МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ИСЛАМА КАРИМОВА

Информационные технологии в технических системах

Методические указания
к выполнению практических занятий

ТАШКЕНТ 2022

УДК 681.3

Методические указания к выполнению практических работ по курсу «Информационные технологии в технических системах. Составители: Д.Каримова, Ч.С.Хошимова, С.М.Хамдамова, Ташкент,ТашГТУ 2022. – 79с.

В методических указаниях приведены описания практических работ. Представлены информация для изучения и использования современных компьютерных платформ, технология создания, представления и обработки электронных документов, используемых в различных технических системах, методика создания информационных ресурсов, работа по обработке графических и видеофайлов с использованием режима мультимедиа. Особое внимание уделено использованию систем автоматизации проектирования при решении инженерных задач(CAD) и вопросам программирования с использованием современных систем ООП.

Описание каждой работы включает в себя краткую теоретическую часть, формулировку цели, упражнения и задания. После каждой работы приведены вопросы для самоконтроля.

Методические указания предназначены для студентов-бакалавров энергетического факультета для проведения практических занятий по курсу «Информационные технологии в технических системах».

Печатаются по решению научно-методического совета Ташкентского государственного технического университета.

Протокол №10 от 29 июня 2022г.

Рецензенты:

к.ф.м.н.доц. Маматкулова М.Ш. (НУУз);
к.т.н., доц. М.М. Кадыров (ТашГТУ)

Введение

Современные тенденции все более широкого использования информационных технологий и средств компьютерного моделирования во всех отраслях экономики ставят новые требования по уровню подготовленности студентов высших технических заведений в области информационных коммуникационных технологий и программирования.

Предлагаемые методические указания ставят своей целью, наряду с лекционными и практическими видами занятий углубить знания по предмету «Информационные технологии в технических системах» и получить основные навыки и умения по базовым разделам второго семестра учебного курса.

Методические указания содержат описание самих работ, необходимый краткий теоретический материал для их выполнения, примеры выполнения и варианты исходных данных для практических работ. Кроме этого, есть примеры выполнения этих работ.

В конце приведен список литературы, позволяющий самостоятельно изучить дополнительный материал по другим источникам.

После выполнения каждой работы студент должен предоставить в письменном виде отчет о проделанной работе, в который входят исходные данные, полученные результаты и выводы.

Практическое занятие №1

Разработка математических моделей инженерных задач в области энергетики с использованием практических программ (MathCAD).

Цель занятия: *Ознакомиться и получить навыки выполнения простейших вычислений в пакете MathCAD.*

Теоретическая часть

MathCAD является интегрированной системой, ориентированной на проведение математических и инженерно-технических расчётов. Он объединяет понятность, ясность, простоту в обращении при вычислениях и т.п. с простотой в обращении, свойственной электронным таблицам. После запуска приложения MathCAD открывается окно, как показано на рис. 1.

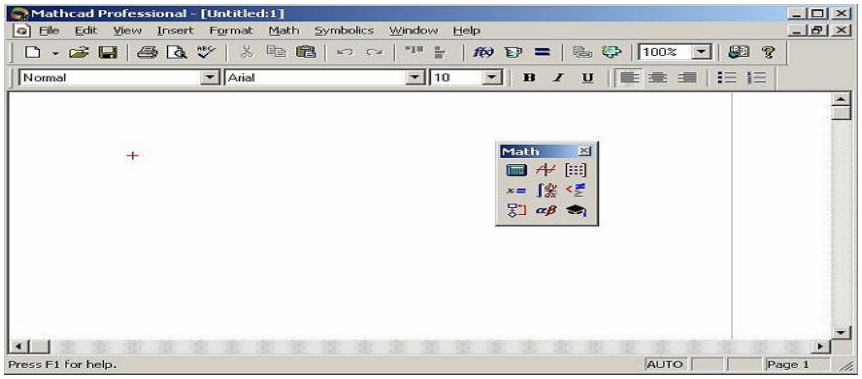


Рис. 1. Рабочее окно системы MathCAD

Основные команды MathCAD

Главное меню системы MathCAD представлено набором команд, общим для большинства приложений операционной системы MS Windows, а также командами, представляющими специфические возможности.

Меню **File** – работа с файлами.

Меню **Edit** – редактирование документов.

Меню **View** – настройка элементов окна. Команды меню View представлены на рис.1.

Меню **Insert** – позволяет помещать в MathCAD документ графики, функции, матрицы, гиперссылки, компоненты и настраивать объекты.

Меню **Format** – содержит команды, предназначенные для задания различных параметров, определяющих внешнее представление чисел, формул, текста, абзацев, колонтитулов и т.д.

Меню **Math** – позволяет установить режимы и параметры вычислений.

Меню **Symbolic** – реализует символьные вычисления.

Меню**Window** – содержит команды для упорядочения взаимного расположения нескольких окон и позволяет активизировать одно их них.

Меню **Help** – информационный центр и справочники. Команда Help открывает окно, представленное на рис. 1.

Кнопки панели Math

Одна из сильных сторон MathCAD – это представление и ввод математических символов и выражений в привычной для человека форме. Открыть соответствующую панель инструментов можно с помощью команды главного меню View →Toolbars. Для удобства работы ссылки на них объединены на панели Math.

На панели Math расположены 9 кнопок. Каждая из кнопок, в свою очередь, открывает панели инструментов специального назначения. Это следующие кнопки (в развёрнутом виде эти панели представлены на рис.2).

Calculator. На этой панели находятся кнопки для задания математических операций, а также некоторых часто используемых функций. Эту кнопку можно использовать как калькулятор.

Boolean – для ввода операторов сравнения и логических операций.

Evaluation – содержит кнопки для ввода операторов присвоения значений переменных и функций.

Graph – инструменты для построения графика.

Vector and Matrix – инструменты для работы с векторами и матрицами.

Calculus – представляет математические выражения с элементами интегрирования, дифференцирования в привычном виде. Кнопки это панели позволяют вычислять значения пределов, сумм, произведений.

Programming – инструменты для написания программ.

Greek Symbol – графический алфавит.

Symbol – Для символьных вычислений.

