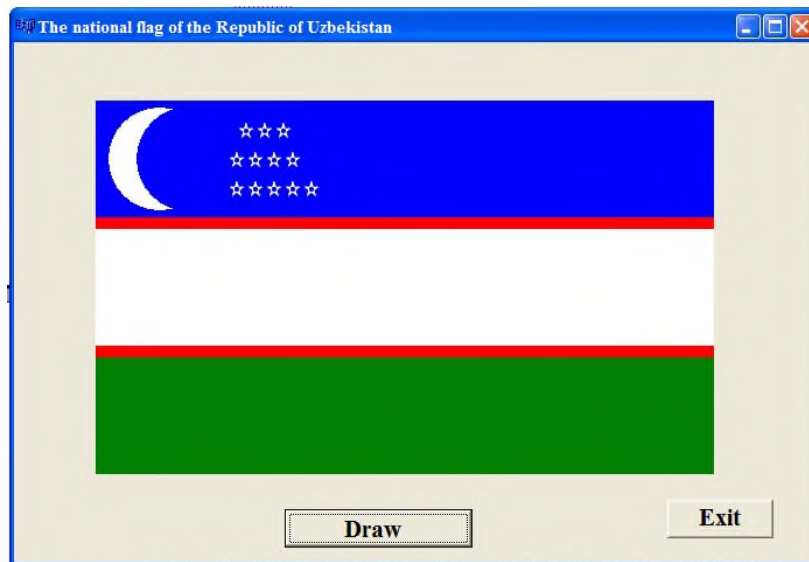


Fig.6 The resulting image when executing the program

Control questions:

1. What methods exist in C++ to animate objects?
2. What tools are used to draw in the Canvas class?
3. What is the Picture class used for?
4. How is the insertion of text for display on the canvas class Canvas?
5. What classes are used to work with images in Borland C++?
6. What components are used to draw on the canvas (Canvas class)
7. What drawing methods of the Canvas class were used in the program?
8. How is the color of an object selected and set in the Canvas?

MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY
NAMED AFTER ISLAM KARIMOV



LABORATORY WORKS

On subject

“Information technologies in technical systems”

Tashkent 2022

Laboratory work №1

Basics of work in the MathCAD system. System capabilities. Program interface. Creation of MathCad-document and their application in technical systems. Usage of simple functions.

The purpose of the lesson: Familiarization with the windows of MathCad. Create a Mathcad document. Using simple functions. Solving systems of equations in MathCad. Using complex variables, explore functions to solve differential equations.

Theoretical part

Creating a Mathcad Document

After starting Mathcad, the main window appears, which has the same structure as most Windows applications. At the top is the menu bar, then the toolbars (standard and formatting by default) and the worksheet. At the very bottom of the window is the status bar.

Using the View / Tools menu, you can call up any panel available in the drop-down submenu on the document workspace. On fig. 1. all kinds of Mathcad panels are shown.

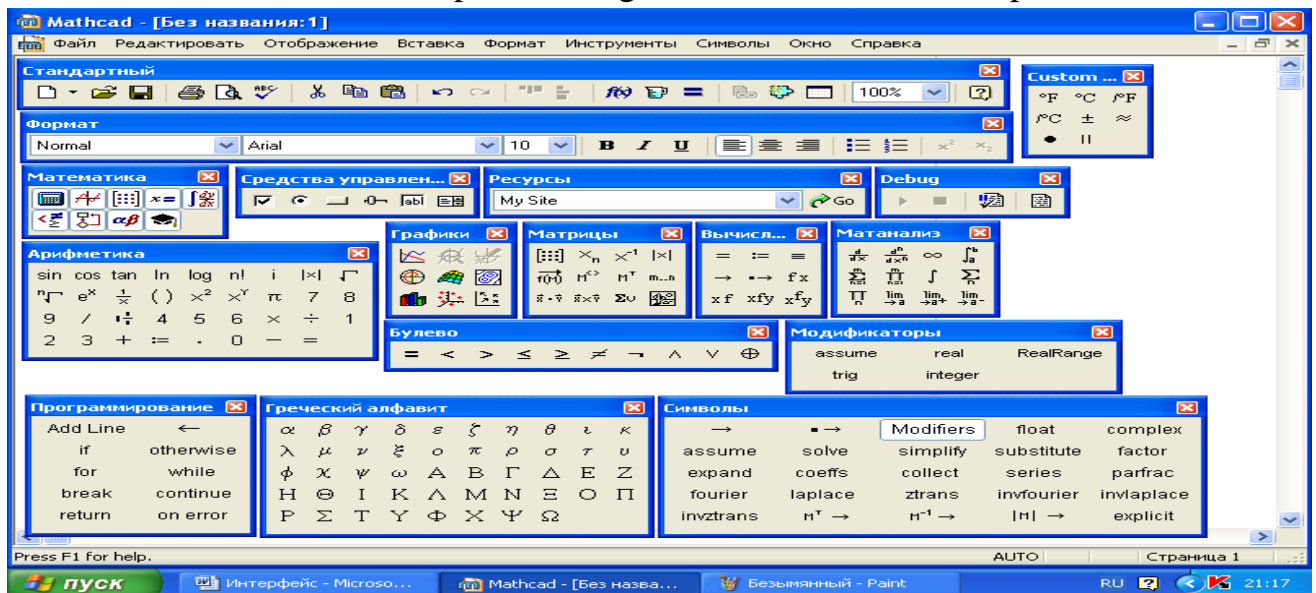


Fig. 1. Working paper Mathcad and its toolbars.

Main elements of the Mathcad user interface:

- Mathcad's main menu (menu bar);
- toolbars Standard, Formatting, Resources and Controls;
- mathematical palette Math and additional mathematical toolbars available through it;
- working area (work sheet), imitating a blank sheet of paper;
- status bar;
- pop-up or context menus (pop-up menus or context menus), called by the right mouse button (RMB);
- dialog boxes or dialogs (dialogs), caused by the left mouse button (LMB);
- Resources window with built-in examples and additional information.

Most commands can be executed both using the menu (main or context), and using toolbars or the keyboard.

Toolbars. Toolbars are used for quick (one-click LMB) execution of the most commonly used commands. Groups of buttons on toolbars are delimited in meaning by vertical lines - separators.

When you move the mouse pointer over any of the buttons, a tooltip appears next to the button - a short text explaining the purpose of the button (in English). Along with the tooltip, a detailed explanation of the command can be found in the status bar. Figure 1 shows the Mathcad 13 window with the following toolbars:

- Standard - serves to perform most operations, such as file operations, editing, inserting objects and accessing help systems;

- Formatting - for formatting (changing the font type and size, alignment, etc.) text and formulas;

- Resources - for quick access to Mathcad resources (examples, textbooks, e-books, etc.);

- Controls - for inserting standard user interface controls (validation checkboxes, input fields, etc.) into documents;

- Math - designed to insert mathematical symbols and operators into documents. With its help, you can call the panel on the screen:

- Calculator(Arithmetic) - is used to insert basic mathematical operations; got its name because of the similarity of the set of buttons with the buttons of a typical calculator;

- Graph (Graphics) - for inserting chart templates;

- Matrix (Matrices) - for inserting matrix templates and matrix operators;

- Evaluation(Expressions) - for inserting calculations control operators;

- Calculus (calculations) - for inserting operators of integration, differentiation, summation;

- Boolean - for inserting logical (boolean) operators;

- Programming - for developing programs and user functions Mathcad;

- Greek (Greek alphabet) - to insert Greek characters;

- Symbolic (Symbols) - for inserting operators and performing symbolic transformations;

- Debug (Debugging) - for debugging programs;

- Modifiers – for modification of analytical transformations;

- Custom Characters (Some characters) - to insert rarely used characters.

Mathcad workspace (worksheet)

In Mathcad's workspace, only three types of zones can be set: mathematical, text and graphic. The following two well-known concepts are also associated with it: the mouse pointer and the cursor.

The mouse pointer is an arrow moved by the mouse manipulator. With its help, you can specify the location of zones, press various buttons located in the main menu, in pop-up dialog boxes, select and execute various commands, make various applications active, etc.

The cursor is a tool for editing the MathCAD workspace. It has various forms:

- cross-shaped cursor (finder). It is displayed in red and is used to place zones in the free space of the workspace. To install it, you need to move the mouse pointer to an empty spot

in the workspace and click LMB. The reticle indicates where the next printed character will be placed, or where the area from the clipboard or object template will be pasted. You can move the reticle around the screen by pressing the cursor movement keys, as well as the PgUp and PgDn keys (by 80% of the screen);

- editing lines. They consist of two lines: an underline (horizontal) and an insert (vertical). They form a blue right angle and are used only in the math zone to edit its contents. Using editing lines to highlight parts of an expression, you can modify, delete, add, etc.;

- input marker. It is a thin vertical red line. It is only used in the text zone.

Creating a math zone

To create a mathematical zone, it is enough to set the sight in the free space of the workspace, and then start typing a certain sequence of characters. Part of the characters are letters and numbers for entering variable names, numbers and functions, the other part is used to create operations. After each operation is entered, MathCAD prints a small black box called an input marker to record the operands of the operation. MathCAD also takes into account their priority when performing operations.

In addition to the ordinary assignment operator, Mathcad has another global assignment operator (\equiv), which is located on the Calculations panel (hot key ~). If you insert it to set the value of a variable in any part of the document (for example, at the very bottom), then this variable will automatically be defined in any part of the document (including at the top). If the variable has not been assigned any value, it is perceived analytically, just as a name. When performing symbolic conversions, the symbolic equals sign (\rightarrow) is used instead of the normal assignment sign. It can be entered into a working document from any of the Evaluation (Expressions) or Symbolic (Symbols) panels.

Entering functions into a MathCad document

When performing complex and cumbersome calculations, you can use the entire arsenal of functions that developers put into the Mathcad system. You can enter the names of such functions from the keyboard, but it is better to use the Insert Function dialog box (Place the function), Fig.2.

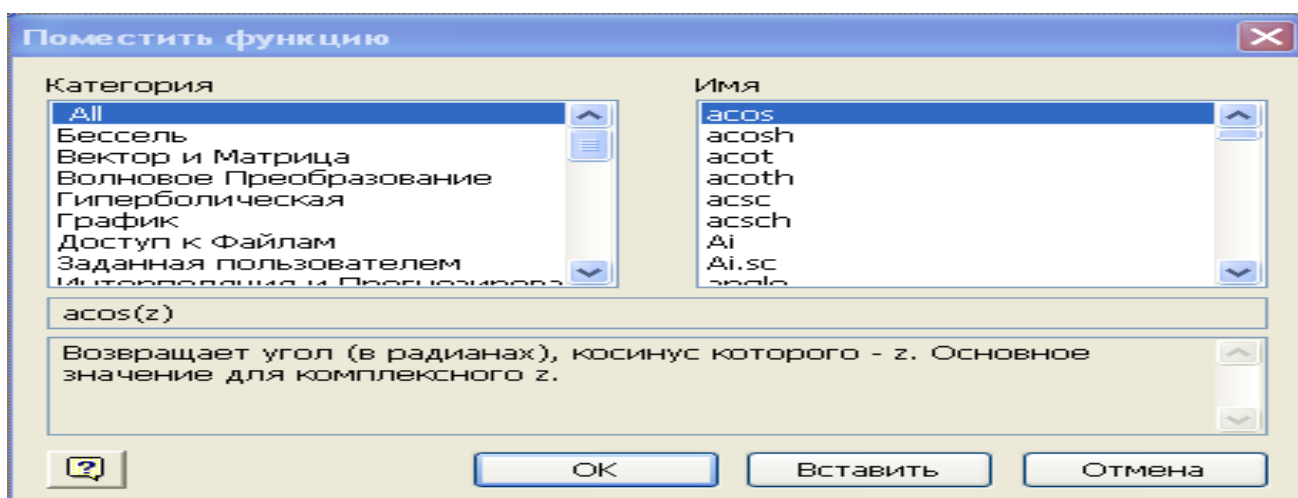


Fig. 2. Inserting a built-in function using a dialog box

To enter a built-in function in an expression, you need to determine the place in the expression where the function should be inserted, press the f (x) button on the standard toolbar. In the Function Category list (Category) of the appeared dialog box Insert Function (Place a function), you need to select the category to which the function belongs, and in the list Function Name - the name of the built-in function and click the OK button. When the function format appears in the document, the required arguments must be entered in its markers.

Similar to assigning numeric values to variables, you can define user functions from one or more arguments,

For example

$$f(x) := x^2 - 3x - 2 \quad f(0) = -2 \quad f(1) = -4$$

Data types used in Mathcad

Let's list the main types of variables that are used in Mathcad.

Real numbers

Any expression that starts with a digit is interpreted by Mathcad as a number. You can organize the input of numbers in decimal, binary, octal or hexadecimal number systems. When writing numbers, suffixes b, o, h are used, for example, fig. 3.

Entering numbers in different number systems

$$\begin{array}{ll} a := 100010b & a = 34 \\ b := 13o & b = 11 \\ c := 0f3h & c = 243 \end{array}$$

Fig.3.

Complex numbers

Most of the operations in the Mathcad environment are by default performed on complex numbers. A complex number is the sum of a real number and an imaginary number, obtained by multiplying any real number by an imaginary unit i. By definition, $i^2 = -1$. To enter an imaginary number, for example 3i, press the <3>, <i> keys. If you simply enter the character "i", then Mathcad interprets it as the variable i. Also, the imaginary unit has the form 1i only when the corresponding formula is highlighted. Otherwise, the imaginary unit is displayed simply as i (Fig. 4.).

Entering an imaginary unit

$$a := i + 10 \quad \boxed{x := 1i} \quad x = i$$

Fig.4.

A complex number can be entered as an ordinary sum, real and imaginary parts, or as any expression containing an imaginary number. Examples of input and output of complex numbers are illustrated in fig.5.

Input/output of complex numbers

$$\begin{array}{ll} a := 2i + 10 & a = 10 + 2i \\ b := 1.77 \cdot e^{2i} & b = -0.737 + 1.609i \\ c = 12 + 25i & c := 25j + 12 \end{array}$$

Fig.5.

You can display the imaginary unit in the results of calculations not as i, but as j. To change the representation, you can select the desired one in the Imaginary Value list (Imaginary value) of the Result Format dialog box (Result Format), accessible by the command Format / Result / Screen settings.

There are built-in functions for working with complex numbers, fig. 6.

Using built-in functions

$$\begin{array}{lll} \text{Im}(a) = 2 & \text{Re}(a) = 10 & \text{arg}(a) = 0.197 \\ |a| = 10.198 & |b| = 1.77 & \text{arg}(b) = 2 \end{array}$$

Fig.6.

Built-in Constants

Some names in Mathcad are reserved for system variables called built-in constants. Built-in constants are divided into two types: mathematical (math constants), which store the values of some commonly used special mathematical symbols, and system variables, which determine the operation of most numerical algorithms implemented in Mathcad. If desired, you can change the value of any of the listed constants or use them as variables in calculations. If you assign a new value to a constant, the old value becomes unavailable.

Mathematical constants are interpreted differently in numerical and symbolic calculations. The computing processor simply perceives them as some numbers, and the symbolic processor recognizes each of them, based on the mathematical context, and is able to issue mathematical constants as a result. Let's list the mathematical constants:

- ∞ - infinity symbol (entered with Ctrl+Shift+z);
- e - the base of the natural logarithm (keyboard);
- π - "pi" number (entered with Ctrl+Shift+p);
- i, j - imaginary unit (entered with keys I or J);
- % - percent symbol, % is equivalent to 0.01.

Solving a system of equations in Mathcad

Let's first consider **SLAE** in Mathcad. To solve them, the **given ...find()** block or the special **lsolve()** function can be used. The use of the **given ...find()** block predetermines the need to set the initial values of the required variables. Further, after the given keyword, the **SLAE** is described and the solution is found using **find()**. It should be pointed out that in

the case when the **SLAE** in Mathcad has an infinite set of solutions, the **given ...find()** block gives a specific result, which should undoubtedly be attributed to disadvantages. If there is no solution, the message “**Matrix is singular. Cannot compute its inverse - The matrix is singular. It is impossible to calculate this inversion.**” Using the **lsolve()** function avoids this shortcoming. The **lsolve(M,b)** function has two arguments. **M** is the matrix of coefficients for unknowns, **b** is the vector of free terms. The listing shows an example of **SLAE** solution. (Fig. 7.)

An example of a **SLAE** solution:

$x := 0 \quad y := 0$ Given $x + 3y = 5$ $2 \cdot x + y = 12$ Find (x, y) = $\begin{pmatrix} 6.2 \\ -0.4 \end{pmatrix}$	$M := \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix} \quad b := \begin{pmatrix} 5 \\ 12 \end{pmatrix}$ $lsolve(M, b) = \begin{pmatrix} 6.2 \\ -0.4 \end{pmatrix}$
---	--

Example of SLAE solution in the case of an infinite set of solutions

$z := 0 \quad p := 0$ Given $2 \cdot z + 3 \cdot p = 6$ $4 \cdot z + 6 \cdot p = 12$ Find (z, p) = $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$	$T := \begin{pmatrix} 2 & 3 \\ 4 & 6 \end{pmatrix} \quad k := \begin{pmatrix} 6 \\ 12 \end{pmatrix}$ $lsolve(T, k) = \blacksquare$ a message "Matrix is singular. Cannot compute its inverse" is issued
---	---

Fig.7.

An ordinary differential equation of the first order, by definition, contains, in addition to the desired function $y(t)$, only its first derivative $y'(t)$. In the vast majority of cases, the differential equation can be written in the standard form (Cauchy form), resolved with respect to the highest derivative: $y'(t)=f(y(t),t)$

Only with this form the Mathcad computing processor can work. A mathematically correct statement of the corresponding Cauchy problem for a first-order ODE should, in addition to the equation itself, contain one initial condition - the value of the function $y(t_0)$ at some point t_0 . It is required to explicitly define the function $y(t)$ on the interval from t_0 to t_x . By the nature of the formulation of the Cauchy problem, they are also called problems with initial conditions (initial value problem), in contrast to boundary value problems. For the numerical integration of one ODE, the user of Mathcad has a choice - either to use the computational block Given - Odesolve(), or built-in functions. The first way is preferable for reasons of visual representation of the problem and results, and the second gives the user

more leverage to influence the parameters of the numerical method. Let's consider both solutions one by one. Computing block Given - Odesolve()

The computational unit for solving one ODE, which implements the Runge-Kutta numerical method, consists of three parts:

- Given - keyword;
- ODE and the initial condition, written using Boolean operators, and the initial condition must be in the form $y(t_0) = b$;
- Odesolve(t, t1) - built-in function for solving ODE with respect to variable t on the interval (t0,t1).

It is acceptable, and even often preferable, to specify the function Odesolve (t, t1, step) with three parameters, where step is an optional internal parameter of the numerical method that determines the number of steps in which the Runge-Kutta method will solve the differential equation. The larger the step, the better the result will be, but the more time will be spent on its solution. Thus, by selecting this parameter, one can noticeably (several times) speed up the calculations without a significant deterioration in their accuracy.

An example of solving the Cauchy problem for the first order ODE $y'=y-y^2$ by means of the computational block Given – Odesolve() is shown in fig. 8. Insert logical operators using the Boolean toolbar. When typing a logical equals sign from the keyboard, use the Ctrl = key combination. The symbol of the derivative can be entered both by means of the Calculus panel (Calculations), as is done in fig. 8 and as a dash (') by typing it using the keyboard shortcut Ctrl + F7.

Given

$$\frac{d}{dt}y(t) = y(t) - y(t)^2 \quad y(0) = 0.01$$

y := Odesolve(t,10)

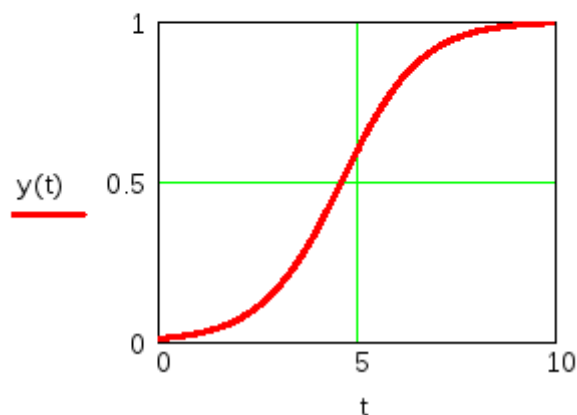


Fig.8.

Higher order ordinary differential equation

An ordinary differential equation with an unknown function $y(t)$, which includes derivatives of this function up to $y^{(n)}(t)$, is called an nth order ODE. If there is such

an equation, then for the correct formulation of the Cauchy problem, in addition to the equation itself, it is required to set n initial conditions for the function $y(t)$ itself and its derivatives from the first to $(n - 1)$ order inclusive. In Mathcad, you can solve higher-order ODEs both with the help of the `Given-Odesolve()` computational block, and with an alternative method using functions of the `rkfixed()` type.

Inside the `Given-Odesolve()` compute block:

- ODE must be linear with respect to the highest derivative;
- initial conditions must have the form $y(t_0)=b$ or $y^{(n)}(t_0)=b$;

Otherwise, the solution of higher order ODEs is no different from the solution of first order equations.

On fig. 9 shows the solution of a second-order differential equation for a damped harmonic oscillator, which describes, for example, the oscillations of a pendulum. For the pendulum model, the function $y(t)$ describes changes in the angle of its deviation from the vertical, $y'(t)$ is the angular velocity of the pendulum, $y''(t)$ is the acceleration, and the initial conditions, respectively, the initial deviation of the pendulum $y(0) = 0.1$ (in radians) and initial velocity $y'(0)=0$.

```
Given
y''(t) + 0.1 · y'(t) + y(t) = 0      y(0) = 0.1   y'(0) = 0
y := Odesolve(t, 50)
```

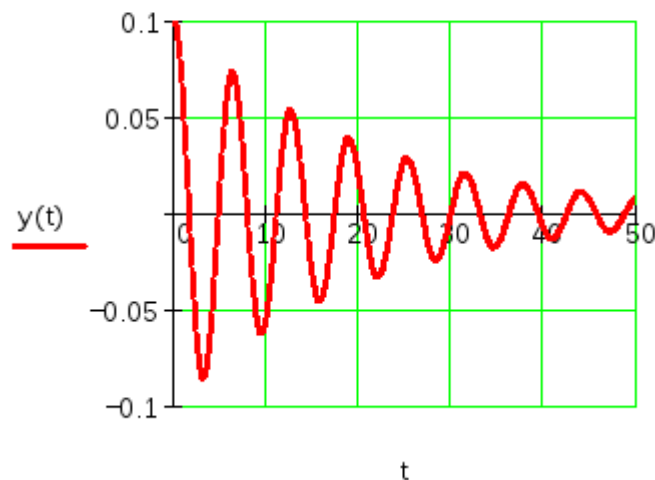


Fig. 9. Solution of a second-order differential equation

Solution with the creation of a decision block and **the given** directive. **The given** directive and the **Find** operator are typed from the keyboard

Given

$$3 \cdot x_1 + 4 \cdot x_2 + 4 \cdot x_3 + 6x_4 + 8x_5 = 21$$

$$5 \cdot x_1 - 7 \cdot x_2 + 8 \cdot x_3 - 7x_4 - 9x_5 = 34$$

$$9 \cdot x_1 + 12 \cdot x_2 + 9 \cdot x_3 + 8x_4 + 4x_5 = 41$$

$$13 \cdot x_1 + 3 \cdot x_2 + 19 \cdot x_3 + 13x_4 + 9x_5 = 14$$

$$23 \cdot x_1 + 13 \cdot x_2 + 9 \cdot x_3 + 3x_4 + 19x_5 = 24$$

$$\text{Find}(x_1, x_2, x_3, x_4, x_5) \rightarrow \begin{bmatrix} \frac{30963}{2539} \\ \frac{-58913}{10156} \\ \frac{4366}{2539} \\ \frac{-15189}{5078} \\ \frac{23741}{10156} \end{bmatrix}$$

Fig10. Symbolic solution of a system of linear equations using **the given** directive

Solving a system with letter coefficients

Given

$$a_1 \cdot x + b_1 \cdot y + c_1 \cdot z = d_1$$

$$a_2 \cdot x + b_2 \cdot y + c_2 \cdot z = d_2$$

$$a_3 \cdot x + b_3 \cdot y + c_3 \cdot z = d_3$$

$$\text{Find}(x, y, z) \rightarrow \begin{bmatrix} \frac{(-d_3) + d_2}{(-a_3) + a_2} \\ \frac{-[(-a_2) \cdot c_2 \cdot d_1 + a_2 \cdot d_3 \cdot c_1 + c_2 \cdot a_1 \cdot d_2 - c_2 \cdot a_1 \cdot d_3 + c_2 \cdot d_1 \cdot a_3 - d_2 \cdot a_3 \cdot c_1]}{[(-a_3) + a_2] \cdot (b_1 \cdot c_2 - c_1 \cdot b_2)} \\ \frac{-[(-a_1) \cdot b_2 \cdot d_2 - b_1 \cdot a_2 \cdot d_3 + b_1 \cdot a_3 \cdot d_2 + a_1 \cdot d_3 \cdot b_2 - d_1 \cdot a_3 \cdot b_2 + d_1 \cdot b_2 \cdot a_2]}{b_1 \cdot a_2 \cdot c_2 - b_1 \cdot a_3 \cdot c_2 + c_1 \cdot a_3 \cdot b_2 - c_1 \cdot b_2 \cdot a_2} \end{bmatrix}$$

Fig.11. Symbolic solution of a system of linear equations given in literal form

Symbolic solution of non-linear algebraic equations

1. Solving a fourth-degree equation with numerical coefficients using **the solve** operator

$$3x^4 + 5x^3 + 7x^2 + 9x + 10 \text{ solve, } x \rightarrow \begin{bmatrix} (-1.1260186005688105923 - .75699694283657936689i) \\ (-1.1260186005688105923 + .75699694283657936689i) \\ .29268526723547725894 + 1.3133859400283537322i \\ .29268526723547725894 - 1.3133859400283537322i \end{bmatrix}$$

Fig.12.

2. Solving a quadratic equation with letter coefficients

$$a \cdot x^2 + b \cdot x + c = 0 \text{ solve } x \rightarrow \left[\frac{-b}{2 \cdot a} \pm \sqrt{\left(\frac{-b}{2 \cdot a} \right)^2 - \frac{c}{a}} \right]$$

Fig.13.

Task 2. Solve the following system on your own

$$3 \cdot x_1 + 4 \cdot x_2 + 4 \cdot x_3 + 6x_4 = 21$$

$$5 \cdot x_1 + 7 \cdot x_2 + 8 \cdot x_3 - 7x_4 = 34$$

$$9 \cdot x_1 + 12 \cdot x_2 + 9 \cdot x_3 + 8x_4 = 41$$

$$13 \cdot x_1 + 3 \cdot x_2 + 19 \cdot x_3 + 18x_4 = 141$$

Fig.14.

Task 3. Solve the following equations yourself

$$1) x^2 + x + 1 = 0$$

$$2) 3x^3 + 2x^2 + 5x + 6 = 0$$

$$3) ax^4 + bx^2 + d = 0$$

Task No. 1 Calculate variables A and B for given values of X, Y on MathCad.