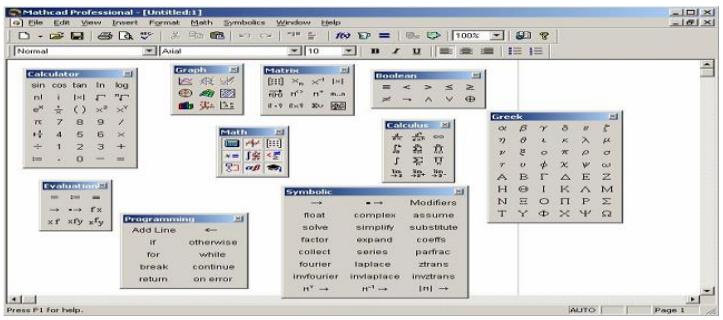


Рис.2. Рабочее окно системы MathCad с развёрнутыми панелями инструментов панели Math.

Математические выражения

К основным элементам математических выражений MathCAD относятся типы данных, операторы, функции и управляющие структуры.

Операторы – элементы MathCAD, с помощью которых можно создавать математические выражения. К ним, например, относятся символы арифметических операций, знаки вычисления сумм, произведений, производной и интеграла и т.д.

Запись команд в рабочем документе системы MathCAD

Запись команд в системе MathCAD на языке очень близка к стандартному языку математических расчётов выполнимых на бумаге, что значительно упрощает постановку и решение задач. В результате главные аспекты решения математических задач смещаются с их программирования на алгоритмическое и математическое описание.

MathCAD реализует вычисления в строго определённом порядке, как это делает человек: читая страницу книги, т.е. слева направо и сверху вниз. Правильный порядок выполнения блоков – основа правильного функционирования системы при обработке документа.

К типам данных относятся числовые константы, обычные и системные переменные, массивы (векторы и матрицы) и данные файлового типа.

Используемые типы констант

В системе MathCAD предусмотрены следующие типы данных:

1. Целые (2, -54,+43).
2. Вещественные (1.3,-2.23).
3. Комплексные (2.5+7i).

Следует иметь в виду, что при записи мнимой единицы следует использовать специальную кнопку панели Calculus.

4. Строковые. Обычно это комментарии вида: «Вычисление суммы».
5. Системные. Системная константа – это предварительно определённая переменная, значение которой задаётся в начале загрузки системы. Примерами таких констант являются числа e или π.

Простые вычисления

Результат арифметического выражения отображается, если после него стоит знак «=» или знак «→». В первом случае результат представляется в численном виде, а во втором – в символьном.

Пример символьного вычисления:

$$\frac{2.45}{6.178} + \frac{4}{52} - 76 - \frac{8}{87} \rightarrow -75.618462477305312281$$

При выполнении вычислительной системой арифметического выражения используются знаки арифметических операций с приоритетами, принятыми в обычной математике. Выражение может содержать также другие операции:

- извлечение корня;
- возведение в степень;
- интегрирование и дифференцирование;
- знаков факториала и суммирования и т.д.

Часть этих операций можно «взять» на панели Calculator. Примером записи выражения может быть:

$$4.5 \cdot \left(\sqrt[5]{56.3} + \sqrt{14.356}\right) + 5.2^{1.8} - 4.89 + \frac{6.52}{4.78} = 43.046$$

Количество значащих цифр, отображаемых при вычислении, можно регулировать с помощью главного меню Format→Result. В этом случае

команда предоставит диалоговое окно, как это показано на рис.3, в котором следует переустановить параметры для вывода результата.

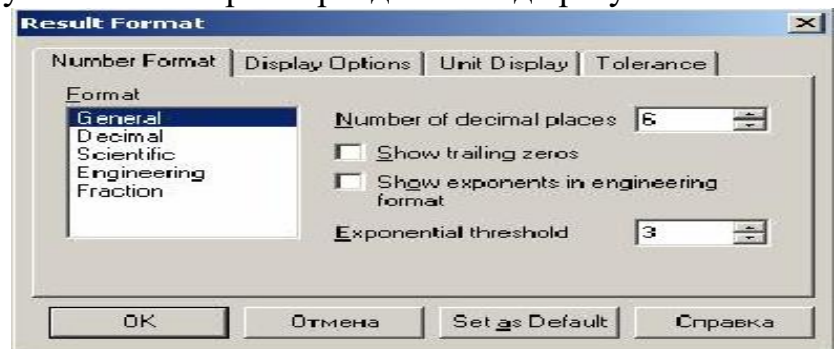


Рис. 3. Рабочее окно команды главного меню Format (формат Result)

Ниже приведён результат символьного вычисления арифметического выражения:

$$\frac{25}{47} - 3^{-2} + \frac{7}{3}\cdot 2.5 + \pi \rightarrow 6.2541371158392434988 + \pi \text{ float},4 \rightarrow 9.396$$

После знака «→» отображён результат символьного вычисления. Для замены результата символьного вычисления численным значением применена команда float, расположенная на панели Symbolic. Эта команда представляет шаблон, в котором пользователю предлагается задать количество знаков (цифр) для отображения результата.

Графики в MathCAD являются и универсальными, и легкими в использовании.

Графические области делятся на три основных типа – двумерные графики, трехмерные графики и импортированные графические образы. Двумерные и трехмерные графики строятся самим MathCAD на основании обработанных данных.

Для создания декартового графика:

Установить визир в пустом месте рабочего документа.

Выбрать команду Insert⇒ Graph ⇒ X-Y Plot, или нажать комбинацию клавиш Shift+@, или щелкнуть кнопку  панели Graph. Появится шаблон декартового графика.

Введите в средней метке под осью X первую независимую переменную, через запятую – вторую и так до 10, например x1, x2, …

Введите в средней метке слева от вертикальной оси Y первую независимую переменную, через запятую – вторую и т. д., например, y1(x1), y2(x2), …, или соответствующие выражения.

Щелкните за пределами области графика, чтобы начать его построение.

Результаты построения двухмерных графиков представлены на рис.4.

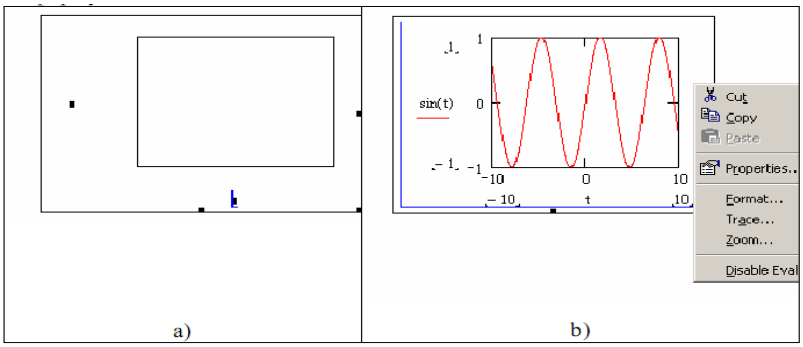


Рис.4. Графическая область в декартовой системе координат

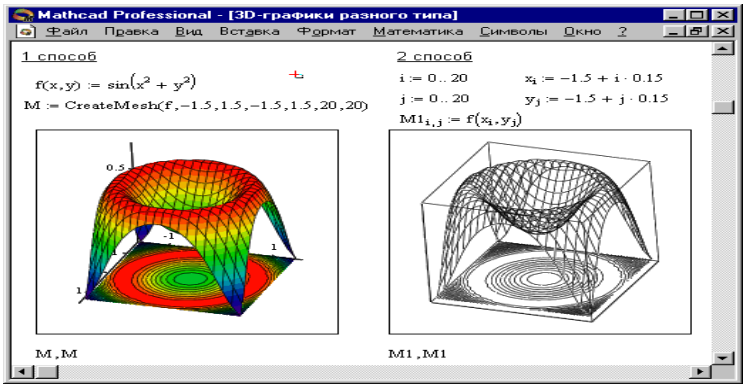


Рис.5. Пример построения на одном рисунке двух 3D-графиков разного типа

Такой график создается операцией Insert⇒ Graph ⇒ 3D Scatter Plot, причем поверхность задается параметрически – с помощью трех матриц (X, Y, Z) (см. рис. 5, способ 2), а не одной как в примере на рис. 6. Для определения исходных данных для такого вида графиков используется функция CreateSpace (см. рис. 6, способ 1).

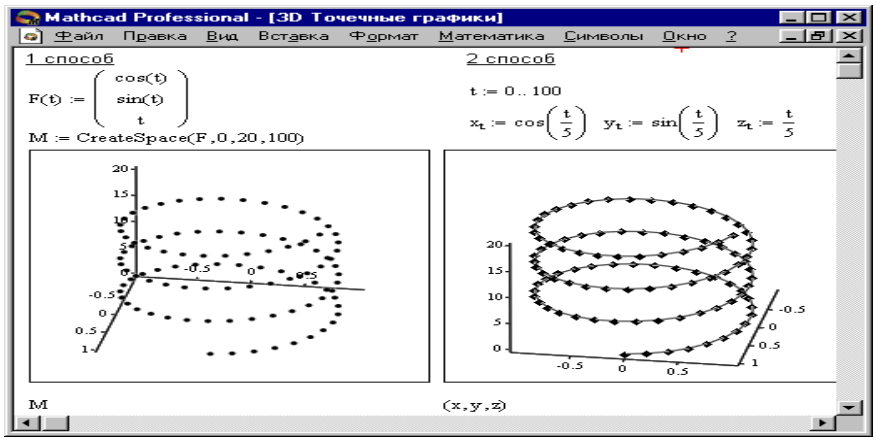


Рис.6. Построение 3D точечных графиков

CreateSpace (F, t0, t1, tgrid, fmap) – возвращает вложенный массив трех векторов, представляющих x-, y-, и z-координаты пространственной кривой, определенной функцией F. t0 и t1 – диапазон изменения переменной, tgrid – размер сетки переменной, fmap – функция отображения.

Упражнение 1. Вычислить выражение: $10\sin x - x^2$

Ввести данные в рабочее окно как показано на рис.7 и получить результат:

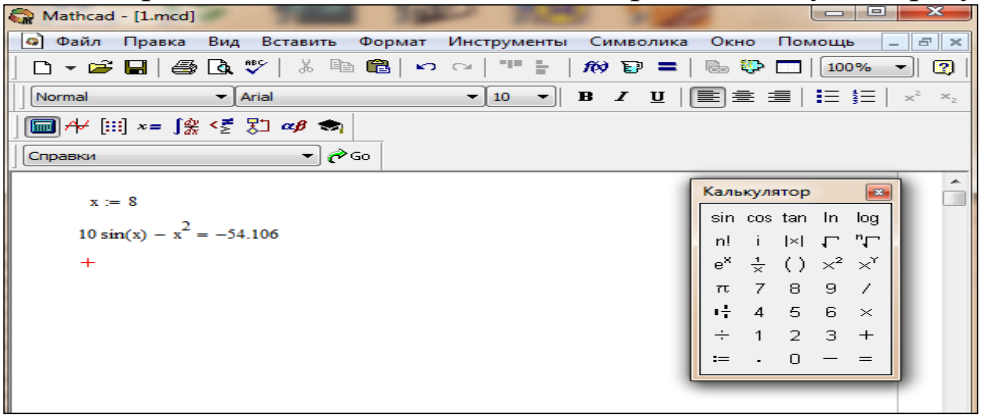


Рис.7. Результат

Упражнение 2.Работа с матрицами (решение уравнений матричным способом)

A				B
13.47	-2.29	3.29	4.75	2.32
2.75	11.11	2.28	-0.75	4.75
0.28	6.25	-9.21	0.79	2.25
3.21	2.21	0.49	7.87	-3.41

В палитре инструментов выберите палитру матриц и введите значение матрицы. В рабочей области введите формулы, как показано на рис.8.