Table 1

Nº	Arithmetic expressions	Given
1	$A = \frac{\sqrt{ x-1 } - \sqrt[3]{y}}{1 + \frac{x^2}{2} + \ln \frac{y}{4}}$ $B = x \cdot (\arctg A) + e^{-(x-1)}$	X=3 Y=-1,4
2	$A = \frac{3 + e^{y-1}}{1 + x^2(y - \cos(x-3))}$ $B = 1 + \sqrt{ A-y } + \sqrt[3]{y-x} + \frac{(y-a)^2}{2}$	X=2 Y=3,1
3	$A = (1+y) \frac{x^2 + 4}{e^{y-2} + \sqrt{x^2 + 4}}$ $B = (1 + \operatorname{tg}^2 A \pi / 2) \sqrt[5]{x^2 + 4}$	X=-2,3 Y=2,7
4	$A = y + x^2 + \left \frac{e^x + x^3}{e^y + 1} \right $ $B = \frac{1 + \cos(\alpha - 2)}{x^4 + \sin^2(x+y)} + \sqrt[5]{x}$	X=-1,3 Y=2,5
5	$A = \cos^3(x+y) + \sqrt[3]{e^x + y}$ $B = \ln(A^2 + 1) + \frac{\sqrt{ x+y }}{10^{-3} + x^2 - x^3}$	X=4,6 Y=-6,2

6	$A = \frac{1 + \sin^2(x-2y)}{x+y^2 + \cos x} + \operatorname{tg}^2 x$ $B = \cos\left(1 + \frac{Ax-y}{e^x + 10^2} - \sqrt[3]{A}\right)$	X=2,3 Y=3,4
7	$A = \ln(y - \sqrt{x} + e^{x+y}) + \sqrt[3]{x-y}$ $B = (x + \operatorname{tg} \frac{2\pi}{A})(5 * 10^{-6} + \frac{ x-7 }{A})$	X=3,6 Y=5,5
8	$A = \left \frac{\sin^2(\pi-x)}{\sqrt{(x-y)^2 + e^x}} \right $ $B = \sqrt{\operatorname{tg} \frac{\pi}{A}} \ln(2 * 10^3 - \cos(x-8y))$	X=3,3 Y=1,8
9	$A = e^{-(x+1)} \sqrt{\frac{1,9 + \sin 2x}{1,1 - \cos(y^2 - 18)}}$ $B = x \ln \left \frac{A}{x^2 + 1} \right + \operatorname{ctg} \frac{y^2 - 18}{x}$	X=0,84 Y=-4,2
10	$A = \frac{e^{x+y} + \sqrt{x+y} - 1,6 * 10^{-7}}{2 - \sin^2(x+y) + xy + 1,3}$ $B = \operatorname{arctg} \frac{x+y}{A} + \frac{A}{\cos(x+y+1,3)}$	X=-0,4 Y=3,25
11	$A = (x+1) \frac{x}{x^2 + y^2} + \frac{A}{\cos(xy)}$ $B = \frac{\sqrt{x+y} - A + 7,6}{\cos^2(x-y) + \sqrt{x^2 + y^2}} + e^x$	X=4,32 Y=-1,6
12	$A = \operatorname{tg} x^2 + (\sin 2y + \ln(x^2 + 2y))^3$ $B = Ax + \sqrt{ 7 - \cos^2 y } + 2x$	X=3 Y=1,24
13	$A = \cos^2(x^2 + 2y) + \frac{\ln(x^2 + 2y)}{x^2 + 2y + e^{2x}}$ $B = \sqrt{\frac{x^2 + 2y}{A + e^y}} \sin x + \sqrt[4]{xy}$	X=0,32 Y=0,17
14	$A = \frac{x}{y} + \frac{x/y + x + \sqrt[3]{y+y}}{\sin(x+y) + 4,32}$ $B = e^{Ax} \sqrt{1 + \ln \frac{x(x+y)}{y}} + \operatorname{tg} Ax$	X=3,1 Y=2,27
15	$A = \sqrt{\frac{x^3 + e^y + \cos^2(y+1)}{x+y}}$ $B = \operatorname{tg}^2(x^3 + A + e^y) + \frac{\lg(Ay)}{4x + 7,3 * 10^{-3}}$	X=3,53 Y=2,4
16	$A = \operatorname{ctg} \frac{xy + \ln(xy) + \sin^3 x}{e^{\cos x} + 0,64 * 10^4}$ $B = \lg \left \frac{2A + \sqrt[3]{xy} + \sin x}{e^x + (1 + xy)^2} \right $	X=-1,9 Y=6,75
17	$A = (\lg x + y^2 + \cos \sqrt[3]{y^2});$ $B = \frac{A(2,7 + y)}{\lg x + y^2 - x} + \sqrt{\lg x + y^2}$	X=10,1 Y=9,5
18	$A = \frac{\sin^2(\Pi - x) + x + e^y}{e^{x+y} (18,6 + x + e^y)^y};$ $B = A^2 \operatorname{arctg} \sqrt{8 + x + e^y} + \ln(x-y)$	X=1,7 Y=2,4
19	$A = \operatorname{ctg} \frac{x+y}{e^y} + \left(\frac{\sin y + x+y}{8y + \cos(x+y)}\right)^{-3};$ $B = \frac{\sqrt{\sqrt{x+y} + A^2 y}}{e^y + x }$	X=-1,12 Y=2,17
20	$A = \cos^2(x-2)^3 + y^2 + 1 + e^{x-2};$ $B = \ln \left \frac{A(y^2 + 1) + \sqrt{e^{x-2}}}{\operatorname{tg}(y^2 + 1 + A)} \right $	X=-4,3 Y=7,11
21	$A = \frac{3,002(x^2 + 4,2)}{\sin^2(y+1)} + \sqrt{x^2 + 4,7 + \sin^2(y)}$ $B = \left \frac{x^4 - y + 1}{x^2 + 4,7 + \operatorname{tg}(y-1)} \right ^{-\sqrt[5]{x+4,7}}$	X=0,07 Y=0,41
22	$A = \frac{(x-y^2) + \operatorname{tg} x}{(x-y^2 + e^x)} + \cos \sqrt{1 + \lg(x+y)}$ $B = \sqrt[3]{A(x-y^2)} + \frac{\operatorname{tg}(x-y^2) * A}{e^x}$	X=6,03 Y=3,42

23	$A = \cos^3\left(\frac{\sqrt{x}}{e^y} + 7,6\right) + \frac{\lg(y) + 1,3 \cdot 10^5}{(x^2 + 1 + \frac{\sqrt{x}}{e^x})}$ $\sqrt{\left \ln \frac{(x^2 + 1)\sqrt{x}}{e^x}\right } + \sqrt[3]{Ay}$	B=	X=2,17 Y=0,35
24	$A = \left(\cos^2 x + y^2 + \frac{x}{1+y}\right)^3$	B=	$\sqrt{\cos^2 x + y^2} + e^x \operatorname{tg} A$ X=1,43 Y=18,6
25	$A = \ln(x^2 + 4,3) + 7,8xy + \sin^2\left(\frac{\sqrt{xy}}{x^2 + 4,3}\right)$ $\frac{\sqrt{xy + 13,2} + e^x + 4,3}{ Axy - 4,6 \cdot 10^6 }$	B=	X=1,5 Y=2,53
26	$A = \operatorname{ctg} \frac{x+y}{e^y} + \left(\frac{\sin y + x + y}{8y + \cos(x+y)}\right)^{-3}$	B=	$\frac{\sqrt{\sqrt{x+y} + A^2 y}}{e^y + x }$ X=-1,12 Y=2,17
27	$A = \frac{(x-y^2) + \operatorname{tg} x}{(x-y^2 + e^x)} + \cos \sqrt{1 + \lg(x+y)}$ $\left \frac{A}{x^2 + 1}\right + \operatorname{ctg} \frac{y^2 - 18}{x}$	B=xln	X=-3,5 Y=2,47
28	$A = \frac{\sin^2(\Pi - x) + x + e^y}{e^{x+y} (18,6 + x + e^y)^y}$	B=	$A^2 \operatorname{arctg} \sqrt{8 + x + e^y} + \ln(x-y)$ X=1,7 Y=2,4
29	$A = \sqrt{\frac{x^3 + e^y + \cos^2(y+1)}{x+y}}$	B=	$x \ln \left \frac{A}{x^2 + 1}\right + \operatorname{ctg} \frac{y^2 - 18}{x}$ X=-4,3 Y=7,11
30	$A = \operatorname{tg} x^2 + (\sin 2y + \ln(x^2 + 2y))^3$	B=	$Ax + \sqrt{ 7 - \cos^2 y } + 2x$ X=3 Y=1,24

Task No2 Solving a system of linear equations (Given, Find)

$$1. \begin{cases} x + 2y - z = 5, \\ 2x - y + 5z = -7, \\ 5x - y + 2z = -4. \end{cases} \quad 2. \begin{cases} 2x + 3y - 5z = 1, \\ 3x + 4y - 3z = 2, \\ x - 3y + 7z = 5. \end{cases} \quad 3. \begin{cases} 7x - 3y + z = 5, \\ x + 2y - z = -4, \\ 3x + y - z = -3. \end{cases}$$

$$4. \begin{cases} 5x + y + 6z = -3, \\ 4x + 3y - z = 2, \\ x + 2y - 5z = 3. \end{cases} \quad 5. \begin{cases} 5x - 3y + z = -3, \\ 3x - y + 2z = 1, \\ x + 5y + z = 1. \end{cases} \quad 6. \begin{cases} 8x + 2y - 7z = 3, \\ x - 3y + 5z = 3, \\ 5x - 2y + 4z = 7. \end{cases}$$

$$7. \begin{cases} 3x - 4y + z = 5, \\ 2x - y + 3z = 1, \\ x + 5y - z = 3. \end{cases} \quad 8. \begin{cases} 7x - y + 2z = 5, \\ 2x + y - 3z = -7, \\ x - 5y + z = 7. \end{cases} \quad 9. \begin{cases} x - 4y - z = -3, \\ 3x + 7y + z = -1, \\ 2x + 3y - z = -4. \end{cases}$$

$$10. \begin{cases} x + y + z = 3, \\ 3x - 2y + z = 2, \\ 5x + 2y - 7z = 0. \end{cases} \quad 11. \begin{cases} x - 5y + z = 1, \\ 3x + y - 2z = -7, \\ 2x + 7y + z = 0. \end{cases} \quad 12. \begin{cases} 3x - 4y + 7z = -1, \\ x + 7y + 2z = 0, \\ 2x - 3y + z = 3. \end{cases}$$

$$13. \begin{cases} 5x - 3y + z = 9, \\ 3x - 7y + 6z = 0, \\ x + 2y + z = 1. \end{cases} \quad 14. \begin{cases} x + 2y + 5z = -1, \\ 5x + y - 3z = 5, \\ 7x - 4y - 3z = -5. \end{cases} \quad 15. \begin{cases} x - y + 7z = -3, \\ 2x + y - 5z = 0, \\ 3x + 2y - 5z = 1. \end{cases}$$

$$16. \begin{cases} x - y - 2z = 3, \\ 2x + 3y - 7z = 1, \\ 5x + 3y - 4z = 7. \end{cases} \quad 17. \begin{cases} 2x + 3y - z = 4, \\ x + y - 5z = 1, \\ 3x + y - 3z = -1. \end{cases} \quad 18. \begin{cases} x + 2y + z = 3, \\ 3x - y + 2z = -4, \\ 5x + 3y - z = 7. \end{cases}$$

$$19. \begin{cases} 2x + 3y - z = 1, \\ x + 3y - 4z = -1, \\ 3x - 2y + 5z = 8. \end{cases} \quad 20. \begin{cases} 3x - 4y + 7z = -1, \\ x + 7y + 2z = 0, \\ 2x - 3y + z = 3. \end{cases} \quad 21. \begin{cases} x + 2y + 5z = -1, \\ 5x + y - 3z = 5, \\ 7x - 4y - 3z = -5. \end{cases}$$

$$22. \begin{cases} x + 2y - z = 5, \\ 2x - y + 5z = -7, \\ 5x - y + 2z = -4. \end{cases} \quad 23. \begin{cases} 2x + 3y - 5z = 1, \\ 3x + 4y - 3z = 2, \\ x - 3y + 7z = 5. \end{cases} \quad 24. \begin{cases} 7x - 3y + z = 5, \\ x + 2y - z = -4, \\ 3x + y - z = -3. \end{cases}$$

$$25. \begin{cases} 3x - 4y + z = 5, \\ 2x - y + 3z = 1, \\ x + 5y - z = 3. \end{cases} \quad 26. \begin{cases} 7x - y + 2z = 5, \\ 2x + y - 3z = -7, \\ x - 5y + z = 7. \end{cases} \quad 27. \begin{cases} x - 4y - z = -3, \\ 3x + 7y + z = -1, \\ 2x + 3y - z = -4. \end{cases}$$

$$28. \begin{cases} x - y - 2z = 3, \\ 2x + 3y - 7z = 1, \\ 5x + 3y - 4z = 7. \end{cases} \quad 29. \begin{cases} x + 2y + 5z = -1, \\ 5x + y - 3z = 5, \\ 7x - 4y - 3z = -5. \end{cases} \quad 30. \begin{cases} x - 5y + 2z = 9, \\ 3x - y + z = 3, \\ 7x + y - z = -3. \end{cases}$$

Control questions:

1. What processes is the MathCAD system intended for?
2. How are simple calculations carried out in the MathCAD system?
3. What processes is the MathCAD system intended for?
4. How are simple calculations carried out in the MathCAD system?
5. What tool is used to plot function graphs?

6. List the capabilities of the MathCAD system?
7. What does the word solve mean?
8. What does the word Find mean?

Laboratory work №2

Programming in the MathCAD system.

The purpose of the work: to study the formats of programming operators and the technology for developing programs in the MathCAD environment. Developing custom programs in the MathCAD environment

Theoretical part

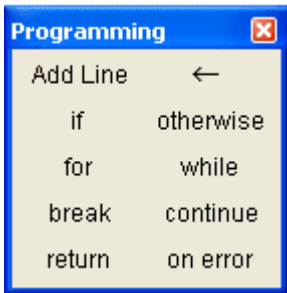
MathCAD allows you to write programs that contain constructs similar to those of programming languages. It has conditional transfers of control, loop statements, distinguishes between scopes of variables, uses subroutines and recursions. Like any expression, a program returns a value if it is followed by an equals sign.

The main difference between a program and an expression is how the calculations are specified. When using an expression, the algorithm for obtaining a response must be described by a single operator. As many statements as needed can be used in a program.

If you click on the mathematical palette button Math (Mathematics) or select Toolbars\Programming from the menu View (Display), you can display the toolbar Programming, the operators of which are presented in table 1. Programming operators are used only in programs MathCAD.

Operators of the Programming panel and their "hot keys".

Table 1

Panel Programming:	Operators:	Hot keys:
	Add line]
	← local assignment	{
	If	}
	otherwise	[Ctrl][Shift]]
	for– cycle operator	[Ctrl][Shift] '
	while – cycle operator	[Ctrl]]
	break –	[Ctrl][Shift] [
	continue	[Ctrl] [
	return	[Ctrl][Shift] \
	on error– error catch operator	[Ctrl] '

Creating programs with a linear structure

Like the functions of the C++ programming language, any MathCAD program usually has a name, arguments written in parentheses, and an assignment operator ($:=$). To the right of the definition sign in the marker marked with a black rectangle (γ), the *AddLine* button writes the program template in the form of a vertical bar and a vector of two lines. You can add a new line to the program by highlighting the previous line and pressing the *AddLine* button again. For local assignment, the program uses *the left arrow*.

Example 1. Figure 1 shows the calculation scheme of epi- and hypocycloid mechanisms. R and r denote the radii of the central wheel and satellite, respectively, the size of the rod is $AB = l$ (letter). Then the projections of the point B of the rod on the x and y axes of the Cartesian coordinate system (as a vector X) will look like:

$$X_0 = (1+k \cdot i) \cdot \cos(\phi) - \lambda \cdot \cos\left(\left(\frac{1+k \cdot i}{i}\right) \cdot \phi + k \cdot \alpha\right) - \text{projection onto the x-axis,}$$

$$X_1 = (1+k \cdot i) \cdot \sin(\phi) - k \cdot \lambda \cdot \sin\left(\left(\frac{1+k \cdot i}{i}\right) \cdot \phi + k \cdot \alpha\right) - \text{projection onto the y-axis,}$$

where the projections of point B are expressed in fractions of the radius R of the central wheel: $i=r/R$; $\lambda=l/R$; $X_0=x/R$; $X_1=y/R$.

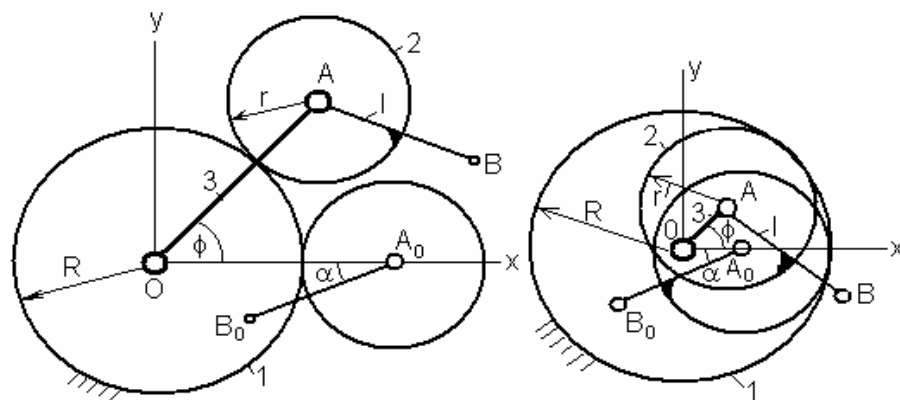


Fig.1. Calculation scheme of epi- and hypocycloid mechanisms

Let the name of the program be S , the formal arguments passed to the program are given in parentheses. The *Addline* button created three lines of the program, which contain:

$$S(k, i, \alpha, \lambda, \varphi) := \left| \begin{array}{l} X_0 \leftarrow (1 + k \cdot i) \cdot \cos(\varphi) - \lambda \cdot \cos\left[\frac{1 + k \cdot i}{i} \cdot \varphi + k \cdot \alpha\right] \\ X_1 \leftarrow (1 + k \cdot i) \cdot \sin(\varphi) - k \cdot \lambda \cdot \sin\left[\frac{1 + k \cdot i}{i} \cdot \varphi + k \cdot \alpha\right] \\ X \end{array} \right.$$

$$F(\varphi) := S(1, .25, 0, 1, \varphi) \quad G(\varphi) := S(-1, .25, 0, 1, \varphi) \quad \varphi := 0, \frac{\pi}{48} .. 2 \cdot \pi$$

Fig.2.

The program S is called by the functions $F(\varphi)$ for epicycloid and $G(\varphi)$ for hypocycloid mechanisms. The trajectory of point B is shown in fig. 3. A circle of radius equal to one (i.e. R) is also shown there.

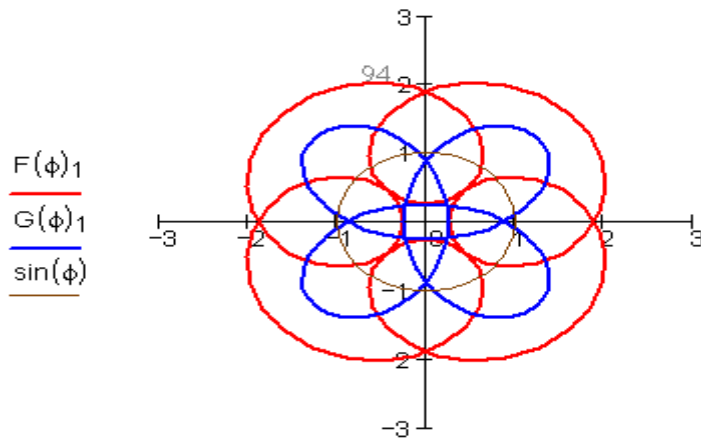


Fig. 3. Point B trajectories

Thus, the AddLine statement creates the first, or new, empty line in the program. You can use the AddLine statement to create a compound statement line in the body of a loop, in a conditional statement, and so on. Remember that the Add Line statement creates a marker after (or before) the highlighted expression.

Local assignment (\leftarrow) assigns the local variable on the left to the value on the right of the arrow. The local variable i has a context (scope of existence) only within the limits of the given program.

Creating Branching Programs

To change the natural order of execution of statements, the if and otherwise control statements called by the buttons on the programming palette are used. Using control statements, you can build branching structures (Fig. 4):

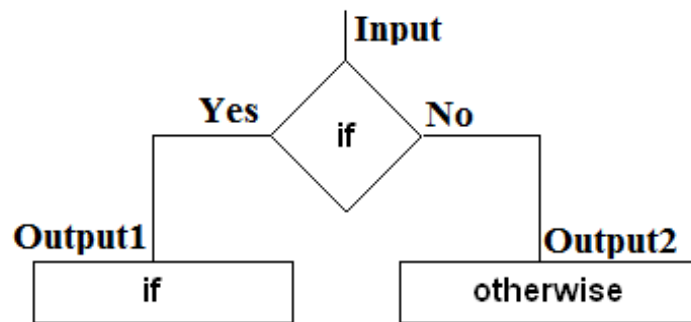


Fig. 4. Block diagram of a branching structure

In earlier versions of Mathcad, the condition was checked using the built-in if function, entered from the keyboard and having the following format: if (condition, expression_Yes, expression_No), Fig.5:

$$f(x) := \text{if}(|x| > 2, 0, \sqrt{4 - x^2})$$

$$f(3) = 0 \quad f(1) = 1.732051$$

Fig. 5.

When developing programs, the keywords if and otherwise are entered only from the programming panel, and not from the keyboard. Format of if and otherwise statements:

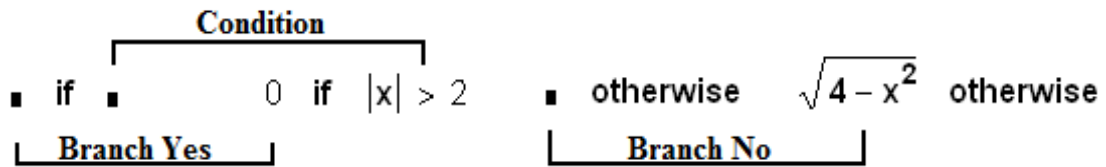


Fig.6. Formats of if and otherwise operators

The condition is written to the marker to the right of the if keyword, and the Yes branch is written to the left (this marker can be multiplied with the Addline key). The otherwise branch is written to the No branch (Fig.6). It can also be propagated with the Addline button. Figure 7 shows a fragment of a program in which a branching structure is implemented:

$$f(x) := \begin{cases} 0 & \text{if } |x| > 2 \\ \sqrt{4-x^2} & \text{otherwise} \end{cases}$$

$f(3) = 0$
 $f(1) = 1.732051$

Fig.7.

The function returns 0 if $|x|$ greater than 2 or $\sqrt{4-x^2}$ otherwise. The number of statements included in each block is indicated by vertical lines. If there is more than one if in the program before the otherwise statement, then the otherwise statement is executed only if all previous if statements are false :

$$f1(x) := \begin{cases} y \leftarrow 0 \\ y \leftarrow 4 + y & \text{if } x > 4 \\ y \leftarrow 3 + y & \text{if } x > 3 \\ y \leftarrow 2 + y & \text{if } x > 2 \\ y \leftarrow 1 + y & \text{otherwise} \\ y \end{cases}$$

Fig.8.

When calling the function, we have: $f1(5) = 9$ and $f1(2) = 1$. There are no "else if" or "case" type operators in Mathcad that allow you to create multi-nested if, but you can combine if ... otherwise.

Example 2. Let's perform the formation of the law of motion of the pusher of the cam mechanism using the `Koef_pol ()` function, which implements the calculation of kinematic coefficients. It is known that during the synthesis of cam mechanisms with an elastic pusher, such laws of its motion are usually selected, in which there are no hard impacts. In this case, it is advisable to use a polynomial of the seventh degree with coefficients written as a vector

b. Let's create a linear program Koef_pol() for calculating displacement coefficients $\zeta(k)$, speed $\delta(k)$ and acceleration $\xi(k)$, in which:

- displacement coefficient $\zeta(k)$ is modeled by the variable K_0 ;
- speed coefficient $\delta(k)$ is modeled by variable K_1 ;
- acceleration factor $\xi(k)$ is modeled by variable K_2 ;

Thus, each element of the returned vector K models a corresponding kinematic coefficient. The input parameters of the polynomial are the coefficient vector b , the constant shift parameter of the degree of the polynomial a , the number of polynomial components n , and the polynomial variable k .

When calculating the coefficients of displacements, velocities and accelerations, the operators of sums and derivatives were used. By changing the heading parameters, it is possible to form various laws of pusher motion. Here is a polynomial that implements the law of motion of the pusher of the cam mechanism without hard and soft shocks:

$$y(k) := 126 \cdot k^5 - 420 \cdot k^6 + 540 \cdot k^7 - 315 \cdot k^8 + 70 \cdot k^9$$

In Fig.9. the calculation program and graphs of the kinematic coefficients obtained for a given polynomial are given.

$$b := (126 \quad -420 \quad 540 \quad -315 \quad 70)^T \quad a := 5 \quad n := 4 \quad k := 0, 0.01.. 1$$

$$\text{Koef_pol}(b, a, n, k) := \begin{cases} K_0 \leftarrow \sum_{i=0}^n b_i \cdot k^{i+a} & S(k) := \text{Koef_pol}(b, a, n, k) \\ K_1 \leftarrow \sum_{i=0}^n b_i \cdot \frac{d}{dk} k^{i+a} \\ K_2 \leftarrow \sum_{i=0}^n b_i \cdot \frac{d^2}{dk^2} k^{i+a} \end{cases}$$

K

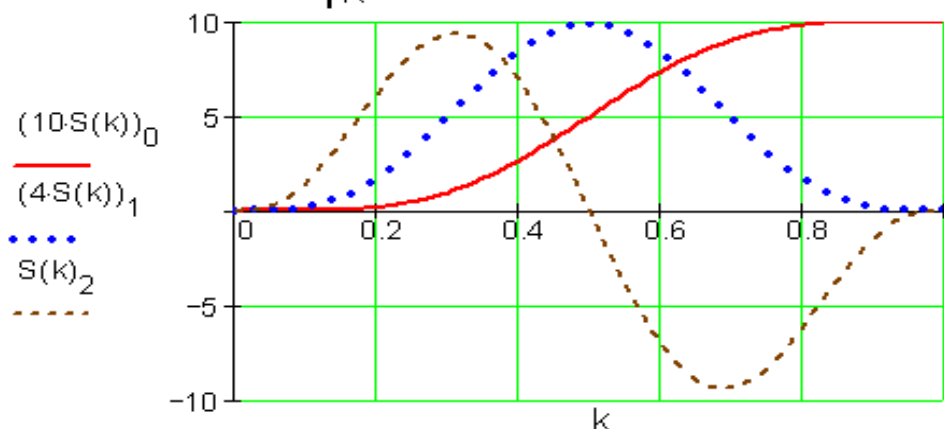


Fig. 9. Kinematic coefficients

Loops with a fixed number of repetitions

One of the ways to create cycles with a fixed number of repetitions is to create a ranked variable x that changes in the interval $x_0 \leq x \leq x_n$ with a step Δx . For this purpose:

- the name of the variable is written chi assignment operator (:=);
- the value x0 is written in the marker and a comma (,) is put;
- the second value of the ranked variable (x+Δx) is written in the new marker. Mathcad subtracts the first value from the second value and automatically generates the step Δx;
- a semicolon (;) is written, which Mathcad converts to a colon (..);
- the value xn is written in the new marker. The ranged variable has been created. For example, let the discrete argument α change in the range $-\pi \leq \alpha \leq \pi$ with a step $\Delta\alpha = \pi/36$. Then you can write:

$$\alpha := -\pi, -\pi + \frac{\pi}{36} .. \pi$$

Fig.10.

If the step is equal to one (Δx=1), then the second value can be omitted.

Thus, the use of ranked variables is a powerful tool in Mathcad, which is much more convenient to organize loops (including nested ones), since they are the basis for the main principles of calculations in Mathcad, in particular, the preparation of graphs.

When programming in Mathcad, loops with a fixed number of repetitions are implemented by the for operator, the template of which is shown in Fig.11:

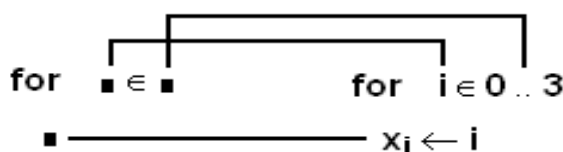


Fig.11. Template format of For

It has three markers - two of them on the for line to write the name of the loop counter and the discrete argument, respectively. The third marker contains the loop body. By selecting it and pressing the AddLine button, the number of cycle body lines can be increased (in this case, the cycle body is marked with a vertical line).