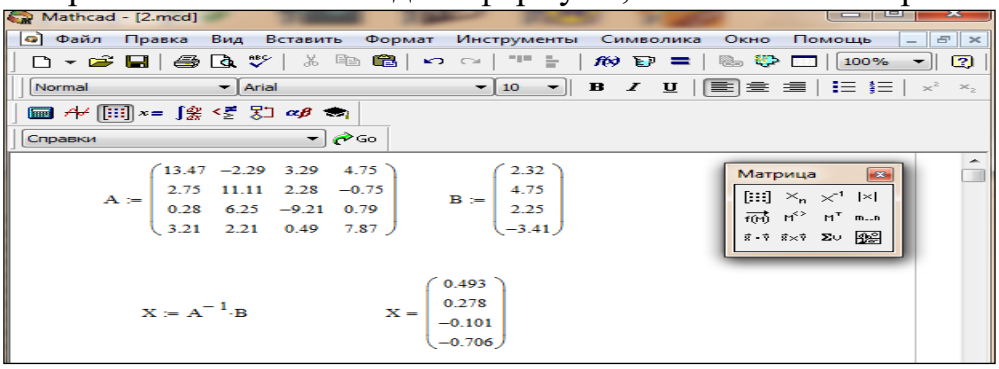


Рис.8.

Упражнение 3. Вычислить: $y = cosx + xsinx + (cosx)xcosx$.

Создать график функции (рис.9).

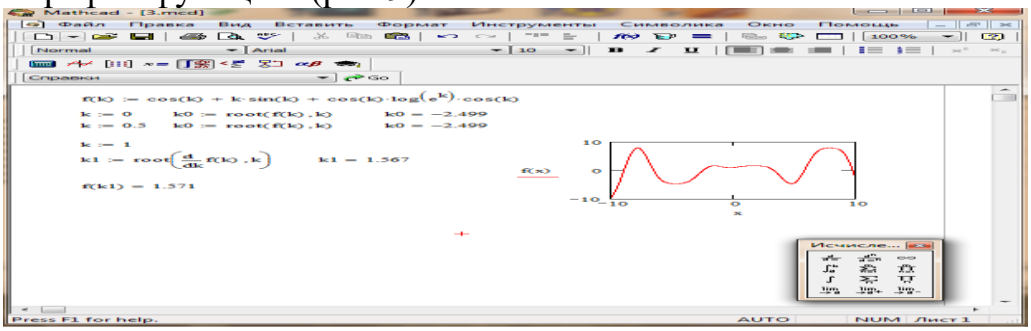


Рис.9.

Упражнение 4. Построение графиков. Для создания *декартового графика*:

1.Установить визир в пустом месте рабочего документа.

2.Выбрать команду **Insert⇒ Graph ⇒ X-Y Plot**, или нажать комбинацию клавиш **Shift+@,** или щелкнуть кнопку  панели **Graph**. Появится шаблон декартового графика.

3.Введите в средней метке под осью *X* первую независимую переменную, через запятую – вторую и так до 10, например, *x*1, *x*2, …

4.Введите в средней метке слева от вертикальной оси *Y* первую независимую переменную, через запятую – вторую и т. д., например, *y*1(*x*1), *y*2(*x*2), …, или соответствующие выражения.

5.Щелкните за пределами области графика, чтобы начать его построение.

Результаты построения двухмерных графиков представлены на рис.10,11.

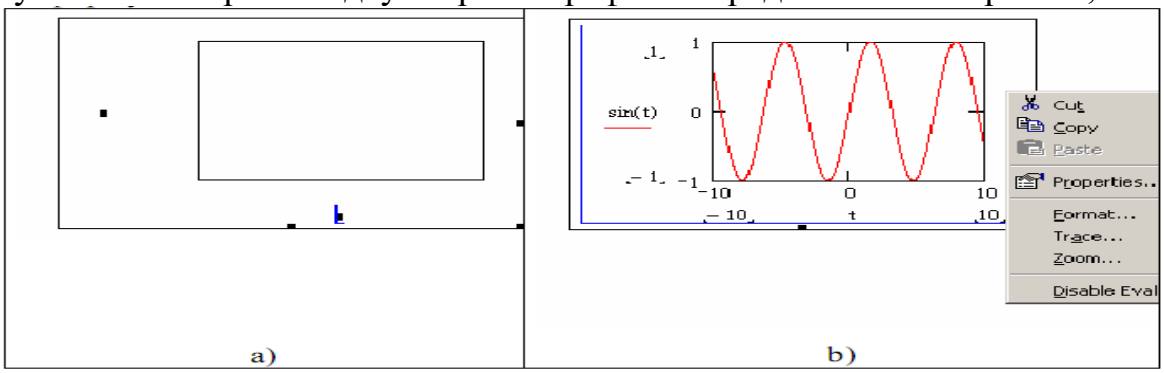


Рис.10.

График в MathCAD можно построить ***по точкам.*** В этом случае задаются два столбца значений *x* и *y* и уже по ним на плоскости строят точки, соответствующие этим столбцам. Столбцы задаются нажатием на кнопку с изображением матрицы на панели **Matrix**. Чтобы получить сам график нужно нажать на кнопку с изображением осей на панели **Graph**. В появившейся рамочке графика будут 2 незаполненных черных прямоугольника – маркера. В один маркер, отвечающий за ординату, нужно поместить название матрицы-столбца, который должен быть отложен по оси ОУ. В другой (нижний) маркер помещают название другого столбца. Далее следует нажать **enter**.

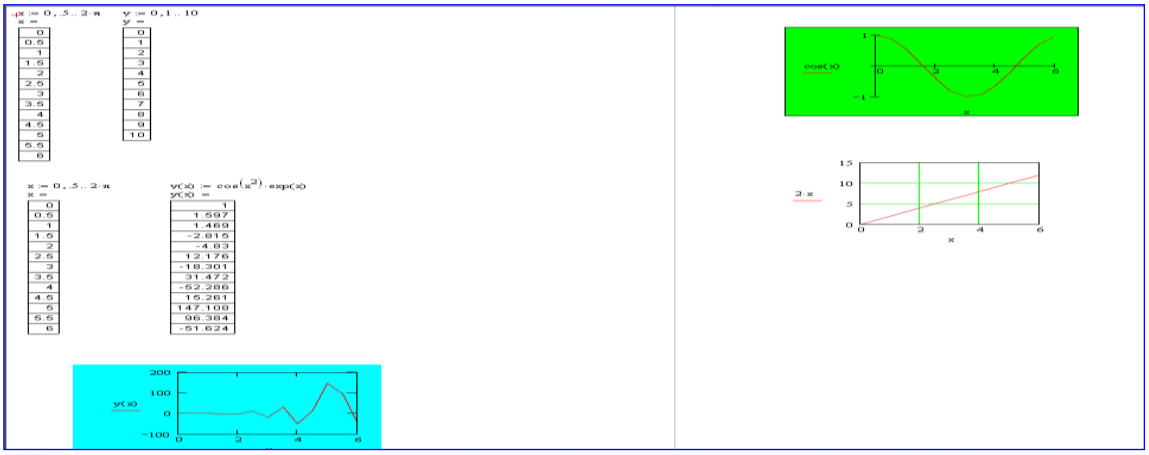


Рис.11

Упражнение 5. Построение трехмерных графиков.

Трехмерные, или ***3D-графики***, отображают функции двух переменных вида *Z*(*X*, *Y*). При построении трехмерных графиков в ранних версиях MathCAD поверхность нужно определить математически (рис. 12, способ 2). Далее применяют функцию MathCAD *CreateMesh.*

CreateMesh(*F* (или *G*, или *f*1, *f*2, *f*3), *x*0, *x*1, *y*0, *y*1, *xgrid*, *ygrid*, *fmap*) – создает сетку на поверхности, определенной функцией *F*. *x*0, *x*1, *y*0, *y*1 – диапазон изменения переменных, *xgrid*, *ygrid* – размеры сетки переменных, *fmap* – функция отображения. Все параметры, за исключением *F*, – факультативные. Функция *CreateMesh* по умолчанию создает сетку на поверхности с диапазоном изменения переменных от -5 до 5 и с сеткой 20×20 точек. Пример использования функции *CreateMesh* для построения 3D-графиков приведен Способ 1. На рис.12. построена одна и та же поверхность разными способами, с разным форматированием, причем изображены поверхности и под ними те же поверхности в виде контурного графика. Такое построение способно придать рисунку большую наглядность.

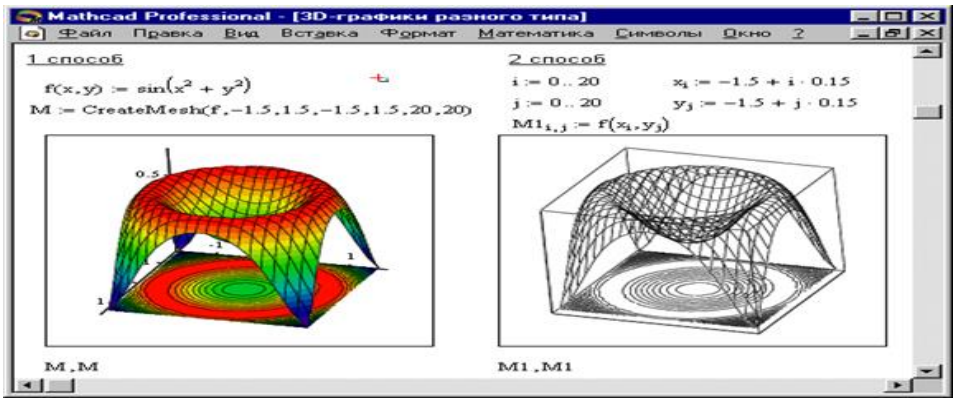


Рис. 12. Пример построения на одном рисунке двух 3D-графиков разного типа

Такой график создается операцией **Insert⇒ Graph ⇒ 3D Scatter Plot**, причем поверхность задается параметрически – с помощью трех матриц (*X*, *Y*, *Z*) (см. рис.8, способ 2), а не одной как в примере на рис.9. Для определения исходных данных для такого вида графиков используется функция *CreateSpace* (см. рис. 6, способ 1).

Нередко поверхности и пространственные кривые представляют в виде точек, кружочков или иных фигур.

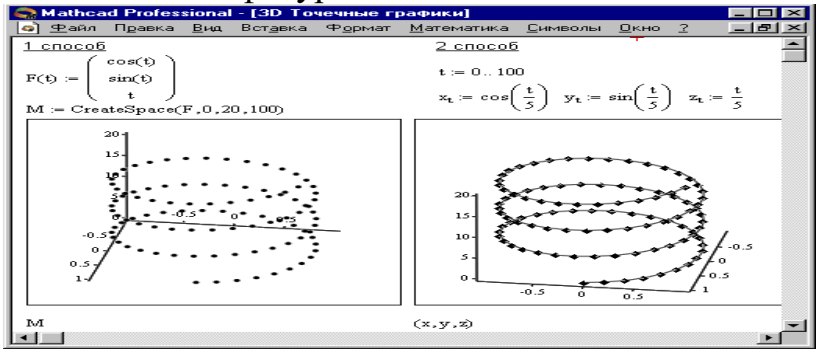


Рис. 13. Построение 3D точечных графиков

CreateSpace (*F*, *t*0, *t*1, *tgrid*, *fmap*) – возвращает вложенный массив трех векторов, представляющих *x*, *y*, z - координаты пространственной кривой, определенной функцией *F*. *t*0 и *t*1 – диапазон изменения переменной, *tgrid* – размер сетки переменной, *fmap* – функция отображения.

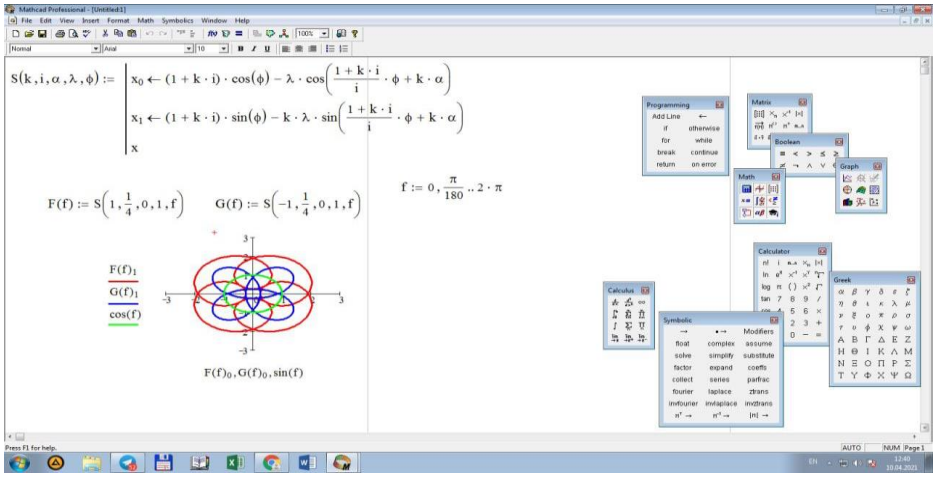


Рис 14.