Task 1. Write a program for finding the values of the Y function for any values of the initial parameters in Mathcad

Table 2

№	Function
1	$\begin{cases} m^2n+1-c, & \text{if } n+1 > 0 \\ (m+n)^2+cm^2, & \text{if } n+1 \leq 0 \end{cases}$
2	$\begin{cases} \frac{1}{\sqrt{9+a^2}}, & \text{if } a < 5 \\ b*\sin a, & \text{if } a \geq 5 \end{cases}$
3	$7x^2-3abx-5ab, \text{ if } a > 0$

	$15a-7b$, if $a \leq 0$
4	$\left\{ \begin{array}{l} \frac{a^2+b^2}{c} + \sqrt{a+x}, \text{ if } x \geq 0 \\ \frac{\sin x + a}{a-b}, \text{ if } x < 0 \end{array} \right.$
5	$\left\{ \begin{array}{l} (nm^2+d)^2 \text{ if } d < m \\ n^2+m^2, \text{ if } d \leq m \end{array} \right.$
6	$\left\{ \begin{array}{l} \frac{ax^2}{e^x} + 5, \text{ if } a < 9 \\ (a+1)^2+cx^3, \text{ if } a \geq 9 \end{array} \right.$
7	$\left\{ \begin{array}{l} \frac{a^3}{3+ac}, \text{ if } a > 0 \\ \sqrt{\frac{ac+2b}{1-a}}, \text{ if } a \leq 0 \end{array} \right.$
8	$\left\{ \begin{array}{l} \sqrt{\frac{x}{1+x}}, \text{ if } 1 \geq x > 0 \\ \frac{\ln(x-1)}{x-a}, \text{ if } x > 1 \end{array} \right.$
9	$\left\{ \begin{array}{l} (1/3) \cdot \ln^3 x, \text{ if } x > 0 \\ (x+3,5)e^x, \text{ if } x \leq 0 \end{array} \right.$
10	$\left\{ \begin{array}{l} \text{Ln}(x+\sqrt{x^2+9}), \text{ if } x \geq 2 \\ \frac{\ln 3}{9x}, \text{ if } x < 2 \end{array} \right.$
11	$\left\{ \begin{array}{l} \frac{a}{a^2+x}, \text{ if } a > 3 \\ 2a^3+\sin^2 x, \text{ if } a \leq 3 \end{array} \right.$
12	$\left\{ \begin{array}{l} \sqrt{a^2 k}, \text{ if } k \geq 3 \\ e^{-k}(1+\text{tg } a \cdot k), \text{ if } k < 3 \end{array} \right.$
13	$\left\{ \begin{array}{l} \frac{1+x}{x^2 - \sqrt[3]{x}}, \text{ if } x < 0,3 \\ \cos^2 2x - e^x, \text{ если } x \geq 0,3 \end{array} \right.$
14	$\left\{ \begin{array}{l} x^2-4+\sqrt{a+x}, \text{ if } x > 4a \\ 1/2 \cdot (5x^2-3x), \text{ if } x \leq 4a \end{array} \right.$
15	$\left\{ \begin{array}{l} 1/2 \cdot (3x^2-ax), \text{ if } x < 10 \\ x^3-5ae^x, \text{ if } x \geq 10 \end{array} \right.$
16	$\left\{ \begin{array}{l} \sqrt{15a^2+17b^2}, \text{ if } a > b \\ \sqrt{17a^2+15b^2}, \text{ if } a \leq b \end{array} \right.$
17	$\left\{ \begin{array}{l} -\ln 2x-3x^2 , \text{ if } x < 5z \end{array} \right.$

	$\ln 2x - 3x^2 $, if $x \geq 5z$
18	$\left\{ \begin{array}{l} \text{Sin}(1+km), \text{ if } km < 2 \\ \text{Ln}(5+k/m), \text{ if } km \geq 2 \end{array} \right.$
19	$\left\{ \begin{array}{l} \sqrt{ 2k_1 - 7k_2 }, \text{ if } k_1 * k_2 < 1 \\ \sqrt[3]{2k_1 + 7k_2}, \text{ if } k_1 * k_2 \geq 1 \end{array} \right.$
20	$\left\{ \begin{array}{l} \frac{4r + 3m^2}{r - m}, \text{ if } r \geq m+1 \\ r - m , \text{ if } r < m+1 \end{array} \right.$
21	$\left\{ \begin{array}{l} \sqrt{3x^2 + 4z^2}, \text{ if } z \leq 2 x \\ \sqrt[3]{3x^2 - 4z^2}, \text{ если } z \leq 2 x \end{array} \right.$
22	$\left\{ \begin{array}{l} \frac{x - 2t}{2x + 5t}, \text{ if } x * t < 0 \\ \sqrt{xt}, \text{ if } x * t \geq 0 \end{array} \right.$
23	$\left\{ \begin{array}{l} \text{Cos}^2(x-2t), \text{ if } x < 2,5 \\ \text{Ln}(x-2t), \text{ if } x \geq 2,5 \end{array} \right.$
24	$\left\{ \begin{array}{l} \text{Sin } \pi x + e^{-ax}, \text{ if } x \leq 3 \\ \text{tg } \pi x + ax^2, \text{ if } x > 3 \end{array} \right.$
25	$\left\{ \begin{array}{l} \text{arctg}(x^2+3t), \text{ if } x^2+3t > 1 \\ \text{arccos}(x^2+3t), \text{ if } x^2+3t \leq 1 \end{array} \right.$

Task 2. Write a program to find the values of the functions f1(x) and f2(x) in the interval [0.1] on Mathcad with a change step $\Delta x=0.1$.

Table 3

№	Function f₁(x)	Function f₂(x)
1	$\text{Sin}(x+4.5)$	$20 / (1+x^2)$
2	$1+\text{cos}(x-2)$	$\sqrt{x^2 + 1}$
3	$e^x + \text{Sin}(x)$	$1+2^x$
4	$ \text{Sin}(x)+1 $	$(1+2x)e^x$
5	$2-\text{cos}(x+1)$	$1-x^2$
6	$\sqrt{2x} + x^2 + 1$	$e^x(1+\text{cos} \frac{x}{2})$
7	$1+2^{x+1}$	$4 * e^{1-x} - 1$
8	$1+1,8^{x+1}$	$\text{Cos}^2(x-1)$
9	x^2+x+1	$\sqrt{1+x} \ln(x+1)$
10	$\text{Cos}(2x+1)$	$x * e^{x+1}$
11	$2-\text{cos}^3(x^6+1)$	$\sqrt{x^2 + 3}$

12	$\sqrt{3x^2+4}$	$\ln(x^2+x+1)$
13	$x(x^2+1)$	$e^{- x+1 }*(1+x)$
14	$1+\ln(2x-1)$	$\sin^2(x+1)*e^x$
15	$e^x/(1+x)$	$(x-1)^2$
16	$(1+x)*e^{-x}$	$x^2+\sin(x)$
17	$1+3^{x+1}$	$(1-x)*\text{tg}^2(x)$
18	$\lg(1+x)$	$1+\cos^2(x)$
19	$\sqrt{x^3+1}*e^{-x}$	$1+2\sin x$
20	$\sqrt{2x-1}$	$\text{Ln}(\cos(x)+1)$
21	$1+0,03x^2$	$\text{Cos}x+\sin x$
22	$x-\sin(2x-1)$	$\text{Ln}(1+2^x)$
23	$0,02*e^{2x+1}$	$\text{Cos}(2x-1)$
24	$\sqrt{x^2+0,09}x$	$\text{tg}(3x-1)$
25	$10^{-6}*\ln(x+1)$	x^2+2x-3
26	$\text{Cos}x+\sin x$	$1+\ln(2x-1)$
27	$4*e^{1-x}-1$	$2-\cos(x+1)$
28	$e^{- x+1 }*(1+x)$	$2-\cos^3(x^3+1)$
29	$\ln(x^2+x+1)$	$ \sin(x)+1 $
30	$\sqrt{1+x}\ln(x+1)$	$\ln 2x-3x^2 $

Control questions:

1. How to enter a function program in Mathcad?
2. What are the formal parameters of the program - functions?
3. How is the call to the program - functions in Mathcad?
4. What is a conditional operator in Mathcad and how to use it?
5. What is an arithmetic progression loop in Mathcad and how to use it?

Laboratory work No3

Representation of data by matrices. Working with vectors and matrices in MatLab.

The purpose of the lesson: to study the implementation of the basic operations with vectors and matrices by means of the MATLAB system.

Theoretical part

By default, all numeric variables in MATLAB are considered matrices, so a scalar is a first-order matrix and vectors are single-column or single-row matrices. A matrix can be entered by specifying its elements or by reading data from a file, or by calling a standard or user-written function.

Matrix data is placed in memory sequentially by columns. Matrix elements within a row are separated by spaces or commas. The direct definition of the matrix can be done in several ways. For example, a column vector, that is, a matrix whose second dimension is equal to one, can be assigned to variable *A* by entering one line:

```
>> A=[7+4i; 4; 3.2]           % Column vector input
```

```
A =
  7.0000 + 4.0000i
  4.0000
  3.2000
```

or by entering multiple rows

```
>> A = [           % Entering a vector by rows
  7+4i
  4
  3.2];
```

Vectors can be formed as ranges using colons separating start value, step value, and limit value. If there is no step value, then its default value is one.

As a result, *n:m:k* will form a vector, the last element of which is no more than *k* for a positive step *m*, and no less than for a negative one: $[n, n+m, n+m+m, \dots]$

For example

```
>> a=1:2:5
a =
  1  3  5
```

Specifying a range is also used when organizing a cycle. Table 1 presents a set of functions for creating special-type matrices.

Table 1

Matrix description functions

Function	Description
eye(m,n)	Identity matrix of $m \times n$ dimensions
zeros(m,n)	Zero matrix of dimension $m \times n$
ones(m,n)	Matrix consisting of one unit of dimension $m \times n$
rand(m,n)	Returns a matrix of random numbers uniformly distributed in the range from 0 to 1, dimension $m \times n$
randn(m, n)	Returns an $m \times n$ matrix consisting of random numbers having a Gaussian distribution
tril(A), triu(A)	Selection of the lower triangular and upper triangular parts of the matrix A
inv(A)	Finding the inverse matrix A

The matrix element is accessed according to the rule - in parentheses after the name of the matrix, indices are given, which must be positive integers indicating the row number and, separated by a comma, the column number. For example, A(2,1) means the element from the second row of the first column of matrix A.

For further examples, let's introduce a 2x2 matrix:

```
>> A=[1 2+5*i; 4.6 3]
```

```
A =
```

```
1.0000    2.0000 + 5.0000i
```

```
4.6000    3.0000
```

To change a matrix element, you need to assign a new value to it

```
>> A(2,2)=10    % The second element of the second row
```

```
A =
```

```
1.0000    2.0000 + 5.0000i
```

```
4.6000    10.0000
```

The size of a matrix can be specified with the size command, and the result of the size command can be used to organize a new matrix.

For example, a zero matrix of the same order as matrix A will be formed by the command

```
>> A2=zeros(size(A))
```

```
A2 =
```

```
0  0
```

```
0  0
```

Using a colon, it is easy to select a part of the matrix. For example, a vector of the first two elements of the second column of matrix A is given by:

```
>> A(1:2, 2)
```

```
ans =
```

```
2.0000 + 5.0000i
```

```
10.0000
```

A colon by itself means the entire row or column. To remove a vector element, it is enough to assign an empty array to it - a pair of square brackets []. To cross out one or more rows (columns) of a matrix, you need to specify the range of rows (columns) to be deleted for one dimension and put a colon for another dimension. You can also use **the length** command to find the length of a vector.

The set of arithmetic operations in MATLAB for working with matrices consists of standard operations of addition - subtraction, multiplication - division, exponentiation

operations and are supplemented with special matrix operations (Table 6). If the operation is applied to matrices whose sizes are inconsistent, an error message will be displayed.

For element-by-element execution of operations of multiplication, division and exponentiation, combined signs (dot and sign of the operation) are used. For example, if a matrix is followed by a (^) sign, then it is raised to a power, and the combination (.^) means that each element of the matrix is raised to a power. When multiplying (addition, subtraction, division) of a matrix by a number, the corresponding operation is always performed element by element.

Table 2

Operation signs

Symbol	Purpose
+,-	The plus and minus symbols indicate the sign of a number or the operation of addition and subtraction of matrices, and the matrices must be of the same dimension
*	The multiplication sign denotes matrix multiplication, for element-wise matrix multiplication, the combined sign (.*) is used
'	An apostrophe denotes a transposition operation (together with complex conjugation), transposition without conjugation calculation is denoted by a combined sign (.)'
/	Left division
\	right division
^	The exponentiation operator, for element-wise exponentiation, the combined sign (.^) is used

We illustrate the difference between ordinary and element-wise multiplication using the following example.

Let's introduce a 2x2 matrix H and a matrix D of units of the same dimension:

```
>> H=[0 1; 2 3], D=ones(size(H))
```

H =

```
0 1
2 3
```

D =

```
1 1
1 1
```

Let's multiply the matrices using the usual multiplication:

```
>> H*D
```

ans =

```
1 1
```

5 5

Now we apply the element-by-element operation:

```
>> H.*D
```

```
ans =
```

```
0 1  
2 3
```

Matrix Functions

The MATLAB system has a number of functions designed to process data given in matrix or vector form (Table 3).

Table 3

Function	Description
size(A)	Returns an array consisting of the number of rows and the number of columns of a matrix.
sum(A)	Returns the sum of all elements by column
mean(A)	Returns the average value of a matrix column
std(A)	Returns the standard deviation of a matrix column
min(A), max(A)	Returns the minimum and maximum respectively, by column of the matrix
sort(A)	Sorts a matrix column in ascending order
prod(A)	Calculates the product of all column elements

Task 1:

Exercise 1. Ouptut:

- arbitrary row vector (v), dimension 2;
- arbitrary column vector (w), dimension 2;
- an arbitrary matrix (m), dimensions 2×2 .

Exercise 2 Create:

- a matrix with zero elements ($m0$), dimensions 2×2 ;
- a matrix with unit elements ($m1$), dimensions 2×2 ;
- matrix with elements having random values (mr), dimension 2×2 ;
- a matrix with unit diagonal elements (me), of dimension 2×2 .

Exercise 3. Calculation of the matrix M using the formula presented in table 8

Exercise 4: Exploring Data Processing Functions:

- determination of the number of rows and columns of the matrix M ;
- determination of the maximum element of the matrix M ;
- determination of the minimum element of the matrix M ;
- summation of the elements of the matrix M ;
- multiplication of the elements of the matrix M .

Task variants

Table 4

№ of variant	Task	№ of variant	Task
1	$M=v*w+m+mr*me$	11	$M=m*w+mr*v'$
2	$M=m+mr*me$	12	$M=m*mr+w*v$
3	$M=(v/m)*(mr+me)$	13	$M=m+mr-100$
4	$M=w*v+mr*me$	14	$M=v'+w+mr*w$
5	$M=m*mr+me$	15	$M=m+m1'*me'$
6	$M=m.*mr+100$	16	$M=(v/m)*(mr+me)$
7	$M=v*w+mr-m$	17	$M=v*mr+v*m1$
8	$M=m+mr*me-10$	18	$M=m'+mr/100$
9	$M=m*w+mr*v'$	19	$M=10*v+w'*mr*m$
10	$M=m'+mr*me$	20	$M=m'+mr*me$

Task 2

Exercise 5. Actions on Matrix

- Transpose matrix **A**;
- Calculate the inverse of matrix **B**;
- Operations on matrices **A*B**; **A+B**; **A-B** **A/B**;
- Merge matrices vertically and horizontally

Task variants

1. $A = \begin{pmatrix} 2 & -1 & -3 \\ 8 & -7 & -6 \\ -3 & 4 & 2 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 3 & -5 & 4 \\ 1 & 2 & 1 \end{pmatrix}$

2. $A = \begin{pmatrix} 7 & -1 & 12 \\ 5 & -7 & -6 \\ 3 & 4 & 10 \end{pmatrix}, B = \begin{pmatrix} 9 & -1 & 2 \\ 3 & 11 & 4 \\ 4 & 2 & -8 \end{pmatrix}$

3. $A = \begin{pmatrix} 2 & -1 & 6 \\ 4 & -7 & -6 \\ 9 & -11 & 2 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 14 & 5 & -4 \\ 16 & 2 & 1 \end{pmatrix}$

4. $A = \begin{pmatrix} 9 & -1 & -3 \\ 8 & -17 & -6 \\ 6 & 4 & 12 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 23 & -5 & 4 \\ 1 & 25 & 1 \end{pmatrix}$

$$5. \quad A = \begin{pmatrix} 7 & -2 & -7 \\ -4 & 6 & -9 \\ 3 & 9 & 25 \end{pmatrix}, B = \begin{pmatrix} -11 & 14 & 7 \\ 14 & -12 & 2 \\ 15 & 0 & 3 \end{pmatrix}$$

$$6. \quad A = \begin{pmatrix} 8 & -9 & 3 \\ 5 & 6 & 2 \\ 2 & -7 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 & 2 & 8 \\ 1 & 4 & 5 \\ 1 & 7 & 2 \end{pmatrix}$$

$$7. \quad A = \begin{pmatrix} -12 & 23 & 14 \\ 14 & 24 & 17 \\ -17 & 25 & 18 \end{pmatrix}, B = \begin{pmatrix} -12 & -11 & 0 \\ -14 & -18 & 1 \\ -17 & -21 & -9 \end{pmatrix}$$

$$8. \quad A = \begin{pmatrix} -10 & 25 & 3 \\ 14 & 24 & 2 \\ 12 & 7 & 1 \end{pmatrix}, B = \begin{pmatrix} -9 & 1 & 7 \\ -6 & -8 & 4 \\ -3 & 4 & 1 \end{pmatrix}$$

$$9. \quad A = \begin{pmatrix} 25 & -32 & 2 \\ 28 & -39 & 1 \\ 29 & 37 & 5 \end{pmatrix}, B = \begin{pmatrix} 4 & 8 & 23 \\ 3 & 9 & 45 \\ 2 & -66 & 78 \end{pmatrix}$$

$$10. \quad A = \begin{pmatrix} 2 & -1 & -3 \\ 8 & -7 & -6 \\ -3 & 4 & 2 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 23 & -5 & 4 \\ 1 & 25 & 1 \end{pmatrix}$$

$$11. \quad A = \begin{pmatrix} 7 & -2 & -7 \\ -4 & 6 & -9 \\ 3 & 9 & 25 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 3 & -5 & 4 \\ 1 & 2 & 1 \end{pmatrix}$$

$$12. \quad A = \begin{pmatrix} -12 & 23 & 14 \\ 14 & 24 & 17 \\ -17 & 25 & 18 \end{pmatrix}, B = \begin{pmatrix} -12 & -11 & 0 \\ -14 & -18 & 1 \\ -17 & -21 & -9 \end{pmatrix}$$

$$13. \quad A = \begin{pmatrix} 9 & -1 & -3 \\ 8 & -17 & -6 \\ 6 & 4 & 12 \end{pmatrix}, B = \begin{pmatrix} 4 & 8 & 23 \\ 3 & 9 & 45 \\ 2 & -66 & 78 \end{pmatrix}$$

$$14. \quad A = \begin{pmatrix} 7 & -1 & 12 \\ 5 & -7 & -6 \\ 3 & 4 & 10 \end{pmatrix}, B = \begin{pmatrix} 0 & 2 & 8 \\ 1 & 4 & 5 \\ 1 & 7 & 2 \end{pmatrix}$$

$$15. \quad A = \begin{pmatrix} 7 & -2 & -7 \\ -4 & 6 & -9 \\ 3 & 9 & 25 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 3 & -5 & 4 \\ 1 & 2 & 1 \end{pmatrix}$$

$$16. \quad A = \begin{pmatrix} -12 & 23 & 14 \\ 14 & 24 & 17 \\ -17 & 25 & 18 \end{pmatrix}, B = \begin{pmatrix} -12 & -11 & 0 \\ -14 & -18 & 1 \\ -17 & -21 & -9 \end{pmatrix}$$

$$17. \quad A = \begin{pmatrix} 2 & -1 & -3 \\ 8 & -7 & -6 \\ -3 & 4 & 2 \end{pmatrix}, B = \begin{pmatrix} 2 & -1 & 2 \\ 3 & -5 & 4 \\ 1 & 2 & 1 \end{pmatrix}$$

$$18. \quad A = \begin{pmatrix} 7 & -2 & -7 \\ -4 & 6 & -9 \\ 3 & 9 & 25 \end{pmatrix}, B = \begin{pmatrix} -11 & 14 & 7 \\ 14 & -12 & 2 \\ 15 & 0 & 3 \end{pmatrix}$$

$$19. \quad A = \begin{pmatrix} -12 & 23 & 14 \\ 14 & 24 & 17 \\ -17 & 25 & 18 \end{pmatrix}, B = \begin{pmatrix} -12 & -11 & 0 \\ -14 & -18 & 1 \\ -17 & -21 & -9 \end{pmatrix}$$

$$20. \quad A = \begin{pmatrix} 9 & -1 & -3 \\ 8 & -17 & -6 \\ 6 & 4 & 12 \end{pmatrix}, B = \begin{pmatrix} 4 & 8 & 23 \\ 3 & 9 & 45 \\ 2 & -66 & 78 \end{pmatrix}$$

Control questions

1. How is the input of a row vector carried out?
2. How is the input of a column vector carried out?
3. How is the matrix entered?
4. What are the zeros, ones, rand, eye commands for?
5. How is the number of rows and columns of a matrix determined?
6. What operations are used to determine the minimum and maximum matrix elements?

Laboratory work No 4

Development and analysis of simulation models of technical objects. Simulation modeling using the Simulink package

The purpose of the lesson: To get acquainted with the capabilities of the Simulink mathematical modeling package.

Theoretical part

The Simulink program is an attachment to the MATLAB package. When modeling using Simulink, the principle of visual programming is implemented, according to which, the user creates a device model on the screen from the library of building blocks and performs calculations. When working with Simulink, the user has the ability to upgrade library blocks, create their own, and create new block libraries. To run the program, you must first run the MATLAB package. The main window of the MATLAB package is shown in Figure 1.

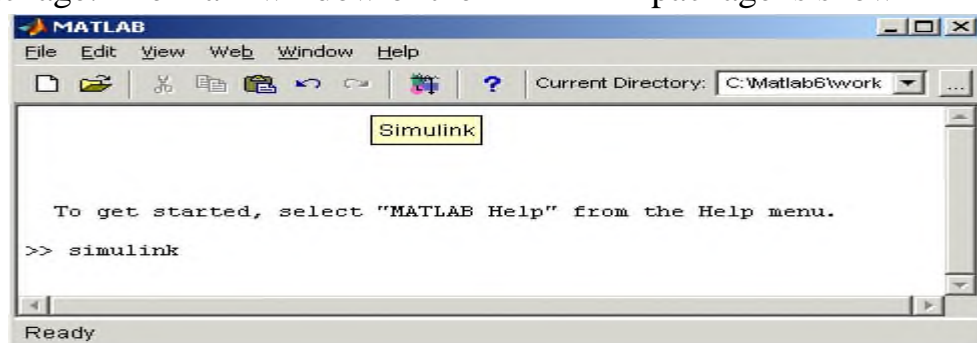


Fig.1. – The main window of the MATLAB program

After opening the main window of the MATLAB program, you need to start the Simulink program. This can be done in one of three ways:

1. Click the (Simulink) button on the toolbar of the MATLAB command window.
2. On the command line of the main MATLAB window, type Simulink and press the Enter key on the keyboard.
3. Run the Open command in the **File** menu and open the model file (mdl - file).

The latter option is convenient to use to run an already finished and debugged model, when you only need to carry out calculations and do not need to add new blocks to the model.

Using the first and second methods leads to the opening of the Simulink library section browser window (Fig. 2).

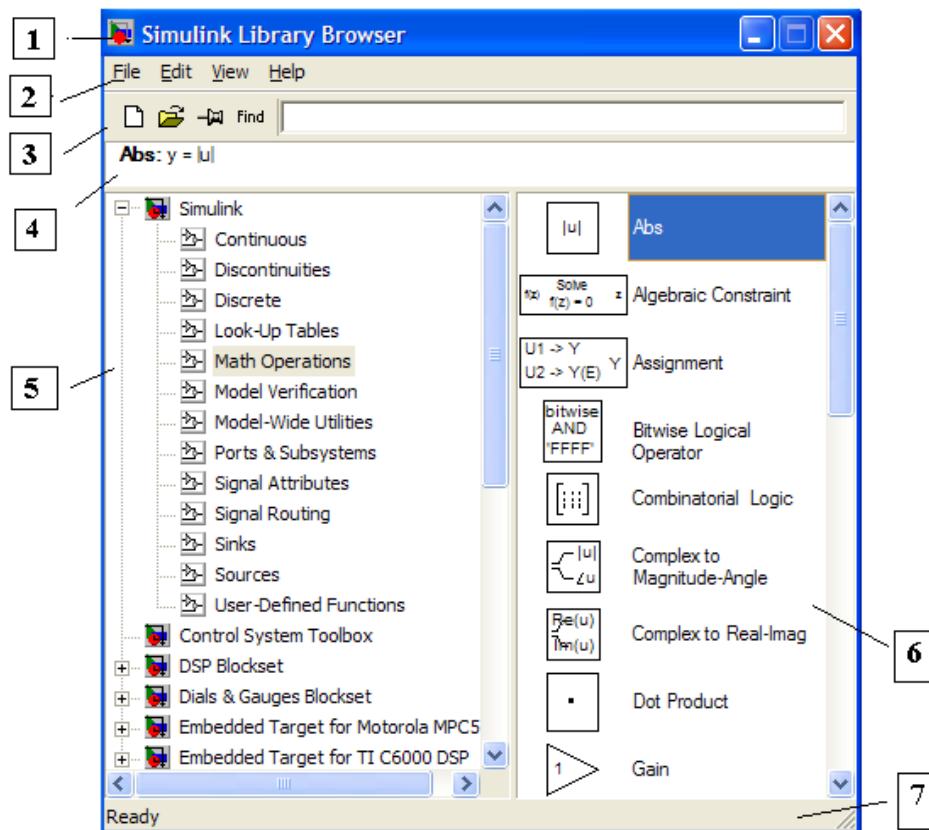


Fig.2. – Simulink Library Section Browser Window

The block library browser window contains the following elements:

- 1) Title, with the name of the window - **Simulink Library Browser**.
- 2) Menu, with commands **File, Edit, View, Help**.
- 3) Toolbar, with shortcuts to the most frequently used commands.
- 4) Comment window for displaying an explanatory message about the selected block.
- 5) List of sections of the library, implemented as a tree.
- 6) Library section content window (list of nested library sections or blocks)
- 7) Status bar containing a hint on the action to be performed.

Figure 2 highlights the main Simulink library (on the left side of the window) and shows its sections (on the right side of the window).

The Simulink library contains the following main sections:

1. **Continuous** - linear blocks.
2. **Discrete** - discrete blocks.
3. **User-Defined Functions** - functions and tables.
4. **Math Operations** - blocks of mathematical operations.
5. **Discontinuities** - non-linear blocks.
6. **Signals Attribute, Signals Routing** - signals and systems.
7. **Sinks** - recording devices.
8. **Sources** - sources of signals and influences.
9. **Ports & Subsystems** - ports and blocks of subsystems.

The list of sections of the Simulink library is presented as a tree, and the rules for working with it are common for lists of this type:

- The icon of a collapsed tree node contains a "+" symbol, and the icon of an expanded tree node contains a "-" symbol.
- In order to expand or collapse a tree node, just click on its icon with the left mouse button.

When you select the appropriate section of the library, its contents are displayed in the right part of the window (Fig. 3).

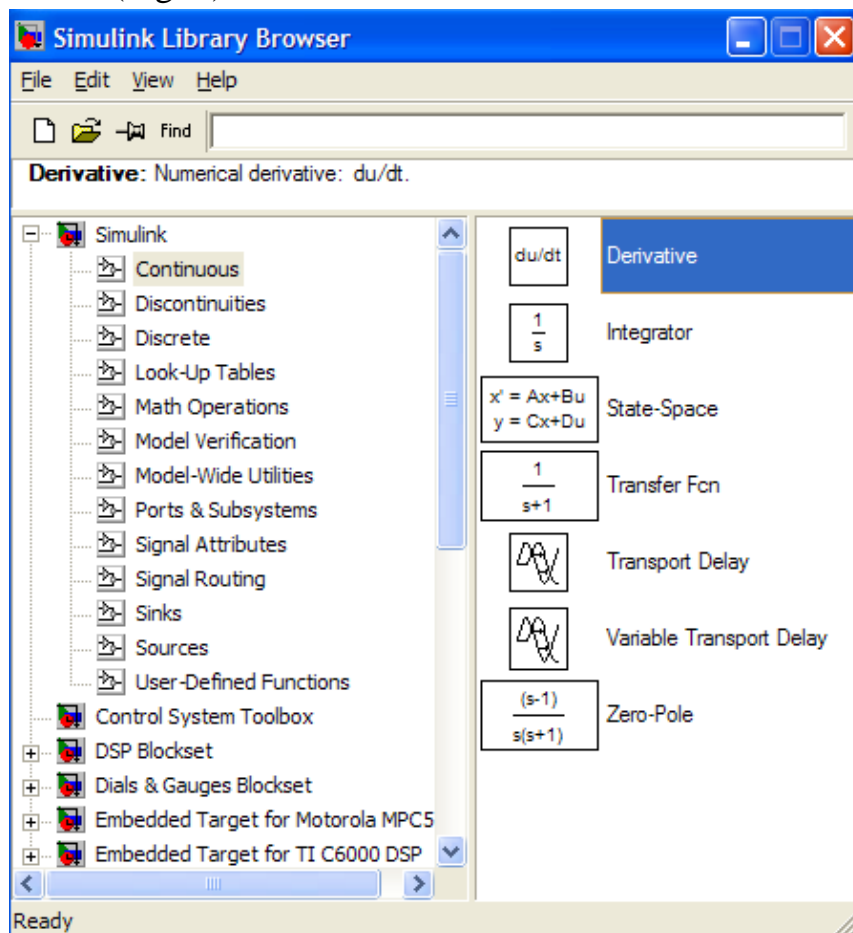


Fig.3.Contents of the **Continuous** section

To create a model in the Simulink environment, you must perform a series of steps in sequence:

1. Create a new model file using the **File/New/Model** command. The newly created model window is shown in Figure4.

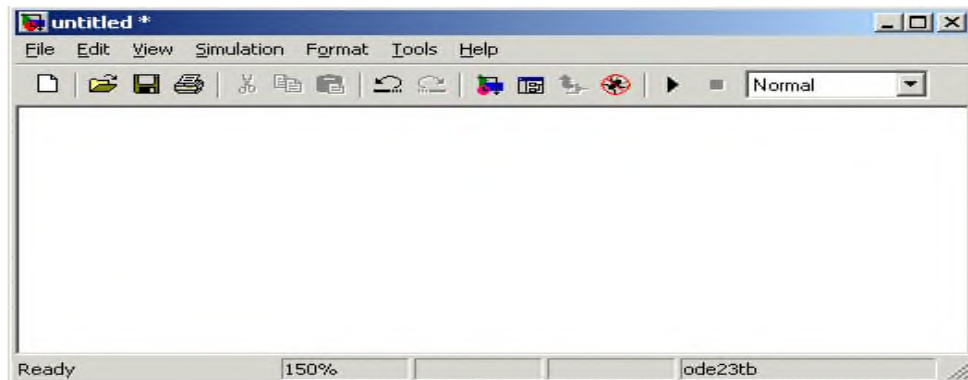


Fig.4. – Empty model window

2. Arrange blocks in the model window. To do this, open the appropriate section of the library (For example, **Sources - Sources**). Further, pointing the cursor at the required block and pressing the left mouse button - “drag” the block into the created window. **The mouse button must be kept pressed.** Figure 5 shows a model window containing blocks.