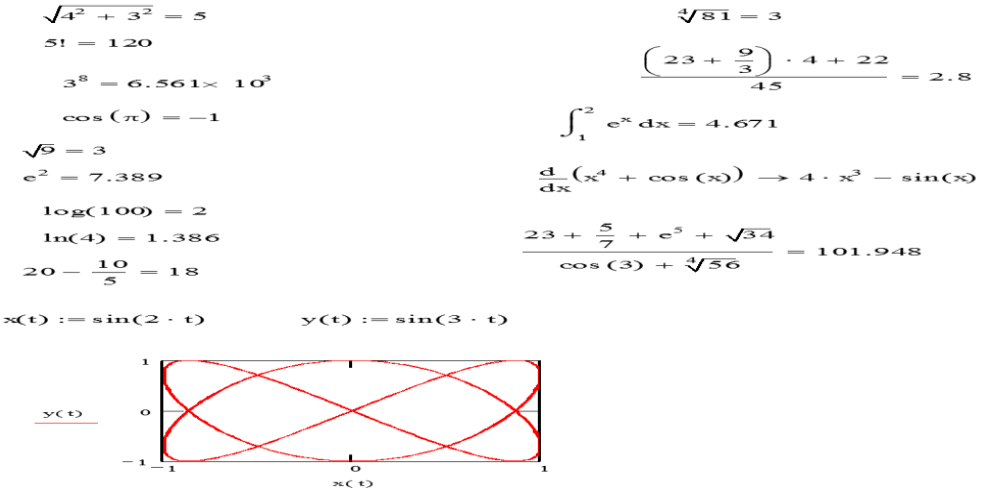


Рис 15.

Контрольные вопросы:

1.Какие вычисления выполняются в системе MathCad?

2.Как в системе MathCad реализуются графики функций?

3.Какие действия с матрицами предусмотрены в MathCad?

4.Как в системе MathCad решаются уравнения?

Практическое занятие №2

Ознакомление с основными этапами моделирования энергетических задач в системе MATLAB.

Цель работы*:* Ознакомиться с программой **MATLAB**, изучить её основные возможности при решении задач моделирования.

Порядок выполнения работы:

1. Изучить теоретическую часть.
2. Ознакомиться с режимами работы системы.
3. Выбрать задачу по специальности для решения.
4. Произвести необходимые операции для решении данной задачи.

Теоретическая часть

Специализированная система Matlab. Модели Simulink.

При инсталляции Matlab можно установить пакет визуального моделирования систем, процессов — Simulink. Чтобы создать новую модель, щёлкните мышкой по пиктограмме  или наберите команду **simulink** . Перед вами появится два окна. В первом, можно из библиотеки выбрать модель, а во втором, как из кирпичиков, вы собираете общую модель системы:

Двойным щелчком, например, по пиктограмме Source (Источники), вы откроете библиотеку источников. Открывая соответствующие библиотеки и перетягиванием мышкой выбранной модели в окно создания моделей системы вы можете набрать модель реальной системы. Далее, запустив модель, вы получите решение. В настройках можно выбрать время работы системы, метод решения и некоторые другие параметры.

«Matlab» является продуктом фирмы «The MathwoksInc» Первая версия пакета «Matlab» была разработана уже более 20 лет тому назад. Развитие и совершенствование этого пакета происходило одновременно с развитием средств вычислительной техники. Название пакета «Matlab» происходит от словосочетания MatrixLaboratory, он ориентирован, в первую очередь, на обработку массивов данных (матриц и векторов). Именно поэтому, несмотря на достаточно высокую скорость смены поколений вычислительной техники, «Matlab» успевал впитывать все наиболее ценное от каждого из них.

Как следует из названия пакета, он ориентирован, в первую очередь, на обработку массивов данных (матриц и векторов). Это позволило его разработчикам существенно повысить эффективность процедур, работающих с указанными типами данных.

В результате к настоящему времени «Matlab» представляет собой богатейшую библиотеку функций, моделей (более 800). Единственная проблема при работе, с ними заключается в умении быстро отыскать те модели и функции из них, которые нужны для решения поставленной задачи.

Именно в сфере математического моделирования «Matlab» позволяет наиболее полно использовать все современные достижения компьютерных технологий, в том числе средства визуализации и аудификации (озвучивания) данных, а также возможности обмена данными через Интернет. Кроме того, пользователь имеет возможность создавать средствами «Matlab» собственный графический интерфейс, отвечающий как его вкусам, так и требованиям решаемой задачи.

Для облегчения работы с пакетом специалистами различных областей науки и техники вся библиотека функций разбита на разделы. Те модели и

функции из них, которые носят более общий характер, входят в состав основного ядра пакета «Matlab». Те же модели и функции, которые являются специфическими для конкретной области, включены в состав пакетов расширения (Toolboxes).

Таким образом, «MatLab» - в первую очередь, это средство математического моделирования, обеспечивающее проведение исследований с точки зрения анализа и синтеза, практически во всех известных областях техники и науки. При этом структура пакета позволяет эффективно сочетать оба основных подхода к созданию модели: аналитический и имитационный.

SIMULINK

Особое место среди наборов инструментов занимает система визуального моделирования Simulink. В определенном смысле Simulink можно рассматривать как самостоятельный продукт фирмы MathWorks (который даже в некоторых случаях продается в «именной» упаковке). Однако он работает только при наличии ядра «Matlab» и использует многие функции, входящие в его состав.

Следует обратить внимание, что пакеты «MatLab», Simulink и их пакеты расширения (Toolboxes, Blocksets) постоянно развиваются и совершенствуются.

Simulink – это уникальный инструмент исследования, не требует от пользователя знания навыков программирования, позволяет приобрести мощнейшие ассоциации и уяснить аналогии между структурой математического описания объекта и структурой блок схемы, реализующей данную модель.

Simulink – интерактивный инструмент для моделирования, имитации и анализа динамических систем. Он дает возможность строить графические блок - диаграммы, имитировать динамические системы, исследовать их работоспособность систем и совершенствовать проекты.

Simulink – полностью интегрирован с «MatLab», обеспечивая немедленный доступ к широкому спектру инструментов анализа и проектирования

Сразу становятся ясными причинно-следственные отношения с цепочкой математических понятий: независимая переменная, дифференциальное уравнение, вынуждающая функция, начальные условия, решение дифференциального уравнения и с соответствующей цепочкой понятий: «аналоговое» моделирование – время моделирования, входной сигнал, структурная схема объекта, начальные условия на интеграторах, выходной сигнал и т. д. При моделировании в среде «MatLab», Simulink - средств измерения, регулирования и систем управления у студентов, слушателей и исследователей прививаются:

-навыки выбора, наладки, эксплуатации более современных средств автоматизации;

- навыки использования современных технологий в процессе обучения;

- четкое определение уравнения статики и динамики изучаемого объекта;

- умение анализировать работу изучаемого объекта;

- умение синтезировать изучаемый объект с желаемой характеристикой.

Пакет «MatLab» располагает широким набором виртуальных элементов, модулей, функций, представленных в виде условных обозначений, которые обладают основными свойствами реальных простых физических элементов, а в совокупности простых устройств, модулей, агрегатов, преобразователей, приборов, регуляторов, систем контроля и регулирования и т.д.

Таким образом, «собрав» на экране монитора из соответствующих элементов требуемую виртуальную лабораторную работу, то есть определенную структурную схему изучаемого объекта, можно выполнить ее полный анализ, изучить его в установившихся и переходных режимах и произвести его синтез с желаемыми характеристиками. При этом можно быть уверенным, что при корректной сборке схемы, (корректном выборе типовых преобразователей, установке необходимых коэффициентов типовых звеньев - элементов, правильном их конфигурировании между собой, с учетом их принципиальной схемы) и умелом проведении экспериментов, результаты исследований совпадут с результатами исследований в реальной схеме, а по точности превзойдет их. В этом суть программных пакетов «MatLab», «Simulink» и их несомненные превосходства и достоинства.

Пакеты «MatLab», «Simulink» полезны, не только для студентов в процессе изучения различных предметов при всех формах обучения, но и инженерам, научным работникам, которые проектируют и испытывают новейшие устройства, ибо эти испытания можно провести на виртуальной модели. Это избавляет студентов и исследователей от необходимости строить физическую модель изучаемого устройства (выбирать необходимые устройства, их монтировать, заполнять необходимыми веществами, настраивать и эксплуатировать) и, следовательно, резко сокращает как материальные, так и временные затраты. Исследование параметров конкретной системы(схемы устройства) с помощью процедур системы MatLab.

Расчёт параметров и исследование функциональных зависимостей средствами MathCad. Используются 2 режима работы в системе MatLab. Режим командного программирования и режим моделирования с использованием пакетов приложений.

Выберем в пакете Simulink блок "Matlab function" и в параметрах пропишем путь к нашему файлу с уравнениями (файл называется Mathfuncs_Diode). Входным параметром будет напряжение, которое измеряется вольтметром, а выходным ток.

Всю систему поместим в один блок - Subsystem. Назовем наш блок с диодом DiodeModel. Создадим маску диода

Для тестирования модели диода соберем схему Блок XY Graph строит ВАХ диода, Multimeter - графики напряжения и тока на диоде и резисторе.

Рассмотрим реакцию системы с единичной ООС и апериодическим звеном в прямой цепи на единичное воздействие. Для её набора нам понадобятся библиотеки модели: Source|Step, Sinks|Scope, Linear|Transfer Fcn и Linear|Sum. Перетянув эти элементарные модели в окно редактирования Simulink, изменяем исходные параметры в соответствии с нашей системой (двойной щелчок мышью) и соединяем их.

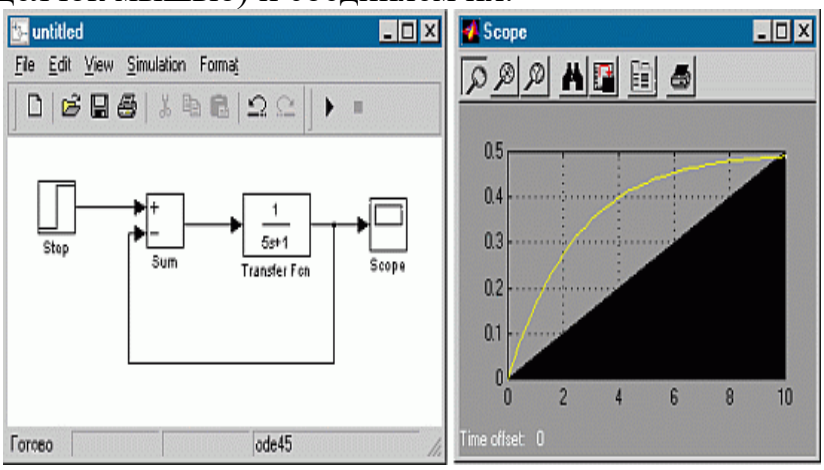


Рис.16.

Чтобы можно было использовать полученное решение в сценарии Matlab нужно добавить компонент Connections|Out. Теперь вы имеете возможность работать с переменной. Данные, которые поступают на этот порт доступны вам из среды Matlab. Чтобы запустить модель, вам достаточно набрать команду **sim** с необходимыми параметрами (наберите HELP SIM для подробной информации).

В последние годы в ВУЗах ряда стран при изучении студентами разных предметов в виде лекционных, лабораторных и практических занятий, при выполнении курсовых работ, проектов, выпускных работ и диссертаций все более широкое применение находит система моделирования и анализа с применением ряда программных комплексов. Среди программных комплексов пакет «Matlab», отличается своей простотой, компактностью, надежностью, расширенными возможностями, отвечающими современным требованиям, базирующимся на новых современных технологиях процессу обучения. Особое место среди наборов инструментов занимает система визуального моделирования Simulink. В определенном смысле Simulink можно рассматривать как самостоятельный продукт фирмы MathWorks (который даже в некоторых случаях продается в «именной» упаковке). Однако он работает только при наличии ядра «Matlab» и использует многие функции, входящие в его состав.