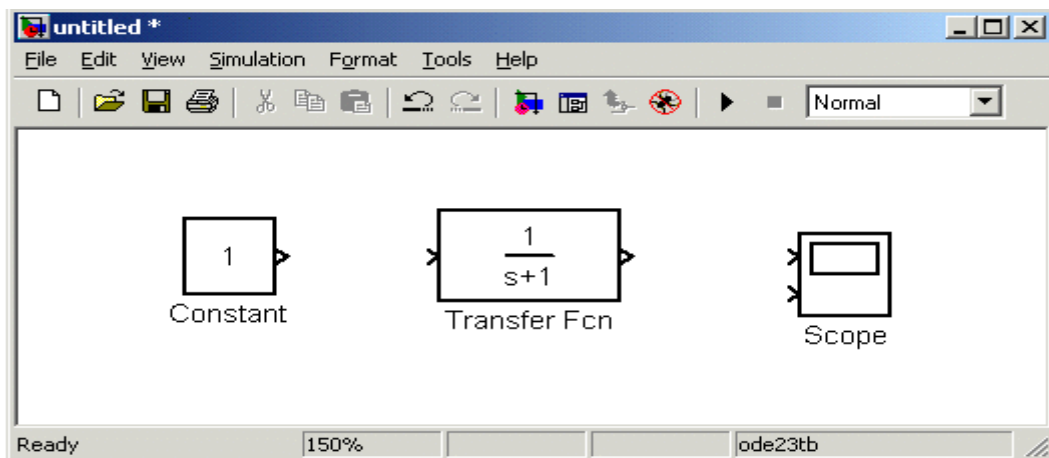


Fig. 5 - Model window containing blocks

To change the parameters of the block, double-click the left mouse button, pointing the cursor at the image of the block. A window for editing the parameters of this block will open. When specifying numeric parameters, keep in mind that the decimal separator must be a dot, not a comma. After making changes, close the window with the **OK** button.

After installing all the blocks from the required libraries on the diagram, you need to connect the elements of the circuit. To connect the blocks, you need to point the cursor at the “output” of the block, and then, press and, without releasing the left mouse button, draw a line to the input of another block. Then release the key. In case of a correct connection, the image of the arrow at the input of the block changes color. To create a branching point

in a connecting line, you need to move the cursor to the proposed node and, by pressing the right mouse button, drag the line. The diagram of the model, in which the connections between the blocks are made, is shown in Figure 6

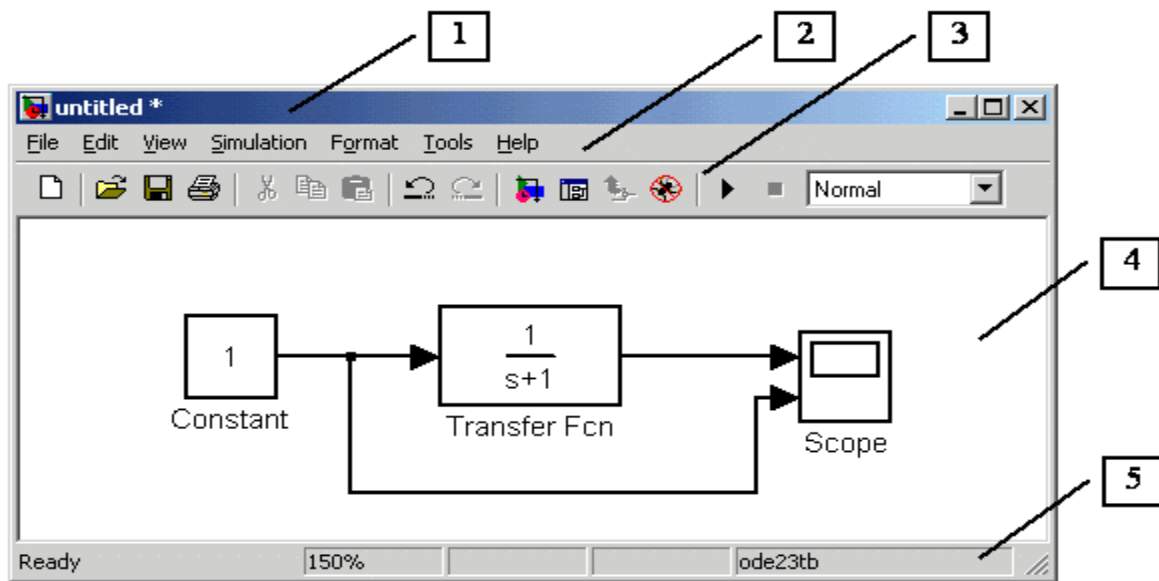


Fig.6. –The diagram of the model

The model window contains the following elements (see Figure 6):

1. Title, with the name of the window. The newly created window is given the name Untitled with the corresponding number.

2. Menu with commands **File**, **Edit**, **View**, etc.

3. Toolbar.

4. Window for creating a model scheme.

5. Status bar containing information about the current state of the model.

The window menu contains commands for editing the model, setting it up and managing the calculation process, working with files, etc.:

a) **File** - Work with model files.

b) **Edit** - Change the model and search for blocks.

c) **View** - Controls the display of interface elements.

d) **Simulation** - Specify settings for simulation and control the calculation process.

e) **Format (Formatting)** - Change the appearance of the blocks and the model as a whole.

f) **Tools** - The use of special tools for working with the model (debugger, linear analysis, etc.)

g) **Help** - Display help windows.

To increase the visibility of the model, it is convenient to use text labels. To create an inscription, you need to specify the location of the inscription with the mouse and double-click the left mouse button. After that, a rectangular frame with an input cursor will appear. In a similar way, you can change the captions for model blocks. It should be borne in mind

that the version of the program under consideration (Simulink 4) is not adapted to the use of Cyrillic fonts, and their use can have a variety of consequences: displaying labels in an unreadable form, cropping labels, error messages, and also the inability to open the model after saving it. . Therefore, the use of inscriptions in Russian for the current version of Simulink is highly undesirable.

Before performing calculations, you must first set the calculation parameters. The calculation parameters are set in the control panel of the **Simulation/Parameters** menu. The view of the control panel is shown in Figure 7

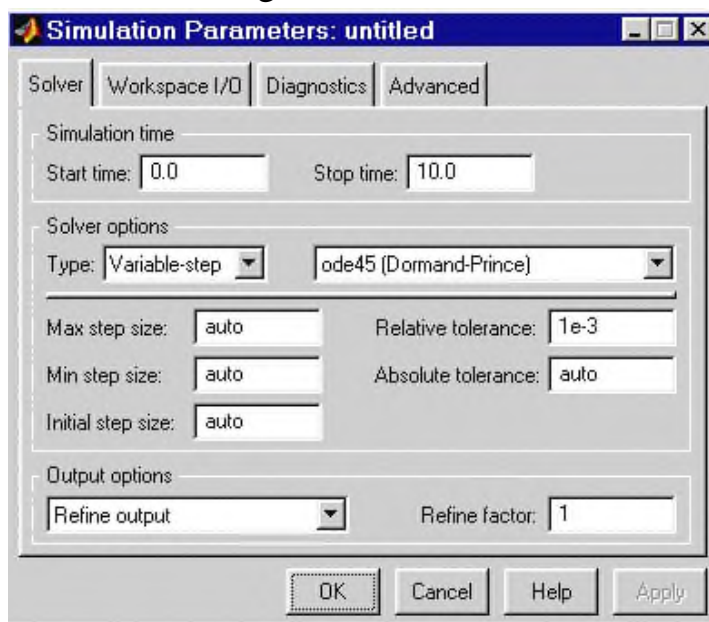


Fig.7. – Control panel

The calculation parameters settings window has 4 tabs:

- **Solver** (Calculation) - Sets the parameters for calculating the model.
- **Workspace I / O** (Data input / output to the workspace) - Setting the parameters for exchanging data with the MATLAB workspace.
- **Diagnostics** - Select diagnostic mode options.
- **Advanced** - Set advanced options.

To visualize the modeling process, virtual registrars (blocks of recipients of information **Sinks**) are used.

Each recorder has its own settings window, which appears when its icon is activated in the components window or in the model window (Fig. 8).

A virtual oscilloscope (**Scope**) allows you to present the simulation results in the form of time diagrams of certain processes in a form that resembles the oscillograms of a modern high-precision oscilloscope with a digitized scale grid made by rays of different colors.

Here, special attention should be paid to the zoom buttons, which allow (along with the context menu commands) to change the size of the waveform. The **Autoscale** button is very convenient - usually it allows you to set a scale at which the waveform image has the maximum possible vertical size and reflects the entire simulation time interval (Figure 9).

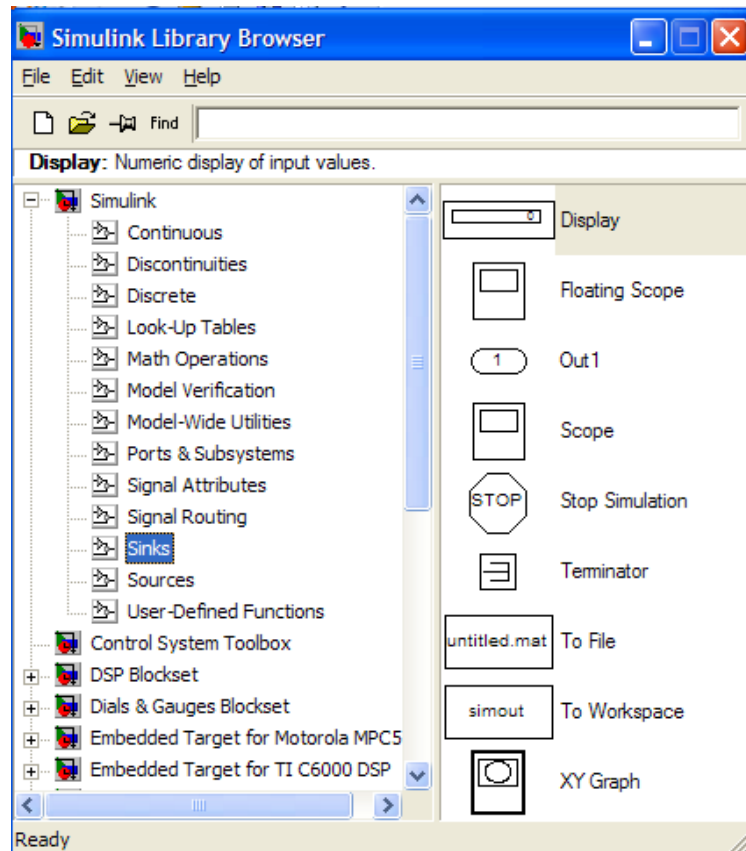


Fig.8. – Contents of the **Sinks** section

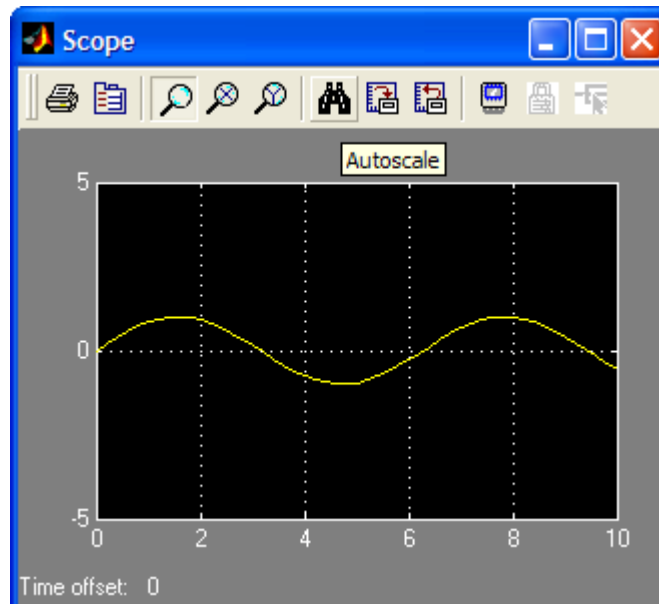


Fig.9. – Window of the virtual oscilloscope

The virtual graph plotter (**XY Graph**) has inputs along the X and Y axes, which allows you to plot functions in the polar coordinate system, Lissajous figures, phase portraits, etc. (Figure 10).

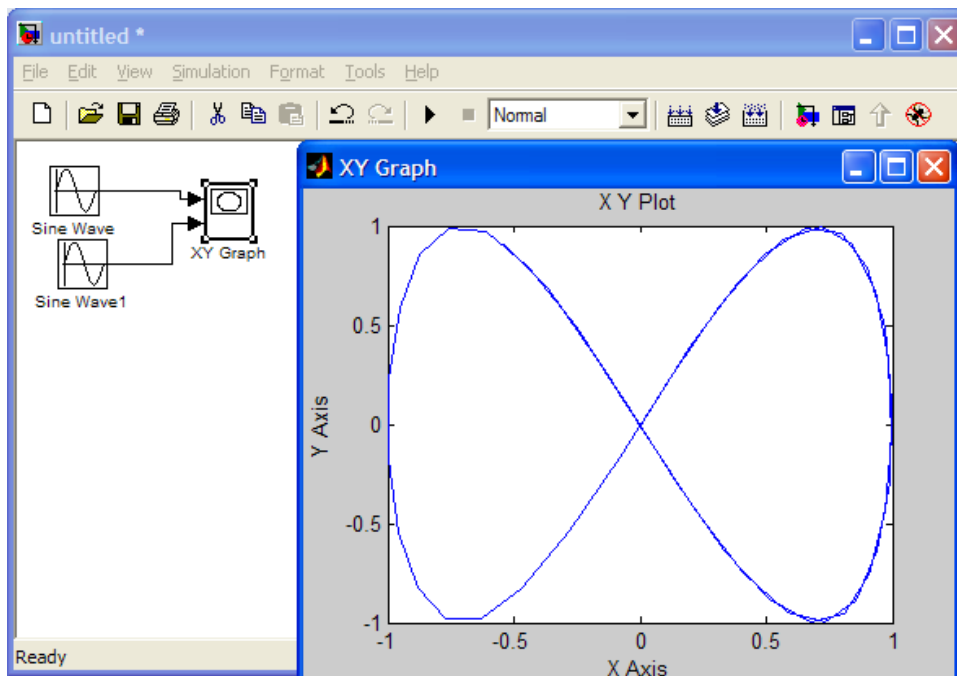


Fig. 10 – Lissajous Figure Chart

To perform arithmetic operations, use the **Math Operations** tab (Figure 11). The simplest mathematical blocks include blocks of arithmetic operations: calculating the absolute value of the number **Abs**, the scalar product **Dot Product**, the usual product **Product**, and the sum **Sum**.

Please note that in the add/subtract block settings window, you can set the type of block representation (round or square) and the number of inputs with operations performed on them. The number of entries and operations are specified by the **List of sign** template. For example, the pattern **++** means that the block has two summing inputs, and **|+--** means that it has three inputs, with the middle one being subtractive, and the extreme ones summing (Figure 12).

Block **Product** (Multiplication) is intended for multiplication and division of a number of input signals. In this case, the operations are specified in the same way as it was described for the summation / subtraction block using the multiplication signs ***** or division **/** in the template.

The **Sign** block is used to control the sign. It returns -1 for a negative input argument, 0 for a zero input argument, and 1 for a positive input argument.

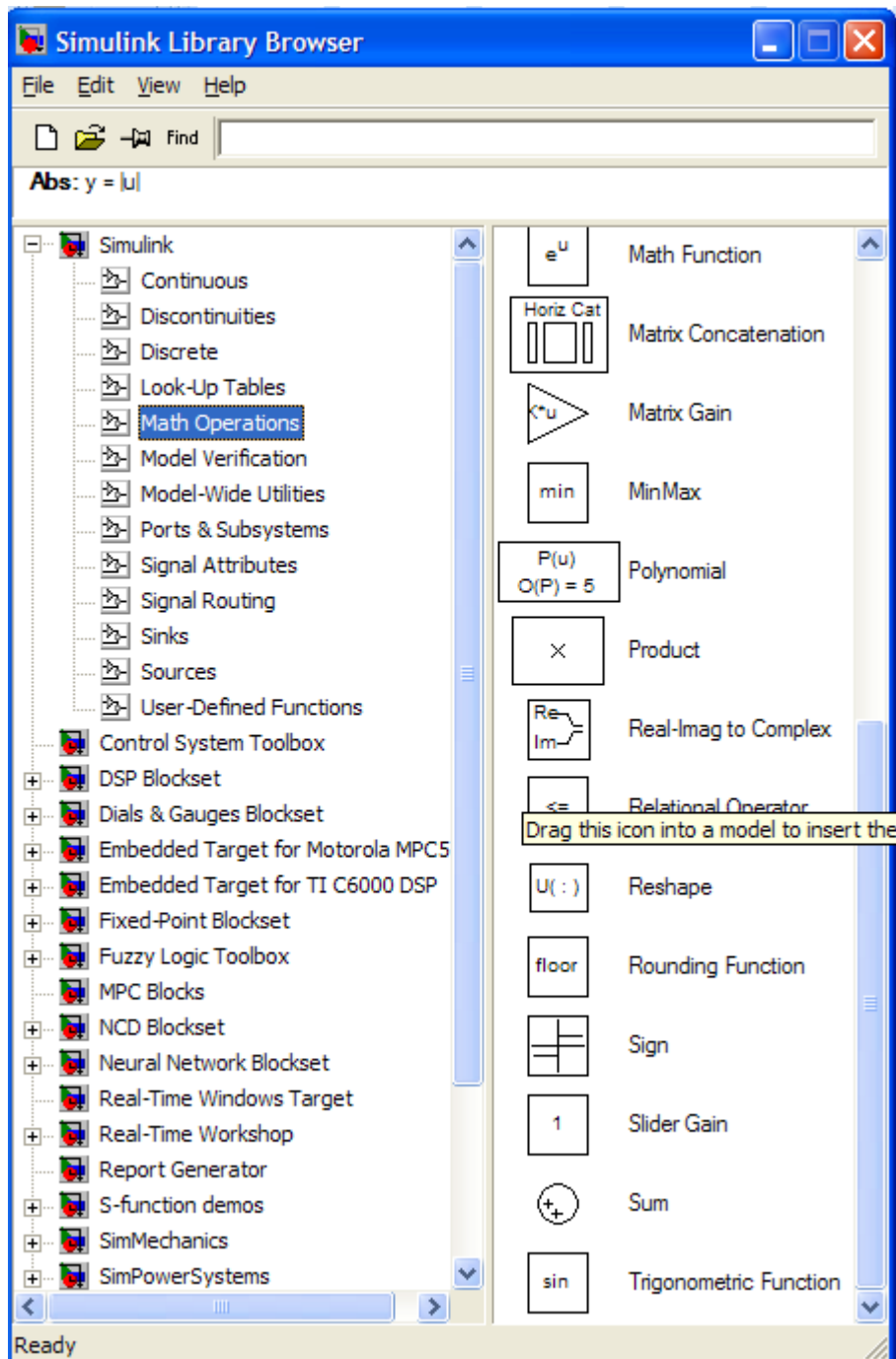


Fig 11 – Contents of the **Math Operations** section

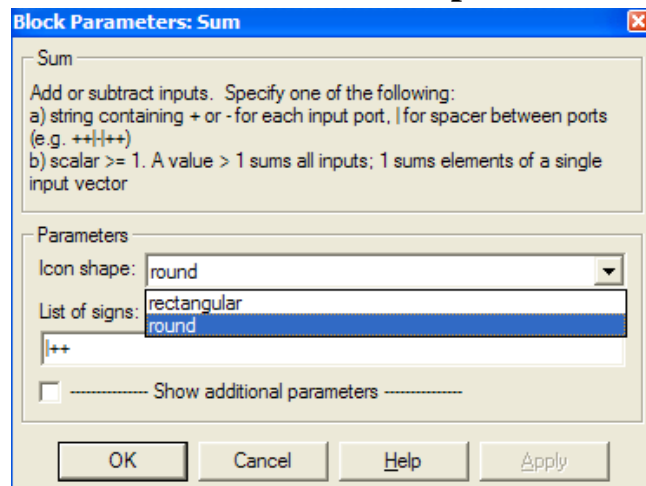


Fig 12 – **Addition/Subtraction** block setting window

The **Gain** and **Slider Gain** scaling blocks are used to scale the data (multiplying them by a given coefficient - a constant). In the **Gain block**, the constant is entered in the parameters window (default 1), and in the **Slider Gain block**, it can be selected using the slider. The **Matrix Gain** block is used to scale matrix data.

The **Math Function** block is used to define mathematical functions of one variable u according to the rules adopted for the programming language of the MATLAB base system. In particular, this means that built-in system functions can occur in the body of a function. The parameter window of this block contains a description of the rules for defining the function and the **Parameters** section, in which the expression for the function and the length of the output vector are specified. If it must match the length of the input signal vector, then the value -1 is entered (Figure 13).

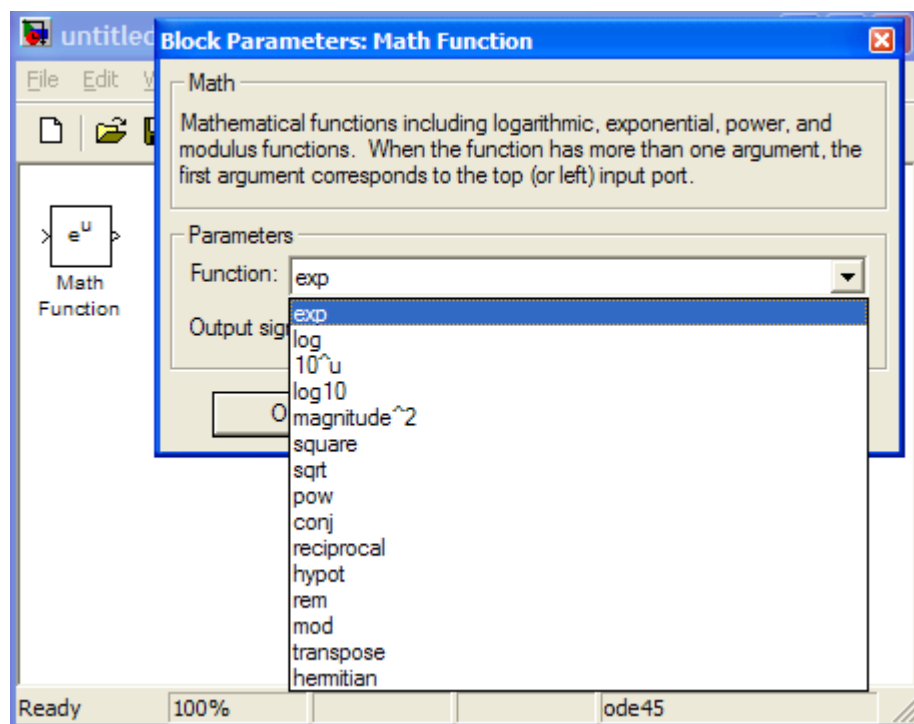


Fig 13 – Window of the parameters of block **Math Function**

An additional Simulink block library is located in the **Simulink Extras** section.

Task 1:

1. Familiarize yourself with the structure of the hierarchical Simulink library.
2. Type the model of the system given by the teacher.
3. Perform system simulation.
4. Compare the results obtained by changing the simulation parameters.

Task 2

1. Airport for 9 aircraft. Average values of time intervals between arriving and departing aircraft are set. The number of planes waiting to land is limited. The model stops if it is

impossible to receive the next aircraft. Variable variables: time intervals between arriving and departing aircraft, number of aircraft waiting to land. Observable variables: average landing time, average number of aircraft on the runway.

2. Gas station selling three types of gasoline. For each species, the probability of its use is given. The model stops when one of the types of gasoline is used up. Variable variables: stocks of each type of gasoline, probabilities of using each type. Observable variables: gross margin, unrealized balances.

3. Line for assembling computers, consisting of five components. For each component, a receipt period is specified, which is a random number. The model stops when the simulation time elapses. The number of components is considered unlimited. Variable variables: the period of receipt of each of the components, the time of assembly of the computer. Observed variables: the number of computers assembled per unit of time.

Control questions

1. What libraries does the Simulink package consist of?
2. How to build a model in Simulink package?
- 3 How to change simulation parameters?
4. What are the ways to visualize the modeling process?

Laboratory work No5

Data backup and recovery policy.

Malware detection and protection tools.

The purpose of the work: Mastering practical skills in the formation of an antivirus security policy and gaining skills in the operation of computer virus protection and antivirus programs

Theoretical part.

Anti-malware policies control settings for actions and notifications when malware is detected. The important settings for anti-malware policies are listed below.

1. **Action.** Specifies what to do if the message contains malware. The following options are available:

- Delete the message (this is the default).
- Replace all attachments with a text file that contains this default text: "Malware was found in one or more of the attachments in this e-mail message. All attachments have been removed."
- Replace all attachments with a text file that contains custom text.

2. **Notifications.** If the antivirus software policy is configured to delete messages, you can choose whether to send a notification message to the sender. The sending of notifications can depend on whether the sender is internal or external. By default, a notification message has the following properties:

- *From:* Postmaster <defaultdomain> postmaster@.com
- *Subject:* indistinguishable message
- *Message text.* This message was generated automatically by mail delivery software. Your email message was not delivered to the recipients because malware was detected."

You can customize message properties for internal and external notifications. You can also specify additional recipients (administrators) who will receive notifications of undelivered messages from internal or external senders.

3. **Recipient filters.** For a custom malaria program policy, you can specify conditions and recipient exceptions that determine who the policy applies to. You can use the following properties for conditions and exceptions:

- by recipient;
- by serviced domain;
- by group membership.

Removal - Removing a virus from an infected computer system may require reinstalling the OS from scratch, deleting files, or removing the virus from an infected file.

The possibility of a virus infection is proportional to the frequency with which new files or applications appear on the computer. Changes to the configuration for browsing the Internet, reading e-mail, and downloading files from external sources all increase the risk of virus infection.

The greater the value of the computer or the data that is in it, the more you need to take care of security measures against viruses. You also need to consider the costs of removing viruses from your computers, as well as from the computers of your customers, which you can infect. Costs are not always limited to finances, the reputation of the organization and other things are also important.

It is also important to remember that viruses usually appear on the system due to user actions (for example, installing an application, reading a file via FTP, reading an email). The prevention policy may therefore pay special attention to restrictions on the download of potentially infected programs and files. It may also specify that, in a high-risk environment, virus scanning should be especially thorough for new files.

1) Low risk. The Software Import Control Policy for a Low Risk Environment should primarily describe measures to communicate to users their obligation to regularly check for viruses.

Prevention: Users should be aware of the potential for viruses and RPS from the Internet and how to use antivirus tools.

Detection: Commercial anti-virus tools can be used to check for viruses weekly. Keeping logs of the antivirus tools is not necessary.

Removal: Any machine that is suspected of being infected with a virus should be immediately disconnected from the network. The machine should not be connected to the

network until the system administrators have verified that the virus has been removed. Where possible, commercial anti-virus programs should be used to remove the virus. If such programs cannot remove the virus, all programs on the computer must be removed, including boot entries if necessary. All these programs must be reinstalled from trusted sources and rescanned for viruses. (In Russia, registered users can contact the manufacturer of the program by e-mail and receive a program update with virus removal tools).

2) Medium risk. The software import control policy for the Medium Risk Environment should require more frequent virus checks, and the use of anti-virus software to scan servers and e-mail.

Prevention: Programs should only be downloaded and installed by a network administrator (who checks them for viruses or tests them).

File servers should have anti-virus software installed to limit the spread of viruses on the network. All programs and data files on file servers should be checked for viruses daily. Workstations should have memory-resident anti-virus programs configured so that all files are scanned for viruses when they are downloaded to the computer. All incoming e-mails must be checked for viruses. It is forbidden to run programs and open files using applications vulnerable to macro viruses before they are scanned for viruses.

The computer security training program should contain the following information about the risk of infection with viruses:

Antivirus programs can only detect viruses that have already been detected by someone before. New, more sophisticated viruses are constantly being developed. Antivirus programs should be updated regularly (monthly or quarterly) in order to be able to detect the latest viruses. It is important to report any unusual behavior on your computer or applications to your system administrator. It is important to immediately disconnect a computer that is infected or suspected of being infected from the network to reduce the risk of the virus spreading.

Detection: Commercial anti-virus programs should be used for daily virus checks. Antivirus programs should be updated every month. All programs or data imported into a computer (from floppy disks, e-mail, etc.) must be checked for viruses before they are used.

Antivirus logs should be kept and reviewed by system administrators. Employees should inform the system administrator about detected viruses, changes in configuration, or strange behavior of the computer or applications.

Upon receipt of information about a virus infection, the system administrator must inform all users who have access to programs and data files that may have been infected with a virus that the virus may have infected their systems. Users should be advised on how to determine if their system is infected and how to remove the virus from the system. Users should report virus scan and virus removal results to system administrators.

Removal: Any machine that is suspected of being infected with a virus should be immediately disconnected from the network. The machine should not be connected to the network until the system administrators have verified that the virus has been removed.

Where possible, commercial anti-virus programs should be used to remove the virus. If such programs cannot remove the virus, all programs on the computer must be removed, including boot entries if necessary. All these programs must be reinstalled from trusted sources and rescanned for viruses.

3) **High risk.** High-risk systems contain data and applications that are critical to the organization. Virus infections can cause significant loss of time, data, and damage to an organization's reputation. The infection can affect a large number of computers. All possible measures should be taken to prevent infection with viruses.

Prevention: A security administrator must allow applications to be used before installing them on a computer. It is forbidden to install unauthorized programs on computers

Removal: Any machine that is suspected of being infected with a virus should be immediately disconnected from the network. The machine should not be connected to the network until the system administrators have verified that the virus has been removed. Where possible, commercial anti-virus programs should be used to remove the virus. If such programs cannot remove the virus, all programs on the computer must be removed, including boot entries if necessary. All these programs must be reinstalled from trusted sources and rescanned for viruses.

Practical part

In the practical part of this work, using the example of the “Anti-Virus Program Policy” of the Department of Informatization, we will study how to draw up an Anti-Virus Program Policy.

The anti-virus software policy consists of the following sections:

1. Terms and definitions
2. Basic provisions
3. General requirements
4. Responsibility.

1. Terms and definitions

The terms, definitions and abbreviations used in this document are used in accordance with the document OIB-TO/3.13/001 "Terms and Definitions".

2. Basic provisions

2.1. This document "Anti-Virus Protection Policy" (hereinafter referred to as the "Policy") - establishes a system of measures aimed at protecting the confidential information of the CSPD TO systems, on which the impact of computer viruses and malicious software (Trojans, bomb connections, etc.) depends, as well as establishes uniform requirements for the organization of an anti-virus protection system for information systems and all types of workstations, requirements for prescribing the use of software and procedures for their operation.

2.2. Implementation of applicable policies applied in the prescribed manner by information security administrators of the IOGV or by a group (or division) involved in performing data functions.

2.3. At the end of the year to carry out the protection system in real conditions. An unscheduled review may also be required when changing the list of tasks to be solved, considering hardware and software.

2.4. This Policy:

- define uniform requirements for the classification of anti-virus protection in all information sources and single out APMs that are part of or associated with the CSPD TO;
- limits the necessary and sufficient anti-virus protection measures for the observed and prosecuted work of users of the CSPD TO;
- the number of participants in the IS processes is calculated when organizing anti-virus protection of objects of the CSDC TO;
- establishes the responsibilities of users of KSPD TO and administrators of security disclosure of IOGV within the organization of anti-virus protection;
- connection to all information systems and separate APMs connected to KSPD TO;
- obligatory for conclusion by all participants, executors under agreements with participants of IBam/contracts and other persons and organizations that constantly use or observe periodic connection to network or information resources of KSPD TO.

3. General requirements.

3.1. It is prohibited to use software, program codes or algorithms in the process of normal (normal) operation of information systems, leading to the destruction, destruction of information resources.

3.2. Anti-malware methods should include three components:

- Prevention - actions to prevent malware infection;
- Detection - methodology for detecting the presence of malicious software;
- Removal - physical removal of malware codes from infected files or an infected system.

3.3. Only licensed anti-virus tools approved for use in the authorities of the Russian Federation are allowed for use in information systems that are part of the KSPD TO.

3.4. All servers, workstations and other technological services and systems must be provided with anti-virus protection if the operating system used on them has a virus vulnerability.

3.5. The versions of used anti-virus tools and anti-virus databases are updated on a regular basis.

3.6. Server system administrators and resource administrators must maintain the highest possible level of security for the software and hardware entrusted to them at all times. For this you need:

- track information coming from developers of system and software about detected errors and vulnerabilities;
- timely install updates and corrections officially recommended by the developers of system and application software;
- disable all unused services and applications of the operating system (or application software);

- keep the installed anti-virus protection tools up to date.
- keep logs of system events and regularly analyze them;
- follow the recommendations of the information security administrator of the IOGV.
- Downloading software and working files to computers, servers and other storage media is carried out with their preliminary check by anti-virus tools.

3.7. Downloading software and working files to computers, servers and other storage media is carried out with their preliminary check by anti-virus tools.

3.8. Users are prohibited from installing and using specialized software on their own without prior approval from the information security administrator of the IOGV.

3.9. The installation of permitted anti-virus protection tools on users' computers is carried out by information security administrators of the IOGV, who have the appropriate authority.

3.10. KSPD TO users are prohibited from disabling anti-virus protection tools installed on computers or making changes to their configuration that reduce the established level of protection.

The task:

Study the procedure for developing an anti-virus program policy and formulate a policy in the sequence indicated in the practical part using the example of an organization

Control questions:

1. How is the security policy of anti-virus programs formed?
2. What sections does the antivirus software policy include?
3. Can I specify the order if I create multiple custom antivirus policies?
4. What settings need to be made in conditions where the risks are "low", "medium" and "high"?
5. What are the components of anti-malware methods?

Laboratory work No 6

Components used in visual programming.

Loop statements in programming

The purpose of the work: to study the loop operators, to learn how to use the simplest components of the loop organization). Write and debug a program with For, While, Do while statements.

Tasks:

1. Study the theoretical part.
2. Download the Borland C++ Builder6 system.
3. Implement the program of your choice, get results.
4. Compile a report on the work done.

Theoretical part

Loop statements. A loop statement is an instruction to the program to repeat a certain sequence of statements a specified number of times or until a certain condition is met. There are three types of loops in C++.

- with precondition (while),
- with postcondition (do while),
- with the (for) parameter.

Any cycle consists of the body of the cycle, that is, those statements that are executed several times, initial settings, modification of the cycle parameters and checking the condition for continuing the execution of the cycle.

One pass of the loop is called an iteration. The condition is checked at each iteration either before the loop body (loop with precondition) or after the loop body (loop with postcondition). The difference between these two methods is that a loop with a precondition may not be executed even once if the condition from the very beginning turns out to be false, and a loop with a postcondition is guaranteed to be executed at least once.

The loop terminates if the continuation condition is not met. If necessary, it is possible to force the end of the entire loop with the break statement or the current iteration with the continue statement.

Loop while (with precondition). The general format of an operator:

while (expression) <operator>;

while (expression) {compound operator};

where *expression* - a logical expression (a condition for repeating the cycle), *operator* - an operator or a sequence of operators included in the body of the cycle.

The execution of the statement begins by checking the condition in parentheses after the while. If the condition is true, the loop statement is executed, otherwise control is transferred to the statement following the loop. If at the first check the condition is not met (equal to false), then the loop will not be executed even once.

This can also be used for purposes such as declaring a loop:

while (true) { . . . } ;

and organize an exit from inside the loop using break.

Loop do while (with postcondition). The general format of an operator:

do <operator> while (expression);

do {compound operator} while expression;

The *do while* loop differs from the *while* loop only in that it checks the condition after the loop, i.e. it will reliably execute at least once.

The algorithm of this cycle is as follows: first, a simple or compound statement is executed, which is the body of the cycle, then the value of the expression is determined. If it is not false, the body of the loop is executed again. The loop terminates when the expression evaluates to false, or when the loop exits with a control transfer statement.

One application of the *do while* loop is when input of some value must continue until it takes on a certain value, for example:

do {

```

cout << "Enter the number from 1 to 100 (0 – output) : ";
cin >> num;
... // actions with numbers
while (num != 0);
}

```

Loop for (with parametr). The general format of an operator:

for (initialization; expression; modifications) operator;

A loop with a parameter is one of the most powerful tools in the C++ language. It allows not only to repeat a sequence of statements a specified number of times, but also to create complex loops with exit conditions and other advanced features.

The scheme of the for loop is as follows:

1. Initialization is in progress. It is performed only once before the start of the cycle. Typically, initialization involves setting the counter variable to an initial value.
2. As in the while loop, the truth of the expression is checked, and if it is true, the body of the loop is executed, otherwise it is skipped.
3. Finally, the operations described in the modifications section are performed, usually this is a change in the control variable.

Let's look at examples of what loops can be created using for:

Example 1.

```
for (int n = 1; n <= 10; n++) { ... }
```

In this example, the variable n takes all values from 1 to 10, that is, the loop body will be executed 10 times (if n does not change in the loop body).

Example 2

```
for (int i = 1, j = 100; i != j; i ++, j --) { ... }
```

This is a more complex example of using the operator. Here, in the initialization section, initial values are assigned to two variables i and j, after each iteration i by one, and the second j is decremented by one. The output from the loop occurs when the values of both variables are equal, that is, there will be $100/2 = 50$ iterations in total.

```
for (int k = 1; ; k++) { ... }
```

In this case, the condition for exiting the loop is omitted - in its place is an empty operator, consisting only of a sign; (semicolon). This means that the loop body will be executed until the control transfer operator is encountered inside, but at the end of each iteration, the value of the counter variable k is incremented by one.

Example 3.

```
for (int i = 1; i <= 10; a[i] = i, i ++);
```

This example considers the case when it is necessary to initialize an array of ten elements (in this case, the ordinal numbers of the elements are the values of the array elements). This

cycle does not even have a body - all the necessary operations are already performed in the header, in the modifications section. After the next iteration, first the value is assigned to the next element of the array, and then the actual increment of the counter.

The *do while* statement is useful when the loop must be executed at least once (for example, it has data input, followed by input validation).

The *for* statement is ideal for a counter-driven loop, but is also handy in many other cases.

In other loops, you can use *while*, especially if the number of iterations is not known in advance, there are no obvious loop parameters, or it is convenient to modify them at the end of the loop.

Solution of one variant:

Task 1. Calculate the value of the functions f1 and f2 for the values of the variable x, changing in the interval from a to b with step h.

$$f1 = \sqrt{x^2} x + 1 * e^{-x} \quad f2 = 1 + 2\sin x$$

```
//-----
#include <math.h>
#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
int main(int argc, char* argv[])
{
    float a, b, h, f1, f2, x ;
    cout<<"Enter the value A, B, H" <<endl;
    cin>>a>>b>>h;
    x=a;
    while (x<=b)
    {
        f1=sqrt(pow(x,2))*x+1*exp(-x);
        f2=1+2*sin(x);
        cout<<"x= "<<x<<" f1="<<f1<<" F2="<<f2<< endl;
        x=x+h;
    }
    getch();
    return 0;
}
```

```

    }
//-----
The program is implemented in console mode.

```

Task 2. Create a program for calculating the function F in the interval $x \in [a, b]$ with step h , using the loop operator with a postcondition. This program uses Memo, Label, Edit, Button components.

```

Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Button1: TButton;
Label6: TLabel;
Memo1: TMemo;

```

Button1 is used to calculate F . Memo1 is used to display all values of the function F on the range $[a, b]$.

Code of the program

```

//-----
#include <math.h>
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void main();

```

```

void __fastcall TForm1::Button1Click(TObject *Sender)
{
float a,b,h,f,x;
h=StrToFloat(Edit1->Text);
a=StrToFloat(Edit2->Text);
b=StrToFloat(Edit3->Text);
x=a;
do {
f=x*x +sin(x) +exp(x);
Memo1->Lines->Add("X= "+FloatToStr(x)+" F= "+FloatToStr(f));
a=a+h; }
while (a<=b);
}
//-----

```

When the program is executed, the results will be obtained in the following Form (Fig. 1):

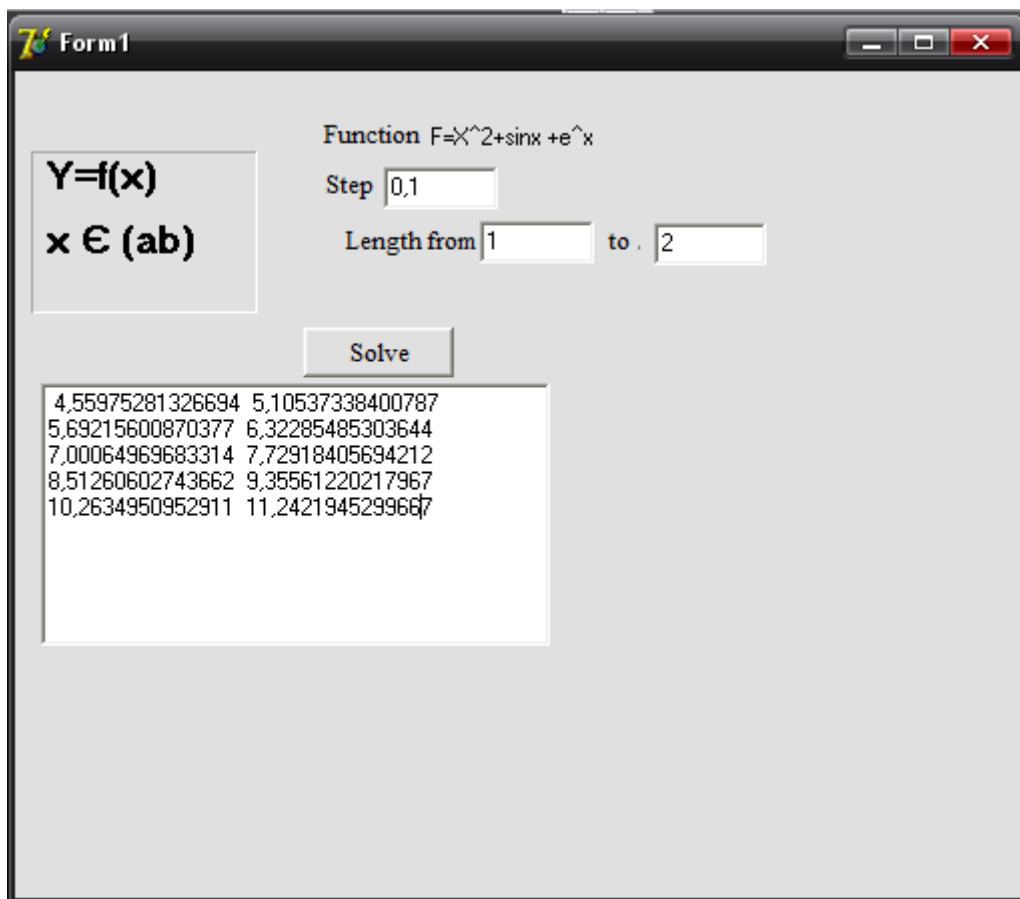


Fig.1. Results form

Write and debug a program to display all the values of the function $S(x)$ for the argument x , which varies in the range from a to b with a step h and a given n .

$$S(x) = \sum_{k=0}^N (-1)^k \frac{x^k}{k!}.$$

Example of creating a windowed application

The text of handler functions can be as follows (standard text omitted):

```
//-----  
void __fastcall TForm1::FormCreate(TObject *Sender)  
{  
    Edit1->Text="0,1";    Edit2->Text="1,0";  
    Edit3->Text="10";    Edit4->Text="0,2";  
    Memo1->Lines->Add("Laboratory work 3");  
}
```

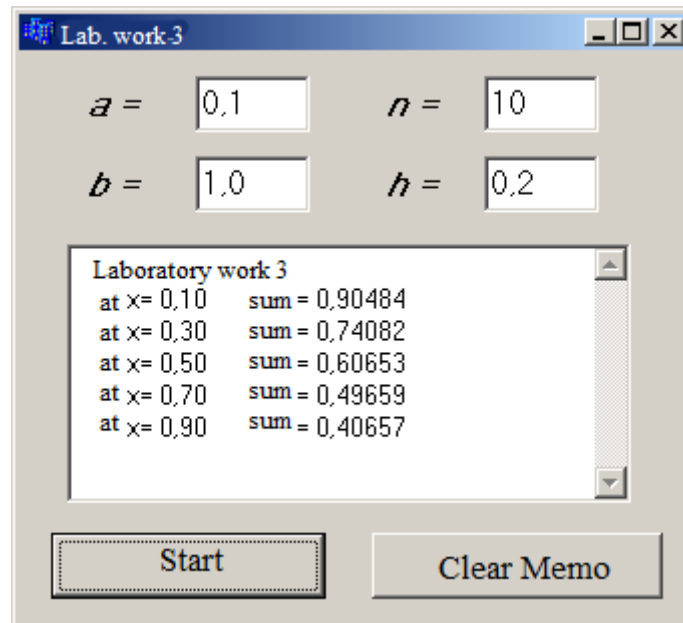


Fig.2.

```
//-----  
void __fastcall TForm1::Button1Click(TObject *Sender)  
{  
    double a, b, x, h, r, s;  
    int n, zn = -1, k;  
    a = StrToFloat(Edit1->Text);  
    b = StrToFloat(Edit2->Text);  
    n = StrToInt(Edit3->Text);  
    h = StrToFloat(Edit4->Text);  
    for(x = a; x<=b; x+=h) {  
        r = s = 1;  
        for(k = 1; k<=n; k++) {  
            r = zn*r*x/k;  
            s+=r;  
        }  
        Memo1->Lines->Add("при x= "+FloatToStrF(x,ffFixed,8,2)  
            +" sum= "+FloatToStrF(s,ffFixed,8,5));  
    }  
}
```

```

    }
}
//-----
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Memo1->Clear();
}

```

An example of creating a console application

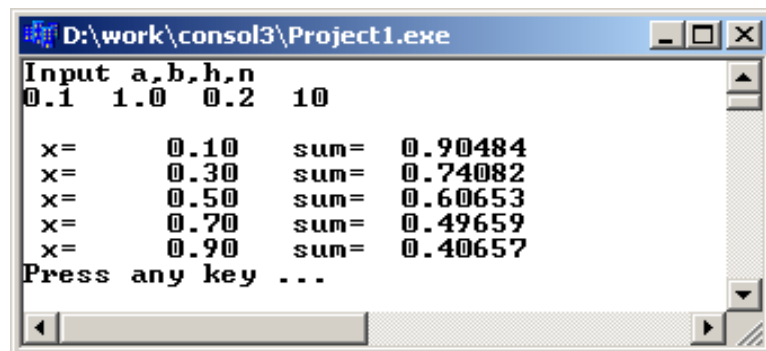
The text of the program of the proposed task may look like

```

#include <vcl.h>
#include <stdio.h>
#include <conio.h>
#pragma hdrstop
#pragma argsused
int main(int argc, char* argv[])
{
    double a, b, x, h, r, s;
    int n, zn = -1, k;
    puts("Input a,b,h,n");
    scanf("%lf%lf%lf%d", &a, &b, &h, &n);
    for(x = a; x<=b; x+=h) {
        r = s = 1;
        for(k = 1; k<=n; k++) {
            r=zn*r*x/k;
            s+=r;
        }
        printf("\n x= %8.2lf  sum= %8.5lf", x,s);
    }
    puts("\nPress any key ... ");
    getch();
    return 0; }

```

The result of the program with the entered values $a=0.1$, $b=1.0$, $h=0.2$ и $n=10$:



```

D:\work\consol3\Project1.exe
Input a,b,h,n
0.1 1.0 0.2 10

x=      0.10    sum=   0.90484
x=      0.30    sum=   0.74082
x=      0.50    sum=   0.60653
x=      0.70    sum=   0.49659
x=      0.90    sum=   0.40657
Press any key ...

```

Fig.3.

Variants of tasks for individual laboratory work:

For each x changing from a to b with step h , find the values of the function $Y(x)$, the sums $S(x)$ and $|Y(x) - S(x)|$ and output it as a table. The values a , b , h and n are entered from the keyboard. Since the value of $S(x)$ is a series of expansion of the function $Y(x)$, with the correct solution, the values of S and Y for a given argument x (for test values of the initial data) must match in the integer part and in the first two to four positions after the decimal point.

Check the operation of the program for $a = 0.1$; $b = 1.0$; $h = 0.1$; select the value of the parameter n depending on the task.

$$1. S(x) = \sum_{k=0}^n (-1)^k \frac{x^{2k+1}}{(2k+1)!}, \quad Y(x) = \sin(x).$$

$$2. S(x) = \sum_{k=1}^n (-1)^{k+1} \frac{x^{2k}}{2k(2k-1)}, \quad Y(x) = x \cdot \arctg(x) - \ln \sqrt{1+x^2}.$$

$$3. S(x) = \sum_{k=0}^n \frac{\cos(k\pi/4)}{k!} x^k, \quad Y(x) = e^{x \cos \frac{\pi}{4}} \cos(x \sin(\pi/4)).$$

$$4. S(x) = \sum_{k=0}^n (-1)^k \frac{x^{2k}}{(2k)!}, \quad Y(x) = \cos(x).$$

$$5. S(x) = \sum_{k=0}^n \frac{\cos(kx)}{k!}, \quad Y(x) = e^{\cos x} \cos(\sin(x)).$$

$$6. S(x) = \sum_{k=0}^n \frac{2k+1}{k!} x^{2k}, \quad Y(x) = (1+2x^2)e^{x^2}.$$

$$7. S(x) = \sum_{k=1}^n \frac{x^k \cos(k\pi/3)}{k}, \quad Y(x) = -\frac{1}{2} \ln(1 - 2x \cos \frac{\pi}{3} + x^2).$$

$$8. S(x) = \sum_{k=0}^n \frac{(2x)^k}{k!}, \quad Y(x) = e^{2x}.$$

$$9. S(x) = \sum_{k=1}^n (-1)^{k+1} \frac{x^{2k+1}}{4k^2 - 1}, \quad Y(x) = \frac{1+x^2}{2} \arctg(x) - x/2.$$

$$10. S(x) = \sum_{k=0}^n \frac{x^{2k}}{(2k)!}, \quad Y(x) = \frac{e^x + e^{-x}}{2}.$$

$$11. S(x) = \sum_{k=0}^n \frac{k^2 + 1}{k!} (x/2)^k, \quad Y(x) = (x^2/4 + x/2 + 1)e^{x/2}.$$

$$12. S(x) = \sum_{k=0}^n (-1)^k \frac{2k^2 + 1}{(2k)!} x^{2k}, \quad Y(x) = (1 - \frac{x^2}{2}) \cos(x) - \frac{x}{2} \sin(x).$$

$$13. S(x) = \sum_{k=1}^n (-1)^k \frac{(2x)^{2k}}{(2k)!}, \quad Y(x) = 2(\cos^2 x - 1).$$

$$14. S(x) = \sum_{k=0}^n \frac{x^{2k+1}}{(2k+1)!}, \quad Y(x) = \frac{e^x - e^{-x}}{2}.$$

$$15. S(x) = \sum_{k=1}^n (-1)^{k+1} \frac{x^{2k}}{2k(2k-1)}, \quad Y(x) = -\ln \sqrt{1+x^2} + x \arctg(x).$$

$$16. S(x) = \sum_{k=0}^n \frac{\cos(k\pi/4)}{k!} x^k, \quad Y(x) = \cos[x \cdot \sin(\pi/4)] e^{x \cos \frac{\pi}{4}}.$$

Control questions:

1. What is a cyclic computing process?
2. What loop operators do you know?
3. What types of variables are used as a parameter in a loop statement with a parameter?
4. Difference between WHILE and DO WHILE loop statements.
5. What components of Borland C++ Builder6 are used in the implementation of cyclic programs
6. What procedures are used to process strings in Borland C++ Builder6?

Laboratory work №7

Programming Engineering Problems in an Integrated Environment Implementing programming through modules and applying engineering problems to object-oriented programs.

The purpose of the work: to study the regular data type - an array, the main properties of the *StringGrid* component. Write and debug a program using one-dimensional arrays.

Tasks:

1. Study the theoretical part.
2. For a given variant, write a program using arrays.
3. Implement the program in console mode or visual mode.
4. Compile a report on the work performed.

Theoretical part

When using simple variables, each area of memory for storing data has its own name. If it is required to perform similar actions with a group of quantities of the same type, they are given the same name, and are distinguished by a serial number.

An array is an object of a complex type, each element of which is defined by a name (*ID*) and an integer value of the index (number) by which the array element is accessed. Consider one-dimensional arrays.

Array indexes in C/C++ start from 0.

In the program, a one-dimensional array is declared as follows:

type ID array [size];

where *size* is the number of elements in the array. The size of an array can be given by a constant or a constant expression. To use arrays of variable size, there is a separate mechanism - dynamic memory allocation.

Array declaration examples:

```
int a[5];
```

```
double b[4] = { 1.5, 2.5, 3.75 };
```

in an integer array, and the first element is *a*[0], the second is *a*[1], ..., the fifth is *a*[4]. The array *b*, consisting of real numbers, has been initialized, and the array elements will have the following values: *b*[0]=1.5, *b*[1]=2.5, *b*[2]=3.75, *b*[3]=0.

The C/C++ language does not check if an index is out of bounds in an array. The correctness of the use of indexes of array elements must be controlled by the programmer.

Examples of array descriptions:

```
const Nmax=10;
```

– setting the maximum value;

```
typedef double mas1[Nmax*2];
```

– description of the type of a one-dimensional array;

```
mas1 a;
```

– declaration of array *a* of type mas1;

```
int ss[10];
```

– an array of ten integers.

Array elements can be used in expressions in the same way as ordinary variables, for example:

```
f = 2*a[3] + a[ss[i] + 1]*3;
```

```
a[n] = 1 + sqrt(fabs(a[n-1]));
```

Creating a Window Application of the StringGrid Component

When working with arrays, the input and output of values is usually organized using the *StringGrid* component, designed to display information in the form of a two-dimensional table, each cell of which is a one-line editor window (similar to the *Edit* window). Information is accessed using the *Cells[ACol][ARow]* element of the *AnsiString* type, where the integer values *ACol*, *ARow* indicate the position of the element.

The first ACol index defines the column number, and the second ARow defines the row number, unlike array indexes.

In the Object Inspector, the *ColCount* and *RowCount* values set the initial values for the number of columns and rows in the table, while *FixedCols* and *FixedRows* set the number of fixed zone columns and rows. The fixed zone is highlighted in a different color and is usually used for labels.

Processing two-dimensional dynamic arrays

Objects of any type can be referred to in C by name, as we have done so far, and by **pointer** (indirect addressing).

A pointer is a variable that can contain the address of some object in the computer's memory, such as the address of another variable. Through a pointer set to a variable, you can access the area of RAM allocated by the compiler for its value.

The pointer is declared like this:

```
type * pointer ID;
```

Before use, the pointer must be initialized either with a specific address or with a *NULL* (0) value - no pointer.

There are two unary operations associated with pointers: & and *. The & operator means "take the address", and the unaddress operator * means "the value located at the address", for example:

```
int x, *y;           // x - int type variable, y - int type pointer
y = &x;             // y - address of x variable
*y = 1;            // to address y write down 1, as a result x = 1
```

When working with pointers, you can use the operations of addition, subtraction, and comparison, and they are performed in units of the type to which the pointer is set.

The operations of addition, subtraction and comparison (more/less) make sense only for sequentially located data - arrays. The comparison operations "==" and "!=" make sense for any pointers, i.e. if two pointers are equal, then they point to the same variable.

Relation of pointers to arrays

Pointers and arrays are closely related. An array identifier is a pointer to its first element, i.e. for the array `int a[10]`, the expressions `a` and `a[0]` have the same values, because the address of the first (with index 0) element of the array is the address of the beginning of the placement of its elements in the RAM.

Let declared - an array of 10 elements and a pointer of *double* type:

```
double a[10], *p;
```

if `p = a`; (setup pointer `p` to the start of array `a`), then the following calls: `a[i]`, `*(a+i)` and `*(p+i)` are equivalent, i.e. for any pointers, two equivalent forms of accessing array elements can be used: `a[i]` and `*(a+i)`. The equivalence of the following expressions is obvious:

```
&a[0] ↔ &>(*p) ↔ p
```

Declaration of a multidimensional array:

```
type ID[size 1][size 2]...[size N];
```

and the last index changes faster, because multidimensional arrays are placed in the RAM in a sequence of columns, for example, an array of integer type, consisting of two rows and three columns (with initialization of initial values)

```
int a[2][3] = {{0,1,2},{3,4,5}};
```

in the RAM will be placed as follows:

```
a[0][0]=0, a[0][1]=1, a[0][2]=2, a[1][0]=3, a[1][1]=4, a[1][2]=5.
```

If there is not enough data in the list of initializers, then the corresponding element is assigned the value 0.

Pointers to pointers

The relationship between pointers and arrays with one dimension is also true for arrays with more dimensions.

If we consider the previous array (*int a[2][3]*;) as an array of two arrays of three elements each, then referring to the element *a[i][j]* corresponds to the equivalent expression **(*(a+i)+j)*, and the declaration of this array using pointers will look like

```
int **a;
```

Thus, the name of a two-dimensional array is the ID of a pointer to a pointer.

Dynamic placement of data

To create arrays with variable dimensions, dynamic placement of data declared by pointers is used.

To work with dynamic memory, the standard functions of the *alloc.h* library are used:

*void *malloc(size)* and *void *calloc(n, size)* – allocate a memory block of *size* and *n*size* bytes, respectively; return a pointer to the selected area, *NULL* on error;

void free(bf); - frees previously allocated memory with address *bf*.

Another, more preferred approach to dynamic memory allocation is to use the C++ language operators *new* and *delete*.

The *new* operation returns the address of the RAM allocated for the dynamically allocated object, *NULL* on error, and the *delete* operation releases the memory.

The minimum set of actions required to dynamically allocate a one-dimensional array of *real numbers of size n*:

```
double *a;
...
a = new double[n];      // Capturing memory for n elements
...
delete []a;            // Freeing up memory
```

The minimum set of actions required to dynamically allocate a two-dimensional array of *real numbers of size nxm*:

```
int i, n, m;           // n, m – array dimensions
double **a;
a = new double *[n];   // Capturing memory under pointers

for(i=0; i<n; i++)    a[i] = new double [m]; // and under elements
...
for(i=0; i<n; i++)    delete []a[i];       // Freeing up memory
delete []a;
```

For modern compilers (versions older than "6"), to free memory, it is enough to write only `delete []a;`

A finite named sequence of values of the same type is called *an array*.

C++ distinguishes between *fixed size* arrays and *variable size (dynamic)* arrays. The number of elements in an array of the first type is known when the program is written and never changes. The compiler allocates memory for such an array. The number of elements of a dynamic array is not known at compile time and usually depends on the input data. Memory for a dynamic array is allocated during program execution using memory allocation operations.

Solution of one variant

Task 1. A one-dimensional array A is given. Find the maximum and minimum elements of the array and their sum.

The program code in console mode can be written in the following form:

```
//-----  
#include<iostream.h>  
#include<conio.h>  
#include <vcl.h>  
#pragma hdrstop  
//-----  
#pragma argsused  
int main(int argc, char* argv[])  
{ const n=20;  
  float a[n]; int k,i, max,min; float sum;  
  cout<<"Number of array elements";  
  cin>>k;  
  for (i=0; i<k; i++) cin>>a[i];  
  max=a[0]; min=a[0];  
  for (i=1; i<k; i++) if (a[i]>max) max=a[i];  
  for (i=1; i<k; i++) if (a[i]<min) min=a[i];  
  sum=max+min;  
  cout<<"max element of array="<<max<<endl;  
  cout<<"min element of array="<<min<<endl;  
  cout<<"Sum Max and Min ="<<sum;  
  getch(); return 0;  
}  
//-----
```

Task 2. Given a square matrix. Calculate the sum of all array elements.

To implement the program for solving this problem in the visual mode, the Lable, Button, StringGrid components are used. The StringGrid component is used to enter or display elements of an array in a tabular form, i.e. this component sets the table to the Form (the

component is included in the Additional tool palette). The StringGrid component has the following main properties:

ColCount – setting the number of columns;

RowCount - setting the number of rows;

FixedRow – setting the number of the initial active row;

FixedCol - setting the number of the initial active column.

Options – defines table options, for example, if the GoEditing parameter is set to *true*, then data in cells is allowed to be edited.

We write the program code in the following form for the matrix A(4X4):

```
-----  
#include <vcl.h>  
#pragma hdrstop  
#include "Unit1.h"  
-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TForm1 *Form1;  
-----  
__fastcall TForm1::TForm1(TComponent* Owner)  
    : TForm(Owner)  
{  
}  
-----  
int sum=0;  
void __fastcall TForm1::Button1Click(TObject *Sender)  
{  
    int i,j, a[4][4];  
    randomize();  
    for(i=0;i<4;i++)  
        for(j=0;j<4;j++)  
            { a[i][j]=random(10);  
              StringGrid1->Cells[i][j]=IntToStr(a[i][j]);  
              sum+=a[i][j];}  
}  
-----  
void __fastcall TForm1::Button2Click(TObject *Sender)  
{  
    Label1->Caption="Sum of all of the elements=" +IntToStr(sum);  
}  
-----
```

Button1 used to fill an array (Stringgrid) with random numbers. **Button2** is used to display the sum of all array elements. The general view of the Form and the results obtained are shown in Figure 1.

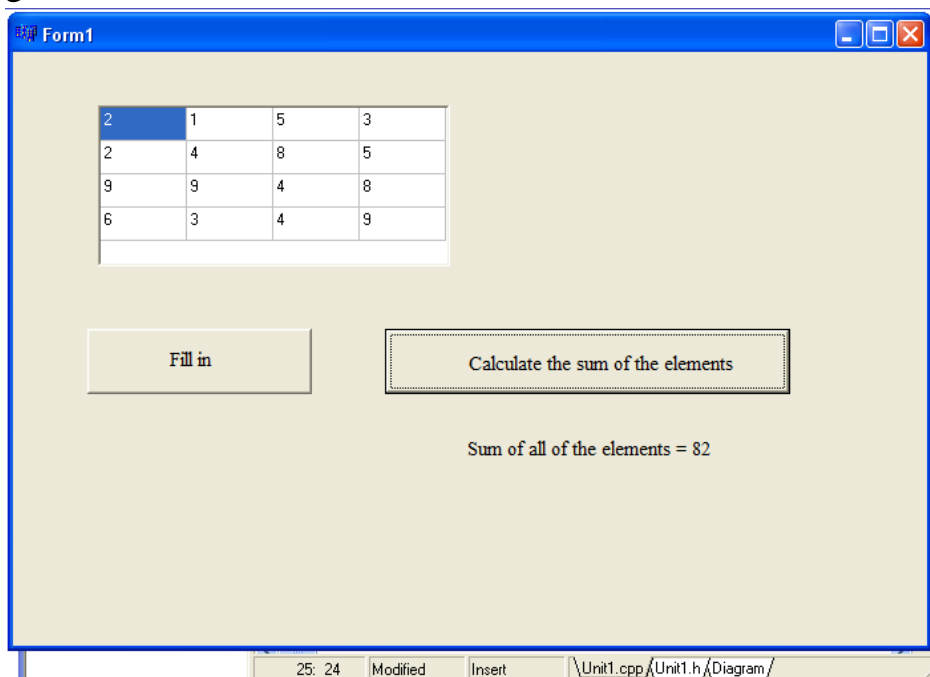


Fig.1. View of the Form with the Stringgrid component and the results

Task 3. Remove from array A of size N , consisting of integers (positive and negative), all negative numbers. Do not create a new array. To fill the array, use the $random(kod)$ function, a generator of random uniformly distributed integers from 0 to $(int)kod$.

Example of creating a windowed application

Enter the value N from *Edit*, the values of array A - from the *StringGrid* component. Output the result to the *StringGrid* component.

The dialog panel and the results of the program execution are shown in Fig. 2.

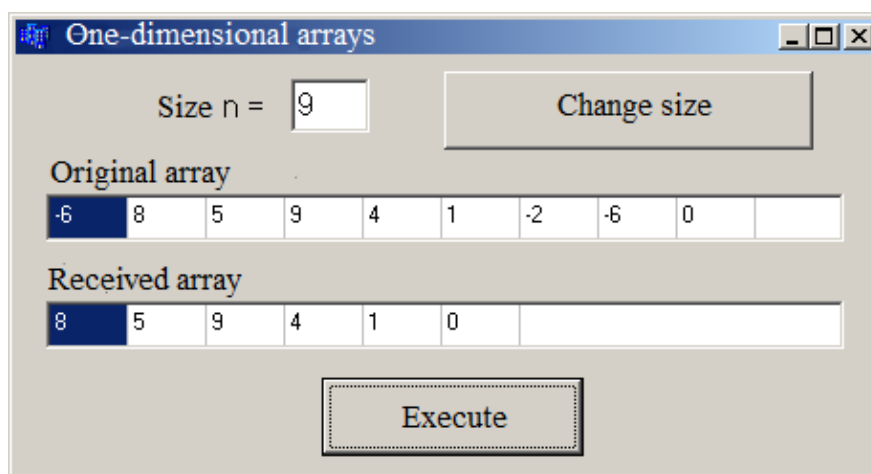



Fig. 2.

Customizing the StringGrid Component

On the *Additional tab*, select the icon , set the *StringGrid1* and *StringGrid2* components and adjust their sizes. In the Object Inspector for both components, set

ColCount to 2, *RowCount* to 1, i.e. two columns and one row, and the *FixedCols* and *FixedRows* values are equal to 0. The *DefaultColWidth* column cell width value is equal to 40.

By default, input to the *StringGrid* component is only allowed programmatically. To allow data input from the keyboard, set the *goEditing* line for the *StringGrid1* component to *true* in the *Options* property.

The text of handler functions may look like this:

```

    ...
int n = 4;
//-----
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    randomize();           // Changing the start address for random()
    Edit1->Text=IntToStr(n);
    StringGrid1->ColCount=n;
    for(int i=0; i<n;i++)   // Filling array A with random numbers
        StringGrid1->Cells[i][0] = IntToStr(random(21)-10);
    Label3->Hide();       // Hide component
    StringGrid2->Hide();
}
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    n=StrToInt(Edit1->Text);
    if(n>10){
        ShowMessage("Maximum amount10!");
        n=10;
        Edit1->Text = "10";
    }
    StringGrid1->ColCount=n;
    for(int i=0; i<n;i++)
        StringGrid1->Cells[i][0]=IntToStr(random(21)-10);
    Label3->Hide();
    StringGrid2->Hide();
}
//-----
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    int i, kol = 0, a[10];   // Declaration of a one-dimensional array
    // Filling array A with elements from table StringGrid1

```

```

for(i=0; i<n;i++)
    a[i]=StrToInt(StringGrid1->Cells[i][0]);
// Removing negative elements from array A
for(i=0; i<n;i++)
    if(a[i]>=0) a[kol++] = a[i];
StringGrid2->ColCount = kol;
StringGrid2->Show();          // Show component
Label3->Show();
// Outputting the result to a StringGrid 2 table
for(i=0; i<kol;i++) StringGrid2->Cells[i][0]=IntToStr(a[i]); }

```

An example of creating a console application

The text of the program may look like this (note that the *main* function is used in its simplest form - without parameters and does not return results):

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int a[10],n, i, kol=0;
    randomize();          // Changing the start address for random()
    printf("Input N (<=10) ");
    scanf("%d", &n);
    puts("\n Massiv A");
    for(i=0; i<n;i++) {
        a[i] = random(21)-10;          // Filling array A with random numbers
        printf("%4d", a[i]);    } // Removing negative elements from array A
    for(i=0; i<n;i++)
        if(a[i]>=0) a[kol++] = a[i];
    puts("\n Rezult massiv A");
    for(i=0; i<kol;i++) printf("%4d", a[i]);
    puts("\n Press any key ... ");
    getch(); }

```

With array A filled with random numbers, the result of the program could be as follows:

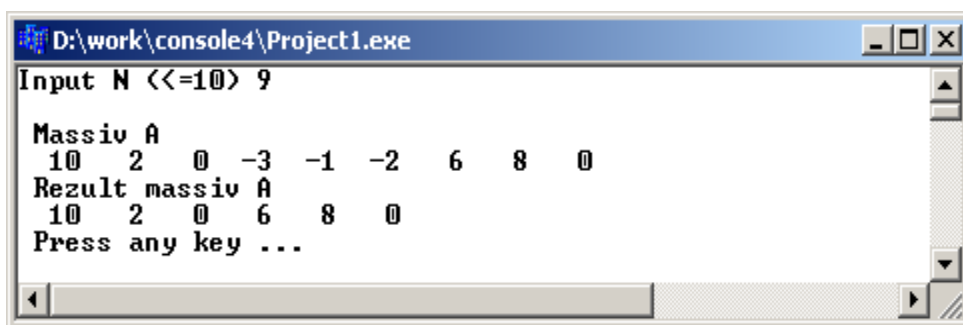


Fig.3.

Task 4. Calculate vector $\vec{Y} = A \cdot \vec{B}$ values, where A – square matrix of $N \times N$ size, Y and B – one-dimensional arrays of size N . Elements of the vector Y are determined by the formula

$$Y_i = \sum_{j=0}^{N-1} A_{ij} \cdot B_j.$$

Example of creating a windowed application

Enter the N value from *Edit*, A and B from the *StringGrid* component. Output the result to the *StringGrid* component.

The dialog panel and program execution results are shown in fig. 4.

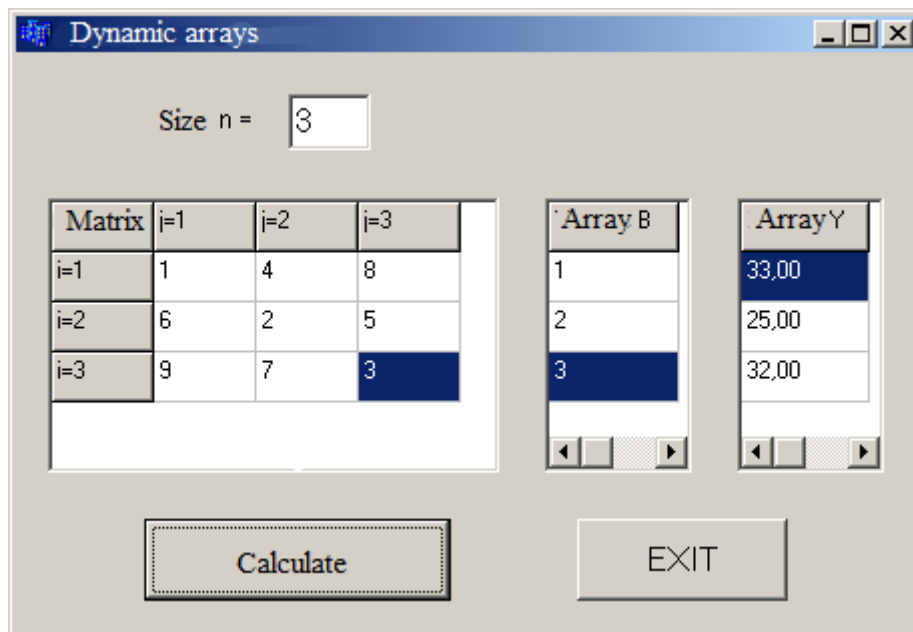


Fig.4.

Setting up the StringGrid component

For the *StringGrid1* component, set the *ColCount* and *RowCount* values to be equal, for example, 3 - three columns and three rows, and *FixedCols* and *FixedRows* - 1.

Since the *StringGrid2* and *StringGrid3* components have only one column, they have *ColCount* = 1, *RowCount* = 3, and *FixedCols* = 0 and *FixedRows* = 1.

In the *Options* property, set the *goEditing* line for the *StringGrid1* and *StringGrid2* components to true.

To change the size of n , use the *EditChange* handler function obtained by double-clicking on the *Edit* component.

The text of the program may look like this:

```

...
//----- Global variables -----
int n = 3;
double **a, *b; // Pointer declarations
//-----
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Edit1->Text=IntToStr(n);

```



```

StringGrid1->ColCount = n+1;    StringGrid1->RowCount = n+1;
StringGrid2->RowCount = n+1;    StringGrid3->RowCount = n+1;
// Entering the names of arrays in the upper left cell of the table
StringGrid1->Cells[0][0] = "Matrix A";
StringGrid2->Cells[0][0] = "Array B";
StringGrid3->Cells[0][0] = " Array Y";
for(int i=1; i<=n;i++){
    StringGrid1->Cells[0][i]="i="+IntToStr(i);
    StringGrid1->Cells[i][0]="j="+IntToStr(i);
} }

//-----
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    int i;
    n=StrToInt(Edit1->Text);
    StringGrid1->ColCount = n+1;    StringGrid1->RowCount = n+1;
    StringGrid2->RowCount = n+1;    StringGrid3->RowCount = n+1;
    for(i=1; i<=n;i++){
        StringGrid1->Cells[0][i]="i="+IntToStr(i);
        StringGrid1->Cells[i][0]="j="+IntToStr(i);
    } }

//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    double s;
    int i,j;
    a = new double*[n];           // Capturing memory for pointers
    for(i=0; i<n;i++) a[i] = new double[n];           // Capturing memory for elements
    b = new double[n];
    // Filling arrays A and B with elements from tables StringGrid1 and StringGrid2
    for(i=0; i<n;i++) {
        for(j=0;    j<n;j++)           a[i][j]=StrToFloat(StringGrid1-
>Cells[j+1][i+1]);
        b[i]=StrToFloat(StringGrid2->Cells[0][i+1]);    }
    // Multiplying a row of matrix A by vector B and outputting the result s to StringGrid
    for(i=0; i<n;i++){
        for(s=0, j=0; j<n;j++) s += a[i][j]*b[j];
        StringGrid3->Cells[0][i+1] = FloatToStrF(s, ffFixed,8,2);
    } }

//-----

```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    delete []a;
    delete []b;
    ShowMessage("Memory freed!");
    Close();
}
```

Control questions:

1. What is an index variable? Types of arrays and their differences.
2. Description of a regular type (array) in C++.
3. How are multidimensional arrays described in C++?
4. What types are used for index type and array component types?
5. Define a dynamic array.
6. What components are used to enter array elements?

MINISTRY OF SECONDARY AND SPECIAL EDUCATION OF THE REPUBLIC
OF UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY
NAMED AFTER ISLAM KARIMOV



Instructions and recommendations for topics for performing independent work

on the subject

“Information technologies in technical systems”

Tashkent 2022

Types, forms and themes of independent works.

Independent work is designed to teach students to independently perform specific educational work, search for and independently analyze the necessary information, as well as the formation and development of skills for making responsible decisions on this basis, as well as educational and methodological support and the full implementation of the study load for independent work, defined in the State educational standard for all areas of undergraduate studies.

To perform current control on independent work, the teachers of the department have developed options for independent work. For intermediate control, topics for independent work are compiled. In the final control, two questions out of five are compiled on the basis of the materials given in independent work.

Recommended topics for independent work:

- Hardware and features of network application systems;
- Software tools and characteristics of network application systems
- Features of modern publishing systems and their application in technical systems.
- Import and export data in MySQL system
- Application of best practices in the creation and processing of information in various fields;
- Solving engineering problems by branches in technical systems
- Data exchange of electronic documents;
- Creation of documents in Corel Draw;
- Using FTP on the Internet;
- Studying the creation and use of Web page tools;
- Sites. Creating and editing pages using HTML;
- The use of translation programs when translating texts in the specialty;
- Opportunities for distance learning, electronic textbooks;
- Programming of simple, branching and cyclic algorithms;
- Programming using structural data in file types;

Individual task

Create a presentation on a given topic (see below, the number of the topic option matches the student's serial number in the journal) in accordance with the requirements:

- the number of slides must be at least 15;
- the presentation should be meaningful;
- each of the presentation slides must have a unique markup;
- each of the slides must contain the "personal brand" of the student who created this presentation;
- The sample notes should explain the content and/or slide show;

- the presentation should have a slide - a table of contents, from where you could get to one of the sections (groups) of slides, and to each of the slides separately (use your own interactive or standard control buttons for implementation);
- from each of the presentation slides it should be possible to return to the slide table of contents;
- a unique transition shape must be used for each of the slides;
- on presentation slides it is not allowed to use repeated effects (sound and visual) of the appearance of slide elements until all available ones have been applied;
- at least one of the presentation slides must run an external program (exe or com file).

Presentation topics:

1. The history of the appearance of the computer
2. Computer architecture (from von Neumann to modern)
3. Monitors and video adapters.
4. Printers
5. Motherboards
6. Processors
7. Scanners
8. External storage media and storage devices
9. Sound cards and multimedia
10. Computer software structure
11. Windows architecture
12. Windows interface
13. Archiving programs and principles of archiving
14. Viruses and anti-virus programs
15. Text processing technology
16. Structured programming and its implementation in the programming language Pascal
17. Operating systems
18. Cryptography
19. Topology of computer networks
20. OLE technology
21. Drag&Drop technology
22. Data archiving
23. Databases
24. Integrated software packages

MINISTRY OF SECONDARY AND SPECIAL EDUCATION OF THE REPUBLIC
OF UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY
NAMED AFTER ISLAM KARIMOV



Glossary

on the subject

“Information technologies in technical systems”

Tashkent 2022

Glossary

Definitions		
English	Russian	Uzbek
<p>Automated information system (AIS) is an assembly of computer hardware, software, firmware, or any combination of these, configured to accomplish specific information-handling operations, such as communication, computation, dissemination, processing, and storage of information.</p>	<p>Автоматизированная информационная система (АИС) - организационно-техническая система, использующая автоматизированные информационные технологии в целях информационно-аналитического обеспечения научно-инженерных работ и процессов управления.</p>	<p>Avtomatlashtirilgan axborot tizimi (AAT)-Tashkiliy texnik tizim bo'lib, avtomatlashtirilgan axborot texnologiyalarini boshqarish jarayonida va ilmiy – muhandislik islarida axborot taxlil ta'minoti maqsadida ishlatiladi.</p>
<p>Automated Information System (AIS) An AIS is a combination of computer hardware and computer software, data, and/or telecommunications that performs functions such as collecting, processing, storing, transmitting and displaying information.</p>	<p>Автоматизированная информационная технология (АИТ)- информационная технология, в которой для передачи, сбора, хранения и обработки данных используются методы и средства вычислительной техники и систем связи.</p>	<p>Avtomatlashtirilgan axborot texnologiyalari (AATex) – Axborot texnologiyasi bo'lib, ma'lumotlarni yig'ish, saqlash, uzatish, qayta ishlashda ishlatiladigan usullar va xisoblash texnikasi vositalari va aloqa tizimidan iborat bo'ladi.</p>
<p>Automated Training System - system, which include the complex of teaching and learning materials (demonstrations, theoretical, practical,</p>	<p>Автоматизированная обучающая система - система, включающая комплекс учебно-методических материалов (демонстрационных, теоретических,</p>	<p>Avtomatlashtirilgan o'qitish tizimi-bu tizim o'quv jarayonini boshqaradigan o'quv-uslubiy kompleks materiallaridan (namoyish, nazariy, amaliy, nazorat) va</p>

control) and computer programs that control the learning process).	практических, контролирующих) и компьютерных программ, управляющих процессом обучения	kompyuter dasturlaridan tashkil topgan.
Automated Data Bank (ADB) - a set of database management system and a specific database (database) data located (are) under its control.	Автоматизированный банк данных (АБД) - совокупность системы управления базами данных и конкретной базы (баз) данных, находящейся (находящихся) под ее управлением.	Avtomatlashtirilgan ma'lumotlar banki (AMB) – Ma'lumotlar bazalarini boshqarish tizimlarining yig'indisi va ularning boshqaruvi ostidagi konkret ma'lumotlar bazasidir.
Security Administrator is a person who manually administers user access rights to systems. A workflow system may call on a Security Administrator to fulfill an approved request on systems where automated administration agents are not available or have not yet been configured.	Администратор безопасности - лицо или группа лиц, ответственных за обеспечение безопасности системы, за реализацию и непрерывность соблюдения установленных административных мер защиты и осуществляющих постоянную организационную поддержку функционирования применяемых физических и технических средств защиты.	Havfsizlik adminstratori – tizim havfsizligini ta'minlashga javobgar, adminstratorlar tomonidan o'rnatilgan himoya tizimini uzluksiz ishlashini kuzatuvchi va ximoyani ta'minlovchi fizikaviy, texnik vositalarni doimiy ishlashini tashkillashtiruvchi shaxs yoki shaxslar guruhidir.
WEB adress- Every computer connected to the internet has its unique web address, without which it cannot be reached by other computers. Also called universal resource locator	Адрес страницы - данные, точно определяющие логический адрес сайта или Web-страницы в Internet.	Sahifa manzili – saytning aniq mantiqiy manzili yoki Internetdagi Web-sahifa.

or Uniform Resource Locator (URL).		
Algorithm - is a set of instructions designed to perform a specific task.	Алгоритм - совокупность действий со строго определенными правилами выполнения.	Algoritm – bajarilish ketma-ketligi qat’iy qoidalarda aniqlangan amallar majmui.
Algorithmization-creating algorithm in order to solve tasks.	Алгоритмизация - составление алгоритмов для решения поставленных задач.	Algoritmlash – Berilgan masalani echish uchun algaritm tuzilishi.
A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images. Byte-a unit of computer information that is equal to eight bits.	База данных - единая система данных, организованная по определенным правилам, которые предусматривают общие принципы описания, хранения и обработки данных.	Ma’lumotlar bazasi – umumiy prinsiplar asosida tavsiflanadigan, saqlanadigan va qayta ishlanadigan, aniq qoidalar asosida tashkil qilingan umumiy ma’lumotlarning yagona tizimi.
Information security, sometimes shortened to InfoSec, is the practice of defending information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. It is a general term that can be used regardless of the form the data may	Безопасность информации - состояние информации, информационных ресурсов и информационных систем, при котором с требуемой вероятностью обеспечивается защита информации от утечки, хищения, утраты и т. д.	Axborotni havfsizligi – axborot, ahborot zaxiralari va axborot tizimlarida talab qilingan extimollikda axborot chiqib ketishidan, o’g’irlanishidan, yo’qotilishidan himoya ta’minlanadi.

take (e.g. electronic, physical).		
Browser - a program with a graphical user interface for displaying HTML files, used to navigate the World Wide Web.	Браузер (Browser) - средство просмотра. Более полно: программное обеспечение, предоставляющее графический интерфейс для интерактивного поиска, обнаружения, просмотра и обработки данных в сети.	Browser - ko'rish vositasi. To'liq: tarmoqda ma'lumotlarni interaktiv qidirush, kashf etish, ko'rish va qayta ishlash uchun grafik interfeys taqdim etadigan dastur ta'minoti.
A web client is an application that communicates with a web server, using Hypertext Transfer Protocol (HTTP)	Веб-клиент - программа, позволяющая пользователю запрашивать документы с веб-сервера.	Veb-klient – foydalanuvchiga veb-server hujjatlarni talab qilish imkonini beruvchi dastur.
A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as Web servers as well.	Веб-сервер - программа, запущенная на компьютере, предназначенная для предоставления документов другим компьютерам WWW, которые посылают соответствующие запросы.	Veb-server – shunday so'rovlarni jo'natuvchi boshqa kompyuterlar WWW xizmati uchun hujjatlarni taqdim etishga mo'ljallangan dastur.
Web page - a hypertext document connected to the World Wide Web.	Веб-страница - одиночный документ, содержащий гиперссылки, размещенный в WWW и определяемый с помощью адреса URL. Его можно открыть и просмотреть содержание с помощью	Веб-sahifa – WWWga joylashtirilgan, URL adres yordamida aniqlanadigan, gipermurojatli alohida hujjat. Uni browser yordamida ko'rish va ochish mumkin. Bu multimediuva hujjatlariga matn, grafika, tovush, video,

	программы просмотра - браузера.	animatsiya, gipermurojat va boshqalar kiradi.
Vector graphics is the use of polygons to represent images in computer graphics. Vectorgraphics are based on vectors, which lead through locations called control points or nodes.	Векторное изображение - это изображение, строящееся при помощи математического описания простых объектов - линий, окружностей, из которых создаются более сложные.	Vektorli tasvir - bu tasvir oddiy ob'ektlar: chiziqlar, doiralarni matematik ta'rifi yordamida barpo etadi va ular yordamida yanada murakkab tasvirlarni yaratadi.
White-board - an area on a display screen common to several users, on which they can write and draw.	Виртуальная аудиторная доска (белая доска) - электронная доска с возможностями непосредственного редактирования текста либо внесения соответствующих пометок поверх исходного текста с передачей этой информации на расстояние.	Virtual auditoriya doskasi (oq doska) – matnni bevosita taxrirlash imkoniyatiga ega bo'lgan yoki berilgan matnga kerakli belgilarni kiritishi mumkin bo'lgan va axborotni masofaga uzatuvchi elektron doska.
A Virtual Library is a collection of resources available on one or more computer systems, where a single interface or entry point to the collections is provided. The key point being that the user need not know where particular resources are located -- the location is "virtual".	Виртуальная библиотека - учебно-методическая и дополнительная литература, размещенная в глобальной сети Интернет.	Virtual kutubxona – global Internet tarmog'iga joylashtirilgan uquv-uslubiy va qo'shimcha adabiyot.
A virtual reality - the computer-generated simulation of a three-dimensional image or environment that can	Виртуальная реальность - новая технология бесконтактного информационного взаимодействия,	Virtual haqiqat - muloqotsiz axborot hamkorlikning yangi texnologiyasi bo'lib, kompleks multimediya amaliyot vositalari

<p>be interacted with in a seemingly real or physical way by a person using special electronic equipment, such as a helmet with a screen inside or gloves fitted with sensors.</p>	<p>реализующая с помощью комплексных мультимедиа-операционных сред иллюзию непосредственного вхождения и присутствия в реальном времени в стереоскопически представленном «экранном мире». Более абстрактно - это мнимый мир, создаваемый в воображении пользователя.</p>	<p>yordamida real vaqt oraliq'ida "dunyo ekrani"ga kirish ilyuzasini ta'minlaydi, Bu foydalanuvchi tasavvuridagi batafsil mavhum xayoliy dunyodir.</p>
<p>The virtual school differs from the traditional school through the physical medium that links administrators, teachers, and students.</p>	<p>Виртуальное учебное заведение - сообщество географически разделенных преподавателей и студентов, которые в процессе обучения общаются и взаимодействуют между собой с использованием электронных средств коммуникаций при минимальном или полностью отсутствующем личном, непосредственном контакте.</p>	<p>Virtual o'quv muassasasi – bir-biri haqida juda oz ma'lumotga ega bo'lgan, yoki umuman tanimagan, geografik jixatdan ajratilgan, o'qituvchi va talabalar jamiyati bo'lib ular electron kommunikatsion vositalar yordamida o'quv jarayonida muloqot va xamkorlik qiladilar.</p>
<p>Leased line is a private bidirectional or symmetric telecommunications line between two or more locations provided in exchange for a monthly rent. Sometimes known as a private circuit or data line in the UK. Hypermedia, a term derived from hypertext, extends the notion of</p>	<p>Выделенная линия - линия связи (канал передачи данных), установленная постоянно или на длительное время. Такой канал может называться также арендуемым, так как оборудование обычно принадлежит телекоммуникационным компаниям и сдается ими в аренду для исключительного пользования.</p>	<p>Ajratilgan liniya –doimiy yoki uzoq muddatga o'rnatilgan aloqa liniyasi (ma'lumotlar uzatish kanali). Telekommunikatsiya kompaniyalari tomonidan bunday kanallar ijaraga beriladi.</p>

<p>the hypertext link to include links among any set of multimedia objects, including sound, motion video, and virtual reality. It can also connote a higher level of user/network interactivity than the interactivity already implicit in hypertext.</p>		
<p>Hyperlink is a word, phrase, or image that you can click on to jump to a new document or a new section within the current document. Hyperlinks are found in nearly all Web pages, allowing users to click their way from page to page. Text hyperlinks are often blue and underlined, but don't have to be. When you move the cursor over a hyperlink, whether it is text or an image, the arrow should change to a small hand pointing at the link. When you click it, a new page or place in the current page will open.</p>	<p>Гиперссылка (Hyperlink) - элемент документа для связи между различными компонентами информации внутри самого документа, в других документах, в том числе и размещенных на различных компьютерах.</p>	<p>Gipermurojat – bir yoki turli kompyuterlarda joylashgan, hujjat ichidagi va tashqarisidagi turli axborot komponentalari orasidagi aloqa.</p>
<p>Hypertext - a software system that links topics on the screen to related</p>	<p>Гипертекст (Hypertext) - понятие, описывающее тип интерактивной среды с</p>	<p>Gipertekst- interaktiv muhit turini tasvirlash tushunchasi bo'lib, murojatlarga o'tish</p>

<p>information and graphics, which are typically accessed by a point-and-click method.</p>	<p>возможностями выполнения переходов по ссылкам. Ссылки (адреса форматаURL), внедренные в слова, фразы или рисунки, позволяют пользователю выбрать (установить указатель и нажать левую кнопку мыши) текст или рисунок и немедленно вывести связанные с ним сведения и материалы мультимедиа.</p>	<p>imkoniyatini bajaradi. So'zlar, iboralar yoki rasmlarga o'rnatilgan murojatlarni (URL manzil formati), foydalanuvchi (murojat ustida sichqonchanning chap tugmasini bosing) tanlashi matn yoki rasm va darhol tegishli ma'lumotlarni va multimedia materiallarini olib chiqish imkonini beradi.</p>
<p>Distance learning - a method of studying in which lectures are broadcast or classes are conducted by correspondence or over the Internet, without the student's needing to attend a school or college. Also called <i>distance education</i>.</p>	<p>Дистанционное обучение - обучение на расстоянии с использованием учебников, персональных компьютеров и сетей ЭВМ.</p>	<p>Masofaviy o'qitish – masofadan turib shaxsiy kompyuter va kompyuter termog'idan, darsliklardan foydalanib o'qitish.</p>
<p>Distance education - teaching system, which implements the method of distance learning educational qualification confirmation.</p>	<p>Дистанционное образование - педагогическая система, в которой реализуются способы дистанционного обучения с подтверждением образовательного ценза.</p>	<p>Masofaviy ta'lim- malaka oshirilganligini tasdiqlovchi pedagogic tizim bo'lib, unda masofaviy o'qitish usullari ishlatiladi.</p>
<p>Document - a piece of written, printed, or electronic matter that provides information or evidence or that serves as an official record.</p>	<p>Документ - информация, зафиксированная на материальном носителе, имеющая реквизиты, позволяющие ее идентифицировать.</p>	<p>Xujat – identifikatsiya imkonini beruvchi rekvizitlarga ega bo'lgan axborot tashuvchiga yozilgan ahborot.</p>

<p>Domain-a group of computers and devices on a network that are administered as a unit with common rules and procedures. Within the Internet, domains are defined by the IP address. All devices sharing a common part of the IP address are said to be in the same domain.</p>	<p>Домен (domain) - организационная единица в Интернете, служащая для идентификации узла или группы родственных узлов. Крупные домены могут подразделяться на поддомены, отражающие различные области интересов или ответственности.</p>	<p>Domen – Internetning tashkiliy birligi bo’lib, identifikatsion tugunlarga yoki qarindosh guruhlar tuguniga hizmat qiladi. Katta domenlar turli soxa qiziqishi yoki masulligini ifodalovchi domen oqiga bo’linadi.</p>
<p>Data protection - Data protection is the process of safeguarding important information from corruption and/or loss.</p>	<p>Защита информации - действия и средства по предотвращению утечки, хищения, искажения или подделки информации.</p>	<p>Axborot himoyasi – axborotni soxtalashtirish, buzish, o’g’rilash, noqonuniy tarqatishni bartaraf etish harakatlari va vositalari.</p>
<p>Internet - The global communication network that allows almost all computers worldwide to connect and exchange information. Some of the early impetus for such a network came from the U.S. government network Arpanet, starting in the 1960s.</p>	<p>Интернет (Internet) - открытая мировая информационная система, состоящая из взаимосвязанных компьютерных сетей, обеспечивающая доступ к удаленной информации и обмен информацией между компьютерами.</p>	<p>Internet – Kompyuterlar orasida axborot almashishini ta’minlaydigan, o’zaro bog’langan kompyuterlar tarmog’i bo’lib, ochiq jahon axborot tizimi tashkil qiladi.</p>
<p>An Internet service provider (ISP) is a company that provides customers with Internet access. Data may be transmitted using several technologies, including dial-up,</p>	<p>Интернет-провайдер (Internet Service Provider, ISP) - организация, предоставляющая пользователям доступ к Интернету.</p>	<p>Internet provayder (Internet Service Provider, ISP) –foydalanuvchilarga Internetga kirishni ta’minlovchi tashkilot.</p>

DSL, cable modem, wireless or dedicated high-speed interconnects.		
Information security, sometimes shortened to InfoSec, is the practice of defending information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. It is a general term that can be used regardless of the form the data may take (e.g. electronic, physical).	Информационная безопасность - состояние защищенности информационной среды, обеспечивающее ее формирование, использование и развитие в интересах граждан, организаций, государства.	Axborot xavsizligi – axborot muxitini himoyalash, fuqorolar, korxonalar, davlatlar axborotini ximoyalangan xolatini taminlash , rivojlantirish va undan foydalanish.
Information technology training - IT Training is specific to the Information Technology (IT) industry, or to the skills necessary for performing information technology jobs. IT training includes courses related to the application, design, development, implementation, support or management of computer-based information systems.	Информационная технология - система научных и инженерных знаний, а также методов и средств, которая используется для создания, сбора, передачи, хранения и обработки информации в предметной области.	Axborot texnologiyalari – ilmiy va muxandislik bilimlarini xamda usullari va vositalrini foydalanish va ularni ygish, uzatish ,saklash va kayta ishlash axborot tizimi majmuasidir.
Case study is an account of an activity, event or problem that	Кейс-технология - технология организации учебного процесса, при	Keys-texnologiyasi – ukuv jarayonida ishlatiladigan ixtiyoriy ukuv uslubiy

contains a real or hypothetical situation and includes the complexities you would encounter in the workplace. Case studies are used to help you see how the complexities of real life influence decisions.	которой учебно-методические материалы комплектуются в специальный набор (кейс) и передаются (пересылаются) студенту для самостоятельного изучения (с периодическими консультациями у назначенных ему преподавателей).	materiallar texnologiyasi va talabaga mustakil ishlarni berish va kabul qilish tuplami.
3D Studio Max – Application for modeling, animation and display of 3D models.	3D Studio Max – это программное обеспечение для моделирование, анимации и визуализации 3D моделей.	3D Studio Max - uch o'lchamli modellashtirish, animatsiya va ko'rsatish uchun dasturiy ta'minoti.
Kompas – Graphic application for diraining documents at verias ASAP and ESKD standatr	Компас – прикладная програма графики с возможностями оформления дизайнерских документов соответствующих серии ASAP стандарта ESKD	Kompas – ASAP seriyasi va ESKD standartlariga mufofiq dizayn qurilish hujatlarini pasmiylashtirish imkoniyatlari bilan kompyuter quvvat dizayn tizimlari oilasi
MICROMINE – multifunctional application program for 3D modelling tasks in mining engenering and processing information with dizaining aperation	MICROMINE –эта много фуункциональная прикладная программа для моделирования задач для горно-геологических ресурсов использующихся для информации горного разведовательного дела с операциями дизайна.	MICROMINE - ko'p funktsiyali tog'-geologik resurslari va zaxiralari uch o'lchovli blok modellari, tasnifi va miqdor qurish 3D muhitda ko'rinish va turli geologik ma'lumotlar talqin, tizimi, shuningdek kon operatsiyalari dizayni uchun.
JavaScript – Script langbge for creating dynamic sites at computer Network	JavaScript – скриптовый язые для создания динамических сайтов компьютерных сетях.	JavaScript – Netscape firmasi tomonidan ishlab chiqilgan skriptlardan tuzilgan yangi tildir. JavaScript yordamida siz ajoyib Web saxifalar yaratishingiz mumkin.

<p>Client-server architecture is common in both local and wide area networks. For example, if an office has a server that stores the company's database on it, the other computers in the office that can access the database are "clients" of the server.</p>	<p>Клиент (client) - программное обеспечение для доступа и получения данных при взаимодействии с программным обеспечением сервера, размещенного на другом компьютере.</p>	<p>Mijoz (client) – Kerak bulgan ma'lumot yoki resursga kirish uchun mijoz dastur ishga tushadi boshka kompyuterga ulanadi.</p>
<p>CD - A compact disc [sometimes spelled disk] (CD) is a small, portable, round medium made of molded polymer (close in size to the floppy disk) for electronically recording, storing, and playing back audio, video, text, and other information in digital form.</p>	<p>Компакт-диск - оптический диск, используемый для постоянного хранения информации больших объемов</p>	<p>Kompakt-disk – optik tolali disk, ma'lumotlarni saqlash uchun.</p>
<p>Computer graphics are simply images displayed on a computer screen. Graphics are often contrasted with text, which is comprised of characters, such as numbers and letters, rather than images.</p>	<p>Компьютерная графика - это создание, демонстрация и обработка графических изображений с помощью компьютера.</p>	<p>Kompyuter grafikasi – grafik ko'rinishdagi ma'lumotlarni kompyuter yordamida namoyish etish va qayta ishlash .</p>
<p>Multimedia - is the field concerned with the computer-controlled integration of text, graphics,</p>	<p>Мультимедиа (Multimedia) - компьютерные системы с интегрированной поддержкой звукозаписей и видеозаписей.</p>	<p>Multimedia (Multimedia) – tovushli va video yozuvlarni kompyuter tizimi orqali bevosita boshqarish.</p>

drawings, still and moving images (Video), animation, audio, and any other media where every type of information can be represented, stored, transmitted and processed digitally.		
---	--	--

MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY NAMED AFTER ISLAM
KARIMOV

« APPROVE »

Head of the department

«Information technologies»

_____ Sagatov M.V..

2022 « ____ » _____

Syllabus

on the subject

“Information technologies in technical systems”

for specialties and directions of training bachelors

of the Faculty of Mechanics

Discussed at a meeting of the Department of Information Technology

Protocol dated « ____ » _____ 2022 г., № ____

Tashkent – 2022

Name of the discipline:	“Information technologies in technical systems”
Subject type:	Mandatory
Subject code:	
Stage:	1
Semester:	1
Form of education:	Full-time
Form of classes and hours allotted per semester:	120
Lecture	30
Practical lessons	16
Laboratory lessons	14
Seminar	-
Self-studying	60
Amount of credits:	4
Form of education:	Full-time
Language of the subject:	English

Authors of the program:	Sagatov Miraziz Vorisovich, Tashmatova Shaxnoza Sobirovna, Kurbanova Kabira Erkinovna, Khamdamova Sevara Mirazizovna
E-mail:	informtgu@mail.ru
Phone number:	+99894 626 04 65
Organization:	TSTU named after Islam Karimov, department “Information technologies”

Brief information about the course (M)	
M1	The purpose of teaching the subject is to acquire knowledge on the use of methods and tools of modern information and communication technologies in the era of digital technologies and the acquisition of computer modeling skills, to lay the methodology for ensuring the cybersecurity of information systems and information resources, the principles of modern programming technology used in professional activities and the ability to apply the acquired knowledge and skills in solving specific tasks and problems of their specialty and makes it possible to use modern information technologies in various areas of professional activity, scientific and practical work. It considers: familiarization with modern techniques and methods of using ICT, IT, IS tools, their use in professional activities in the specialty,

	<p>familiarization with the possibilities of the basis of digital technologies, factors in the development of the digital economy; skills in solving modeling problems, using multimedia technologies, artificial intelligence systems, computer-based information systems that automate the input, accumulation, processing, transmission, operational management of information, the use of software tools for solving various problems in order to obtain a reliable result and extract maximum effect when using a computer system.</p> <p>The objective of the discipline is to teach students how to use a computer system and computer networks in solving problems of computer modeling, design, in the development of product design using computer-aided design systems, modern programming tools and technologies, to demonstrate the capabilities of information and communication technologies through the formation of a scientific worldview, to apply analysis methods studied phenomena, processes and design solutions in the development of IT technologies requiring legal solutions in situations, analyze the information security of objects and systems, conduct experiments according to a given methodology, process the results, evaluate the error and determine the reliability of the results, as well as manage cybersecurity in the development and implementation of solutions in the field of modern digital technologies.</p>
--	--

Initial requirements for the course	
1.	Maths, informatics and programming

Competences (learning outcomes) (LO)	
LO 1	the possibility of using technical, software and network resources of information technology in areas of specialty related to the use of information technology in technical systems.
LO 2	have ideas and knowledge about the modernization of digital infrastructure in order to develop the digital economy.
LO 3	use of the global information system;
LO 4	select and use analytical, numerical methods for the study of mathematical models, work with application packages for analytical and numerical study of mathematical models.
LO 5	document processing and problem solving in various CAD design systems.
LO 6	carrying out numerical experiments on a computer and a method for analyzing the results obtained;
LO 7	use of the graphical capabilities of application programs in the design.
LO 8	knowledge of the basics of algorithmization and methods of programming tasks in modern programming systems

Form of lesson: lecture (L)		Hours
L1	The subject of the ITTS course, its goals and objectives.	2
L2	Modern automated design systems and their use in technical systems	2
L3	Modeling. Computer modelling. Object-oriented modeling. Computational experiment. Software for computer design	2
L4	Mathematical modeling. Use of mathematical packages for the study and analysis of mathematical models, mathematical packages (3D Max, Solid Works, Matlab and MathCAD).	2
L5	Graphic modeling Processing of numerical and graphic information in engineering problems. Implementation of static and dynamic mathematical models in Matlab and MathCAD systems.	2
L6	Fundamentals of simulation modeling. Classification of software simulation tools. Simulation modeling using the Simulink package	2
L7	Programming in MATLAB. Modules and their functions. Methods for constructing graphical models in MATLAB.	2
L8	Information security systems. Mathematical foundations of cryptology. Issues of information security in computer networks.	2
L9	Cyberspace and the basics of cybersecurity. Computer technology objects used in cybersecurity.	2
L10	Modern programming technologies. Programming languages	2
L11	Object programming systems. Basic constructions of languages and features of programming in the system. Classes, methods and properties	2
L12	Logic programming technology. The logical structure of the program. Conditional, unconditional and select statements.	2
L13	Components used in visual programming. Loop operators. Their different forms Complex algorithms.	2
L14	Functions and modules. Standard and user-defined functions in programming languages. Local, static, dynamic variables.	2
L15	Application in systems of graphic and multimedia programming. Possibilities of the graphic module and their use.	2
Total		30
Form of lesson: Practice (P)		Hours
P1	Modern computer platforms and their technical features. Architecture and logical nodes of the computer. Requirements for hardware and software when using CAD systems.	2
P2	MathCad document for calculating the values of expressions. Representation of data by matrices. Working with vectors and matrices in MathCAD. Solving systems of equations	2

P3	Fundamentals of work in the MatLab system. System capabilities. Program interface. Creation of MatLab-document and their application in technical systems.	2
P4	Using the programming mode in the MatLab system to solve problems in mechanics.	2
P5	Cryptographic methods of information protection	2
P6	Mathematical modeling of the problem. Algorithmization of tasks. The use of programming constructs in solving engineering problems in mechanics	2
P7	Visual programming. Components used in visual programming in the field of mechanics	2
P8	Using classes and methods to work with graphical objects in programming.	2
Total		16
Form of lesson: lab (L)		Hours
L1	Basics of work in the MathCAD system System capabilities. Program interface. Creation of MathCad-document and their application in technical systems. Using simple functions.	2
L2	Exploring the graphical capabilities of the MathCAD system. Programming in the MathCAD system.	2
L3	Representation of data by matrices. Working with vectors and matrices in MatLab.	2
L4	Development and analysis of simulation models of technical objects Simulation modeling using the Simulink package	2
L5	The use of cryptographic elements in computer networks. Data backup and recovery policy. Logical control of access to data.	2
L6	Logical structure. Conditional, unconditional and select statements. Components used in visual programming. Cycle operators in programming in the field of mechanics.	2
L7	Programming of technical problems in an integrated environment for solving problems in the field of mechanics. Implementation of programming through modules and application of engineering problems to object-oriented programs.	2
Total		14

Learning strategy

Studying in information technology courses in technical systems based on the credit system of education includes lectures, practical and laboratory products, video lectures, presentations, as well as assignments and independent assignments. Materials on the indicated topics related to practice and laboratory work are given, the procedure for

conducting practical and laboratory work and calculating the results is explained. Course materials are studied by students on their own, practical and laboratory work, students perform tests individually.

The following materials are available to students:

- Video lectures;
- Lecture summary in electronic form;
- Presentation slides for each topic;
- Guidelines for practical and laboratory work;
- Assignments and control exercises for each topic;
- Educational aids and instructions in electronic form.

During theoretical classes, students will be provided with the necessary notes in the form of a video lecture. Students will be instructed to use presentations, textbooks, guides, and other study aids to further reinforce the topic. To check the level of assimilation of the topic by students, after each topic, a test control is carried out. If the student completes these tests at the required level, they will be allowed to move on to the next topic.

During practical classes, students are provided with materials, presentations, examples for solving problems on each topic, as well as tasks to check the level of mastering the topic.

Students who have fully mastered all the topics of lectures, practical and laboratory materials are allowed to participate in the final control. The student passes the final control at the end of the semester.

Student assessment

Assessment of students' knowledge is based on the study of educational materials (test, assignment and written results of work) during the semester and final control.

In the course "Information technologies in technical systems" students are evaluated on a 100-point scale. Of these, 50% of the points are allocated for attendance, assessment of the current and intermediate control, and 50% of the points relate to the final control. A student who scores 30 or more points on the final control is considered to have mastered the subject.

Current and final points are distributed as follows:

Tasks	Max score	
P1 tasks	1,5	maximum score for current control – 26 points
P2 tasks	2	
P3 tasks	2	
P4 tasks	2	
P5 tasks	1,5	
P6 tasks	1.5	
P7 tasks	2	
P8 tasks	1,5	
L1 tasks	2	
L2 tasks	2	
L3 tasks	1.5	

L4 tasks	1,5	
L5 tasks	2	
L6 tasks	1.5	
L7 tasks	1.5	
maximum attendance score	4	
maximum score for intermediate control	20	
maximum score on the final control	50	
Total:	100	100 points

Main educational literature	
1.	Кадиров М.М. Ахборот технологиялари. Ўқув қлланма, 1-қисм. -Т.:Сано-стандарт, 2018. - 320 б.
2.	Кадиров М.М. Техник тизимларда ахборот технологиялари. Дарслик, 2-қисм. -Т.:Ўзбекистон файласуфлари миллий жамияти, 2019. -306 б.
3.	Дадабаева Р.А., Насридинова Ш.Т., Шоахмедова Н.Х., Ибрагимова Л.Т., Ерматов Ш.Т. Ахборот-коммуникация технологиялари ва тизимлари. Ўқув қлланма. -Т.:Сано-стандарт, 2017, - 552 б
4.	Кенжабайев А.Т., Икромов М.М., Алланазаров А.Ш. Ахборот-коммуникация технологиялари. Ўқув қлланма. – Т.: Ўзбекистон файласуфлари миллий жамияти, 2017. - 408 б.
5.	Сангин В.Ф., «Комплексная защита информации в корпоративных системах», Учебное пособие. М.: ИД. «ФОРУМ» - ИНФРА М. 2019, 591с.
Extra educational literature	
1.	Назирова Ш.А., обулов Р.В., Бобожонов М.Р., Рахманов .С. С ва С++ тили. Т.: - Ворис-нашриёт, 2013. - 488 б.
2.	Кеннет С. Лаудон, Жане. П. Лаудон. Манагемент Информатион Системс: Манагинг те Дигитал Фирм, 13т Едитион, Пеарсон Едусатион, УСА 2014. П 621.
3.	Кунwoo Лее. Принциплес оф САД/САМ/САЕ: Те Сомпутер Аидед Енгинееринг Десигн Сериес. 5ст Едитион. Аддисон Веслей Лонгман, УСА, 2015..
4.	Алех Аллаин. Жумпинг инто С++. УСА, 2014. п 340.
5.	Азимджанова М.Т., Мурадова М.Т., Пазилов М.С. Информатика ва ахборот технологиялари. Ўқув қлланма. –Т.: Ўзбекистон файласуфлари миллий жамияти, 2013. -176 б.
6.	Арипов М., Доттойев С., Файзийева М. Веб технологиялари. Ўқув қлланма. – Т.: Ўзбекистон файласуфлари миллий жамияти, 2013. -280 б.

7.	Ганийев С.К., Каримов М.М., Ташев К.А. Ахборот хавфсизлиги. Дарслик. – Т.:Фан ва технология, 2017. - 372 б.
Internet sites:	
	www.ziynet.uz – таълим портали.
	www.lex.uz – Ўзбекистон Республикаси қонун ужжатлари маълумотлари миллий базаси.

**MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN
TASHKENT STATE TECHNICAL UNIVERSITY NAMED AFTER ISLAM
KARIMOV**

«A P P R O V E»

Vice Rector for Academic Affairs

_____ O.O. Zaripov

«_____» _____ 2022

**INFORMATION TECHNOLOGIES IN TECHNICAL SYSTEMS
WORKING CURRICULUM**

Field of knowledge: 700 000 – Engineering, manufacturing and construction industries.

Field of education: 710 000 – Engineering.
720 000 – Industrial and technological sphere.

Directions of education: Directions of education indicated in the field of education (for the Faculty of Mechanics)

Code and name of the direction of education:	Study load of students, hours								Semesters, hours
	Total load	Classroom activities						Self-studying	1
		Total	including						
			Lecture	Practical lessons	Laboratory lessons	Seminar	Course work		
5320100 - Materials science and technology of new materials (by industry);	120	60	30	16	14	-	-	60	120
5313600 – Metal forming machines	120	60	30	16	14	-	-	60	120
5322100 – Rolling production technology	120	60	30	16	14	-	-	60	120
5312800 – Foundry technologies	120	60	30	16	14	-	-	60	120
5320300 - Technological machines and equipment (by industry)	120	60	30	16	14	-	-	60	120
5314800 – Welding technology and equipment	120	60	30	16	14	-	-	60	120
5314900 - Machines and units of refrigeration and cryogenic equipment and air conditioning systems	120	60	30	16	14	-	-	60	120

Tashkent – 2022

SUBJECT/CODE OF MODULE MBIAF		Academic year 2022-2023	Semestr(s) 1	ECTS - Credits 4	
Subject/module type Mandatory		Language of education English		Hours in a week 4	
1.	Name of the subject	Classroom lessons (hours)	Self-studying (Hours)	Total volume (Hours)	
	Information technologies in technical systems	60	60	120	
2.	<p style="text-align: center;">The content of the discipline</p> <p style="text-align: center;">2.1 Purpose and objectives of the discipline</p> <p>The purpose of teaching the subject is to acquire knowledge on the use of methods and tools of modern information and communication technologies in the era of digital technologies and the acquisition of computer modeling skills, to lay the methodology for ensuring the cybersecurity of information systems and information resources, the principles of modern programming technology used in professional activities and the ability to apply the acquired knowledge and skills in solving specific tasks and problems of their specialty.</p> <p>The objective of the discipline is to teach students how to use a computer system and computer networks in solving problems of computer modeling, design, in the development of product design using computer-aided design systems, modern programming tools and technologies, to demonstrate the capabilities of information and communication technologies through the formation of a scientific worldview, to apply analysis methods studied phenomena, processes and design solutions in the development of IT technologies requiring legal solutions in situations, analyze the information security of objects and systems, conduct experiments according to a given methodology, process the results, evaluate the error and determine the reliability of the results, as well as manage cybersecurity in the development and implementation of solutions in the field of modern digital technologies.</p>				

Distribution of the total load by types of educational process								
№	Name of theme	Teaching load by forms of education, hours						
		Total load	Classroom lessons (hours)					Self-studying
			Total	Lecture	Lab	Practice	Course work	
1	Introduction to the subject "Information technologies in technical systems". The subject of the course, its goals and objectives. Main functions and tasks of ICT in technical systems. Principles for the implementation of ICT in technical areas, modernization of digital infrastructure in order to develop the digital economy.	8	4	2	-	2	-	4
2	Modern computer-aided design systems and their use in technical systems (CAD/CAE/CAM-systems). CAD in the field of mechanics. Classification of CAD by intended purpose. used in computer-aided design.	8	4	2	2	-	-	4
3	Modeling. The main types of modeling, their scope. Computer modelling. Classification of models. The principle of computer simulation. Expert systems.	8	4	2	-	2	-	4
4	Mathematical modeling. The use of mathematical packages for the study and analysis of mathematical models, mathematical packages (3D Max, Solid Works, Matlab and MathCAD).	8	4	2	2	-	-	4
5	Graphic modeling. Processing of numerical and graphic information in engineering problems. Implementation of static and dynamic mathematical models in Matlab and MathCAD systems.	8	4	2	-	2	-	4

6.	Fundamentals of simulation modeling. Classification of simulation software. Computational experiment. Varieties of simulation modeling. Simulation modeling using the Simulink package	8	4	2	2	-	-	4
7	Programming in MATLAB. Modules and their functions. Methods for constructing graphical models in MATLAB.	8	4	2	-	2	2	
8	Information security systems. Mathematical foundations of cryptology. Issues of information security in computer networks.	8	4	2	2	-	-	4
9	Cyberspace and the basics of cyber security. Computer technology objects used in cybersecurity.	8	4	2	-	2	-	4
10	Modern programming technologies. Programming languages.	8	4	2	2	-	-	4
11	Object programming systems. Basic constructions of languages and features of programming in the system. Classes, methods and properties.	8	4	2	-	2	-	4
12	Logic programming technology. The logical structure of the program. Conditional, unconditional and select statements.	8	4	2	2	-	-	4
13	Components used in visual programming. Loop operators. Their different forms (parametric, conditional check before and after). Complex algorithms.	8	4	2	-	2	-	4
14	Functions and modules. Standard and user-defined functions in programming languages. Local, static, dynamic variables.	8	4	2	2	-	-	4
15	Application in systems of graphic and multimedia programming. Capabilities of the graphic module and their use. Object animation, animation options	8	4	2	-	2	-	4
Total for the subject		120	60	30	14	16	-	60

2.2 The main theoretical part (lectures)

The content of the subject topics:

Topic 1. Introduction to the subject "Information technologies in technical systems". The subject of the course, its goals and objectives. Main functions and tasks of ICT in technical systems. Principles for the implementation of ICT in technical areas, modernization of digital infrastructure in order to develop the digital economy. Spheres of application of information technologies at the present stage of development of science and technology. Information, computer and applied ethics as theoretical components of the ethics of the global communication space.

Topic 2. Modern automated design systems and their use in technical systems.

Modern computer-aided design systems and their use in technical systems (CAD/CAE/CAM-systems). CAD in the field of mechanics. Classification of CAD by purpose, classification of models and parameters used in computer-aided design.

Topic 3. Modeling. Basic types of modeling. Computer modelling.

Modeling. Model, the concept of a model, a general property of a model. Classification of models. The main types of modeling, their scope. Computer modelling. The principle of computer simulation. Expert systems.

Topic 4. Mathematical modeling. Basic concepts and principles of mathematical modeling. The main stages of the method of mathematical modeling.

Mathematical modeling. Basic concepts and principles of mathematical modeling. The main stages of the method of mathematical modeling. The use of mathematical packages for the study and analysis of mathematical models, mathematical packages (3D Max, Solid Works, Matlab and MathCAD).

Topic 5 Graphical modeling Processing of numerical and graphical information in engineering problems.

Graphical modeling Processing of numerical and graphic information in engineering problems Implementation of static and dynamic mathematical models in 3D Max, Solid Works, Matlab and MathCAD systems. Mathcad graphics,

Topic 6. Simulation modeling. Simulation modeling using the Simulink package

Fundamentals of simulation modeling. Classification of simulation software. The role and place of simulation modeling in the study of complex systems. The essence of simulation modeling. Object-oriented modeling. Computational experiment. Varieties of simulation modeling. Simulation modeling using the Simulink package

Topic7. Programming in MATLAB. Modules and their functions. Methods for constructing graphical models in MATLAB.

Topic 8. Information security systems. Issues of information security in computer networks. Mathematical foundations of cryptology. Issues of information

security in computer networks. Cryptographic methods of information protection. Electronic digital signature

Topic 9. Cyberspace and the basics of cybersecurity. Objects of computer technologies used in ensuring cybersecurity Comprehensive assessments of the security of automated information and telecommunication systems. Organization and management of cybersecurity in the digitalization of the company's internal processes (service provision, operations, etc.), cybersecurity management in the development and implementation of solutions in the field of modern digital technologies.

Topic 10. Modern programming technologies. Programming languages. Modern approach to programming. Systems of object-oriented programming. Modern programming languages. Programming languages and systems, features of their use and classification.

Topic 11. Systems of object programming. Programming systems The main constructions of languages and features of programming in the system. Classes, methods and properties. The concept of a class. Type class. Base classes. Properties and their application to objects. Constructors and destructors. The object and its components. Operators, types, procedures. The structure of the program project. Components of the program. Examples of programs for solving engineering problems in mechanics.

Topic 12. Logic programming technology. Logical structure of the program The logical structure of the program. Conditional, unconditional and select statements. Components used in visual programming. Loop operators. Their different forms (parametric, conditional check before and after).

Topic 13. Visual programming. Components used in visual programming. Methods of visualization of technical problems. Operators. Their different forms (parametric, conditional check before and after). Complex algorithms.

Topic 14. Functions and modules. Using Functions and Modules in Programming

Standard and custom functions. Implementation of programming through modules and application of engineering problems to object-oriented programs. Application of functions on practical examples. Local, static, dynamic variables. Application of structured programs in technical systems. Programming with arrays. Working with dynamic arrays.

Topic 15. Application in graphical and multimedia programming systems. Application in systems of graphic and multimedia programming. Capabilities of the graphic module and their use. The role and essence of visualized programming in technical systems. Object animation, animation options Classes and methods for working with graphic objects. Animation of objects.

2.3. Instructions and recommendations for the implementation of practical work.

In practical classes, students get acquainted and study the architecture and main platforms of a computer system, modern software tools, the principles of using a software package, applications for creating and processing graphic objects, the implementation of mathematical models in MatLab, MathCad systems. Fundamentals of algorithmization, tools for creating programs in languages for the implementation of algorithms for solving engineering problems in OOP.

To conduct practical classes, the teaching staff of the department develops methodological instructions, options for tasks. In practical classes, students strengthen knowledge and acquire skills acquired in lectures. It also provides for the strengthening of students' knowledge through the use of textbooks and electronic lessons.

Approximate list of practical lessons:

1. Modern computer platforms and their technical features. Computer architecture. Logical nodes of the computer. Execution of programs using CAD systems. Requirements for hardware and software when using CAD systems.
2. Working with vectors and matrices in MathCAD. Solving systems of equations
3. Basics of work in the MatLab system. System capabilities. Program interface. Creation of MatLab-document and their application in technical systems.
4. Using the programming mode in the MatLab system to solve problems in mechanics.
5. Cryptographic methods of information protection.
6. Mathematical modeling of the problem. Algorithmization of tasks. The use of programming constructs in solving engineering problems in mechanics
7. Visual programming. Components used in visual programming. Loop statements in mechanical programming
8. Using classes and methods to work with graphical objects in programming.

2.4. Instructions and recommendations for the performance of laboratory work.

The purpose of laboratory works is the assimilation and consolidation of theoretical knowledge by students obtained in lectures and practical classes on the subject, and the acquisition of dialogue skills with a computer is provided. Namely, in laboratory classes, students learn and acquire the skills to use devices (multimedia tools, a scanner, etc.), purposefully apply the necessary software applications in a specific area, taking into account their specialty, and also acquire skills and abilities for processing information in technical systems and implementing engineering tasks to create electronic documents for their specialty. They get acquainted and master the technology of applying mathematical and simulation modeling, visual design in CAD

systems, study programming and implementation of complex algorithms and the basics of an object-oriented language.

Approximate list of laboratory lessons:

1. Fundamentals of work in the MathCAD system System capabilities. Program interface. Creation of MathCad-document and their application in technical systems. Using simple functions.
2. Studying the graphic capabilities of the MathCAD system. Programming in the MathCAD system.
3. Data representation by matrices. Working with vectors and matrices in MatLab.
4. Development and analysis of simulation models of technical objects Simulation modeling using the Simulink package
5. Use of elements of cryptography in computer networks. Data backup and recovery policy. Logical control of access to data.
6. Logical structure. Conditional, unconditional and select statements. Components used in visual programming. Loop statements in mechanical programming
7. Programming of technical problems in an integrated environment for solving problems in the field of mechanics. Implementation of programming through modules and application of engineering problems to object-oriented programs.

2.5. Instructions and recommendations for the implementation of the course work (project)

Coursework (project) is not provided for by the curriculum.

2.6. Instructions and recommendations for the performance of independent work.

Independent work is designed to teach students to independently perform specific educational work, search for and independently analyze the necessary information, as well as the formation and development of skills for making responsible decisions on this basis, as well as educational and methodological support and the full implementation of the study load for independent work, defined in the State educational standard for all areas of undergraduate studies.

To perform current control on independent work, the teachers of the department have developed options for independent work. For intermediate control, topics for independent work are compiled. In the final control, two questions out of five are compiled on the basis of the materials given in independent work.

Recommended topics for self-study:

1. The role of information and communication technologies in the development of the digital economy.
2. The main directions of development of the information and communication sphere in Uzbekistan, current laws, decrees of the President of the Republic of Uzbekistan and resolutions of the Cabinet of Ministers.

	<p>3. Trends in the development of system and application software.</p> <p>4. The role of expert systems in management and their application in economic sectors.</p> <p>5. Prospects for the use of intelligent control systems in the field of robotics.</p> <p>6. Application of computer-aided design systems in electronics, mechanics, mechanical engineering and other areas.</p> <p>7. Technologies used in the design of 3D graphics capabilities.</p> <p>8. International documents on cybersecurity and the experience of foreign countries.</p> <p>9. Information security in information communication systems.</p> <p>10. Creation of non-standard modules and their use in the program.</p> <p>11. Application systems based on web programming.</p> <p>12. Students are encouraged to prepare and submit an essay on independently mastered topics.</p>
3.	<p>Learning outcomes of the discipline (competencies)</p> <p>As a result of mastering the subject, the student:</p> <ul style="list-style-type: none"> • understands and knows the concepts and foundations of digital technologies, factors in the development of the digital economy; • acquires skills in solving modeling problems, knows the features of design automation and can use CAD systems for the tasks of his specialty. • must know the content and technologies of programming, their use in the fields of technology and be able to make decisions in problems related to the use of information and communication technologies.
4.	<p>Educational technologies and methods:</p> <ul style="list-style-type: none"> • lectures; • interactive case studies; • seminars (logical thinking, quick questions and answers); • work in groups; • prepare presentations and video tutorials. • individual projects; • Projects for collaboration and advocacy.
5.	<p>Literature:</p> <p>5.1. Main literature</p> <p>1. Информатика. Базовый курс. 2-е издание./ Под ред. Симонович С.В.- СПб., Питер, 2005-640 с. ил.</p> <p>2. Учебник для вузов.</p> <p>3. Павловская Т.А. С/С++. Программирование на языке высокого уровня. Учебник для вузов.- СПб., Питер, 2003-464 с. ил</p>

4. Kadirov M.M. Axborot texnologiyalari. O‘quv qo‘llanma, 1-qism. –T.: “Fan va texnologiya”, 2018. -316 b.
5. Kadirov M.M. Texnik tizimlarda axborot texnologiyalari. Darslik, 2-qism. –T.: “Fan va texnologiya”, 2018. -306 b.
6. Kenneth C. Laudon, Jane. P. Laudon. Management Information Systems: Managing the Digital Firm, 13th Edition, Pearson Education, USA 2014. p 621.
7. Faithe Wempen. Computing Fundamentals IC3 EDITION. John Wiley & Sons Ltd, United Kingdom. 2014. P 722.
8. Beth Melton. Microsoft Office Professional 2013. Step by Step. USA 2013. p 1184.
9. Kunwoo Lee. Principles of CAD/CAM/CAE: The Computer Aided Engineering Design Series. 5st Edition. Addison Wesley Longman, USA, 2015.
10. Alex Allain. Jumping into C++. USA, 2014. p 340.
11. Nazirov Sh.A., Qobulov R.V., Bobojonov M.R., Rahmanov Q.S. C va C++ tili. Darslik. –T.: “Voriz”, 2013. -488 b.

5.2. Extra literature:

12. Мирзиёев Ш.М. Критический анализ, строгая дисциплина и личная ответственность должны быть повседневным правилом каждого руководителя. Выступление Президента Республики Узбекистан на заседании Кабинета Министров Республики Узбекистан об итогах 2016 года и перспективах на 2017 год. // Газета "Народное слово". 2017, 16 января, №11.
13. Конституция Республики Узбекистан. - Т.: Узбекистан, 2017. - 46 с.
14. Аюпов Р.Х., Болтабоева Г.Р. Инновационные методы и инструменты обучения. узб. яз. ТМИ, 2014. -160 с.12. Петров М.Н., Молочков В.П. Компьютерная графика. Учебник для вузов. -СПб: Питер,2003,736 с.
15. Попов В.Б. Практикум по интернет технологиям. Учебный курс- СПб., 16. Питер,2005-480с.ил.
17. Кренке Д. Теория и практика построения баз данных; перев.с англ.- СПб., 18. Питер,2003-800с.ил

5.3. Informational resources.

19. www.gov.uz – Государственный портал Республики Узбекистан.
20. www.lex.uz–Национальная база законодательства Республики Узбекистан.
21. www.ru.wikipedia.org
22. <http://www.intuit.ru/departament/informatics/intinfo/>
23. <http://www.dstu.edu.ru/informatics/mtdss/index.html>

6. The model program was approved by the Tashkent State Technical University protocol No. ____ dated ” ____ ” _____ 2022.

7. **Responsible for the subject (module):**

Sagatov M.V. - TSTU, head of the department "Information technologies" doctor of technical sciences, prof.
 Karimova D. - TSTU, associate professor of the department "Information technologies"
 Kadirov M.M. - TSTU, Associate Professor of the Department "Information technologies"
 Tashmatova Sh.S. - TSTU, senior lecturer of the department "Information technologies".

8. **The working curriculum was discussed at a meeting of the Department of Information Technology of the Faculty of Electronics and Automation and recommended to the Faculty's Educational and Methodological Council (Protocol No. _____ dated " ____ " _____ 2022).**

Head of the Department **Sagatov M.V.**

Secretary **Akbarova Sh.A.**

The working curriculum was considered at a meeting of the Educational and Methodological Council of the Faculty of Mechanics and recommended to the Scientific and Methodological Council of the University (Protocol No. _____ dated " ____ " _____ 2022).

Chairman of the educational and methodological council of the faculty _____

Secretary _____

The working curriculum was reviewed and approved by the Scientific and Methodological Council of the Tashkent State Technical University (Protocol No. _____ dated " ____ " _____ 2022).

Secretary **N. Mambetov**

**MINISTRY OF HIGHER AND SECONDARY SPECIAL EDUCATION OF THE
REPUBLIC OF UZBEKISTAN**

TASHKENT STATE TECHNICAL UNIVERSITY



Tests

on the subject

“Information technologies in technical systems”

Tashkent 2022

1. Hardware and software

All information can be processed by a computer if it is presented:

in binary sign system*

in decimal sign system

in the form of symbols and numbers

only in the form of characters of the Latin alphabet

2. The data is:

information that is processed by a computer in binary computer code

a sequence of instructions that a computer executes in the process of processing data

numeric and text information

sound and graphic information

3. The program is:

information that is processed by a computer in binary computer code

a sequence of instructions that a computer executes in the process of processing data

numeric and text information

sound and graphic information

4. Processes data in accordance with a given program:

CPU

Input Devices

RAM

output devices

5. During processing, the program and data must be loaded:

into working memory

into permanent memory

into long-term memory

6. The number of bits perceived by the microprocessor as a whole is:

processor capacity

clock frequency

computer internal memory

computer performance

7. The number of cycles per second is:

processor capacity

clock frequency

computer internal memory

computer performance

8. The program for testing, setting the necessary parameters for the equipment used in this computer and loading the operating system is located:

in RAM

in permanent memory

in long-term memory

9. For long-term storage of information, the following is used:

1) external memory

2) RAM

3) permanent memory

10. Write-once discs:

1) CD-ROM and DVD-ROM

2) CD-R and DVD-R

3) CD-RW and DVD-RW

11. Rewritable discs:

1) CD-ROM and DVD-ROM

2) CD-R and DVD-R

3) CD-RW and DVD-RW

12. Read-only drives:

1) CD-ROM and DVD-ROM

2) CD-R and DVD-R

3) CD-RW and DVD-RW

13. A non-volatile type of memory that allows you to record and store data in microcircuits:

1) hard drive

2) floppy disk

3) laser disc

4) flash memory

14. Information input devices include:

1) keyboard

2) monitor

3) mouse

4) scanner

5) modem

15. Output devices include:

- 1) monitor
- 2) scanner
- 3) mouse
- 4) modem
- 5) printer

16. A device capable of reading graphic information and converting it into digital form is:

- 1) monitor
- 2) scanner
- 3) mouse
- 4) modem
- 5) printer

17. Devices that allow you to receive video images and photographs directly in digital (computer) format are:

- 1) monitor
- 2) scanner
- 3) mouse
- 4) digital cameras
- 5) printer

18. Device for displaying text and graphic information:

- 1) monitor
- 2) scanner
- 3) mouse
- 4) modem
- 5) printer

19. Device for printing text and graphic information on paper:

- 1) monitor
- 2) scanner
- 3) mouse
- 4) modem
- 5) printer

20. A device for entering numerical and textual information into a computer:

- 1) monitor
- 2) scanner
- 3) keyboard
- 4) modem
- 5) printer

21. To connect a computer to a local network, use:

- 1) network card
- 2) modem
- 3) joystick
- 4) touch panel
- 5) graphics tablet

22. To connect a computer to a telephone line for transmitting and receiving information over a long distance, use:

- 1) network card
- 2) modem
- 3) joystick
- 4) touch panel
- 5) graphics tablet

23. Programs designed for the operation and maintenance of computers:

- 1) systemic
- 2) Programming systems
- 3) applied

24. Operating systems are ... programs:

- 1) systemic
- 2) Programming systems
- 3) applied

25. Device drivers are ... programs:

- 1) systemic
- 2) Programming systems
- 3) applied

26. Antivirus programs are ... programs:

systemic

programming systems

applied

27. Programs that the user uses to solve various problems without resorting to programming:

systemic

programming systems

applied

?Which panel is used to insert mathematical symbols and operators into documents?

+ Math

= Controls

= Evaluation

= Standard

?MathCAD window elements include...

+title bar, menu bar, toolbar buttons, worksheet window, status bar

= character conversions, toolbar buttons, program name

= toolbar buttons, program name, context menu

= title bar, program name, context menu

? Which panel is used to insert templates of integration, differentiation, summation?

+Calculus

=Graph

=Evaluation

=Matrix

? What is "+" in MathCAD document?

+ input cursor

= input lines

= character placeholder

= mouse pointer

? Dividing X by Z in MathCAD:

+ $\frac{X}{Z}$

Z

= X / Z

= $X : Z$

= $X \setminus Z$

? How to enter Latin numerals into a mathematical expression?

- + typing on the keyboard
- = using the Greek toolbar
- = using panel panel graphic
- = using panel symbolic panel

? In MathCAD there are:

- + local and global variables
- = global and simple variables
- = local and simple variables
- = complex and simple variables

? Find the correct notation for the mathematical expression: $A=(x+3)/2$ in the MathCad package:

- + $A = \frac{x+3}{2}$
- = $A=(x+3)/2$
- = $A=2/x+3$
- = $A=3+x/2$

? Which keyboard shortcut cuts parts of a formula to the clipboard?

- + Ctrl+X
- = Ctrl+C
- = Shift+V
- = Shift+S

? To insert a hyperlink, use the "Insert / Hyperlink" command?

- + True
- = False
- = Shift+C
- = Ctrl+C

? Which panel contains arithmetic operators?

- + Calculator Toolbar
- = Matrix Toolbar
- = Greek Symbol Toolbar
- = Graph Toolbar

? How to enter the number "pi" on the keyboard?

- + Ctrl+Shift+p
- = ++j +Shift
- = Ctrl+Shift+z
- = Ctrl+Shift+e

? How to place two charts on one template?

- + having typed the name of the first function on the Oy axis, press the comma key and enter the name of the second function
- = having typed the name of the function on the Oy axis, press Enter and enter the name of the second function
- = having typed the name of the function on the Oy axis, press the spacebar and enter the name of the second function
- =by typing on the y-axis the name of the function, press Page Down and the name of the second function

? What does a red star mean next to an expression in MathCAD?

- + looking for a symbolic solution
- = contains an error
- = result output
- =input variables

? What is the keyboard shortcut for the equal sign?

- + Ctrl+<=>
- = Ctrl+<;>
- = Alt+<.>
- = Alt+<:>

? Which of the operations cannot be performed symbolically with a matrix in MathCAD

- + define parameters
- = compute determinant
- = find inverse matrix
- = transpose

? Which of the signs cannot be used inside the block for solving a system of equations?

- + :=
- = ≤

= \geq
= =

? What is the keyboard shortcut for the matrix template?

+ Ctrl+m
= Ctrl+v
= Alt+v
= Alt+n

? How many toolbars in MathCAD

+3
=4
=5
=6

? To enter the derivative of a function, you must use

+ Calculus
= Evaluation
= Boolean
= Calculator

? In the optimization problem, the function to be optimized is introduced using the sign:

+ :=
= =
= +
= :

? The matrix is given in the form: $i=1..5, R_{\{i\}}=3 \cdot i$. How many elements are in the matrix?

+ 5
= 7
= 8
= 6

? Choose the correct assignment operator

+ $k:=p+a*\cos(a)$
= $g = a +b +c$
= $a:= b**c$

= $y := \cos x + \text{gamma}$

? **Choose the correct assignment operator**

+ $b := \text{atan}(2.5) + \log(5)$

= $b := \text{arctg}(2.5) + \log(5)$

= $b := \text{atg}(2.5) + \log(5)$

= $b := \text{arctan}(2.5) + \lg(5)$

? **What is the ORIGIN system variable used for?**

+ To set the initial value of the element number in the array

= To find an original solution to the problem

= To set the accuracy of calculations

= To determine the dimension of the matrix

? **Given two vectors V1 and V2 containing 8 elements each. What will be the result of this operation**

+ Matrix with 8 rows and 8 columns

= vector of 8 elements

= one number

= number 64

? **What types of areas does a Mathcad document consist of?**

+ from computing, graphic and text

= from text and sound

= from graphic and slide

= from slide and sound

? **MathCAD allows you to create and edit files with the extension ...**

+ mcd

= txt

= rtf

= mp3

? **Check the built-in functions of MathCAD that can be called using the toolbar "Calculator" ("Calculator").**

+ sin

= Isolve

= solve

= root

? Check the operators that are used to assign a value to a variable in MathCAD.

- + :=
- = =
- = :
- = -

? Check the operators that are used in MathCAD to set the range of values.

- + . .
- = *
- = :=
- = =

? The MathCAD package is intended for:

- + perform mathematical calculations
- = work with graphic files
- = creating text documents
- = create presentations

? What type of special-purpose programs does the MathCad package belong to:

- + math packages
- = automatic systems
- = expert systems
- = create presentations

? What is the name of the MathCad program document:

- + worksheet;
- = workbook;
- = workspace.
- = workplace.

? With the help of which in the MathCad program the values of the letter designations are set:

- + using the assignment operator;
- = using a special control panel;
- = using function keys
- = using the Enter key.

? In the MathCad "Insert Matrix" dialog box, you can define a matrix as follows:

- + set the number of rows and columns, and click OK;
- = select the template of the required matrix;
- = choose the number of rows and columns in the $m \times n$ matrix.
- = using the Enter key

? What graphs can be built in MathCad:

- + 2D and 3D
- = 3D only
- = 2D only
- =multidimensional

? What needs to be done to format a graph in MathCad:

- + double click in the graph area;
- = execute command Insert;
- = Right-click and select Format.
- = Right-click and select Standard.

? Files with extension . mcd refer to:

- + MS MathCAD
- = MS PowerPoint
- = MS Word
- = MS Excel

? MathCAD allows you to create and edit files with the extension ...

- + .mcd
- = .txt
- = . rtf
- = . Mp3

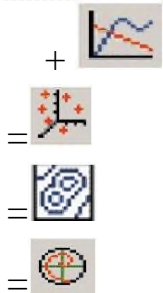
? The MathCAD Math Panel does not contain a button:

- +programming panel
- = symbolic computation keywords
- =bar of trigonometric functions
- =calculator

? To build two graphs in the same coordinate system, both functions are entered in the expression window, between which the sign ?

- + ,
- ;
- /
- :

? In order to build a graph of the function $f(x)$ in a rectangular Cartesian coordinate system, you need to select the button in the graphs panel?



? What are arithmetic operators used for?

- + to perform arithmetic calculations
- = for plotting;
- = to solve logical problems;
- = for solving static problems;

? What are relativity operators used for?

- + to compare numeric operands
- =for plotting;
- = to solve logical problems;
- = for solving static problems;

? What are logical operators used for?

- + to create boolean expressions
- = to perform arithmetic calculations.
- = for solving static problems;
- = for plotting

? The main toolbars are:

- +Standard panel; Formatting panel; Math panel
- = Panel Standard; Formatting panel; Drawing panel
- = Formatting panel; Drawing panel; Math panel

= Math panel; Standard panel; Drawing panel

? To create a new document in Mathcad, the function buttons are used:

+Ctrl+N

=Ctrl+O

=Ctrl+W

=Ctrl+P

? How to save a document created in Mathcad:

+Ctrl+.W

=Ctrl+O

=Ctrl+H

=Ctrl+P

? What keys can be used to activate the matrix window in Mathcad

+Ctrl+M

=Ctrl+O

=Ctrl+H

=Ctrl+P


? Which keys can be used to activate the window for inserting functions in Mathcad:

+Ctrl+E

=Ctrl+O

=Ctrl+H

=Ctrl+P


? what is the  panel used for?

+ for building two- and three-dimensional graphs

= to enter logical operators

= to enter vectors and matrices

= to enter Greek letters

? what is the  panel used for?

+ to build two-dimensional graphs;

= To enter logical operators

= To enter vectors and matrices

= To enter Greek letters

? what is the  panel used for?

- + for scaling Cartesian coordinates
- = for 2D plotting
- =for 3D plotting
- = to enter Greek letters

? what is the  panel used for ?

- +for tracing the displayed coordinates of the selected graph
- = to enter Greek letters
- = for 2D plotting
- =for 3D plotting

? what is the  panel used for ?

- + for building polar graphs
- = to enter Greek letters
- =for 3D plotting
- = for 2D plotting

? what is the  panel used for ?

- + for surface plotting
- = to enter Greek letters
- =for 3D plotting
- = for 2D plotting

? what is the  panel used for ?

- + to set the graph level line
- = to enter Greek letters
- =for 3D plotting
- = to build two-dimensional graphs;

? what is the  panel used for?

- + for surface plotting
- = to set characters
- = to enter Greek letters
- = for 2D plotting



? **what is the  panel used for?**

- + for building 3D graphs
- = to set characters
- = to enter Greek letters
- = for 2D plotting



? **what is the  panel used for?**

- + to enter vectors
- = to set characters
- = to enter Greek letters
- = for 2D plotting

? **What is " " in a MathCAD document?**

- + character placeholder
- = assignment operator
- = input cursor
- = mouse pointer

? **Choose the correct assignment operator**

- + $c := \text{atan}(8.3) + \log(15)$
- = $c := \text{arctg}(8.3) + \log(15)$
- = $b := \text{atg}(8.3) + \lg(15)$
- = $c := \text{arctan}(8.3) + \lg_5(15)$

? **Check the built-in functions of MathCAD that can be called using the toolbar "Calculator" ("Calculator").**

- + cos
- = Isolve
- = solve

MINISTRY OF SECONDARY AND SPECIAL EDUCATION OF THE REPUBLIC OF
UZBEKISTAN

TASHKENT STATE TECHNICAL UNIVERSITY
NAMED AFTER ISLAM KARIMOV



Literature

on subject

“Information technologies in technical systems”

Tashkent 2022

Main literature

1. Kadirov M.M. Axborot texnologiyalari. O‘quv qo‘llanma, 1-qism. –T.: “Fan va texnologiya”, 2018. -316 b.
2. Kadirov M.M. Texnik tizimlarda axborot texnologiyalari. Darslik, 2-qism. –T.: “Fan va texnologiya”, 2018. -306 b.
3. Kenneth C. Laudon, Jane. P. Laudon. Management Information Systems: Managing the Digital Firm, 13th Edition, Pearson Education, USA 2014. P 621.
4. Faithe Wempen. Computing Fundamentals IC3 EDITION. John Wiley & Sons Ltd, United Kingdom. 2014. P 722.
5. Beth Melton. Microsoft Office Professional 2013. Step by Step. USA 2013. P 1184.
6. Kunwoo Lee. Principles of CAD/CAM/CAE: The Computer Aided Engineering Design Series. 5st Edition. Addison Wesley Longman, USA, 2015.
7. Alex Allain. Jumping into C++. USA, 2014. p 340.
8. Nazirov Sh.A., Qobulov R.V., Bobojonov M.R., Raxmanov Q.S. C va C++ tili. Darslik. – T.: “Voriz”, 2013. -488 b.

Extra literature

1. Mirziyoyev Sh.M. Tanqidiy tahlil, qat’iy tartib-intizom va shaxsiy javobgarlik – har bir rahbar faoliyatining kundalik qoidasi bo‘lishi kerak. O‘zbekiston Respublikasi Vazirlar Mahkamasining 2016 yil yakunlari va 2017 yil istiqbollariga bag‘ishlangan majlisidagi O‘zbekiston Respublikasi Prezidentining nutqi. // “Xalq so‘zi” gazetasi. 2017 y., 16 yanvar, №11.
2. O‘zbekiston Respublikasi Konstitutsiyasi. - T.: O‘zbekiston, 2017. - 46 b.
3. Ayupov R.X., Boltaboyeva G.R. Innovatsiyon ta’lim usullari va vositalari. TMI, 2014. - 160 b.

Internet sites

1. www.gov.uz – Government portal of the Republic of Uzbekistan.
2. www.lex.uz– National database of information on legal documents of the Republic of Uzbekistan.
3. www.ru.wikipedia.org
4. <http://www.intuit.ru/department/informatics/intinfo/>
5. <http://www.dstu.edu.ru/informatics/mtdss/index.html>