Следует обратить внимание, что пакеты «MatLab», Simulink и их пакеты расширения (Toolboxes, Blocksets) постоянно развиваются и совершенствуются.

Simulink – это уникальный инструмент исследования, не требует от пользователя знания навыков программирования, позволяет приобрести

мощнейшие ассоциации и уяснить аналогии между структурой математического описания объекта и структурой блок-схемы, реализующей данную модель.

Simulink – интерактивный инструмент для моделирования, имитации и анализа динамических систем. Он дает возможность строить графические блок - диаграммы, имитировать динамические системы, исследовать их работоспособность систем и совершенствовать проекты.

Simulink – полностью интегрирован с «MatLab», обеспечивая немедленный доступ к широкому спектру инструментов анализа и проектирования

Сразу становятся ясными причинно-следственные отношения с цепочкой математических понятий: независимая переменная, дифференциальное уравнение, вынуждающая функция, начальные условия, решение дифференциального уравнения и с соответствующей цепочкой понятий: «аналоговое» моделирование – время моделирования, входной сигнал, структурная схема объекта, начальные условия на интеграторах, выходной сигнал и т. д. При моделировании в среде «MatLab», Simulink - средств измерения, регулирования и систем управления у студентов, слушателей и исследователей прививаются:

-навыки выбора, наладки, эксплуатации более современных средств автоматизации;

- навыки использования современных технологий в процессе обучения;
- четкое определение уравнения статики и динамики изучаемого объекта;
- умение анализировать работу изучаемого объекта;
- умение синтезировать изучаемый объект с желаемой характеристикой.

Пакет «MatLab» располагает широким набором виртуальных элементов, модулей, функций, представленных в виде условных обозначений, которые обладают основными свойствами реальных простых физических элементов, а в совокупности простых устройств, модулей, агрегатов, преобразователей, приборов, регуляторов, систем контроля и регулирования и т.д.

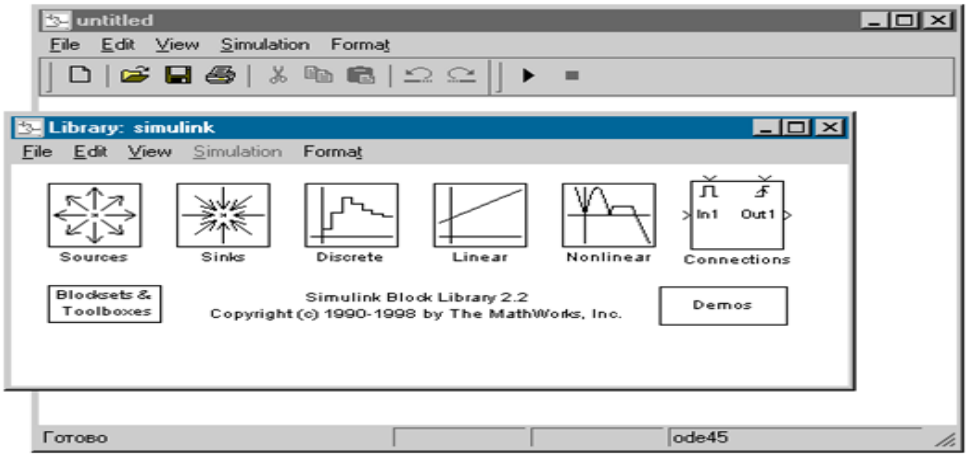


Рис 17.

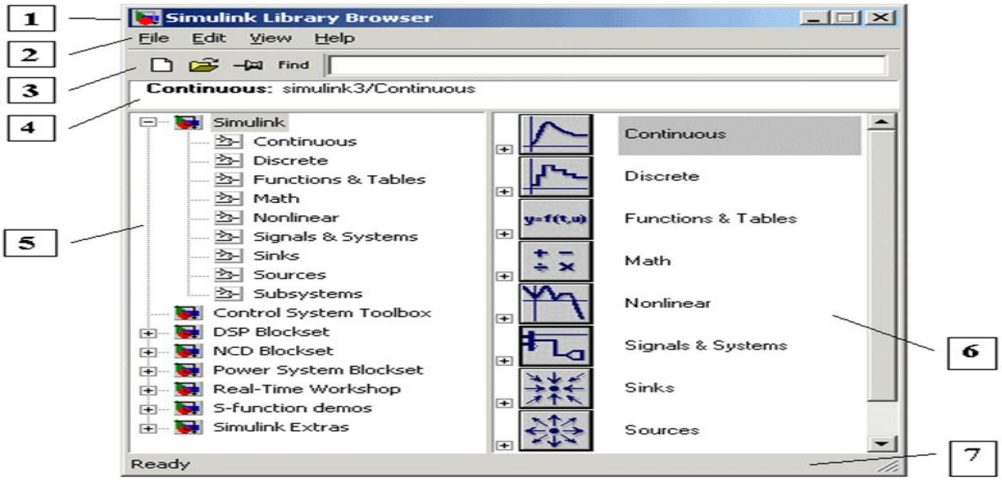


Рис 18

Таким образом, «собрав» на экране монитора из соответствующих элементов требуемую виртуальную лабораторную работу, то - есть определенную структурную схему изучаемого объекта, можно выполнить ее полный анализ, изучить его в установившихся и переходных режимах и произвести его синтез с желаемыми характеристиками. При этом можно быть уверенным, что при корректной сборке схемы, (корректном выборе типовых преобразователей, установке необходимых коэффициентов типовых звеньев - элементов, правильном их конфигурировании между собой, с учетом их принципиальной схемы) и умелом проведении экспериментов, результаты исследований совпадут с результатами исследований в реальной схеме, а по точности превзойдет их. В этом суть программных пакетов «MatLab», «Simulink» и их несомненные превосходства и достоинства.

Пакеты «MatLab», «Simulink» полезны, не только для студентов в процессе изучения различных предметов при всех формах обучения, но и инженерам, научным работникам, которые проектируют и испытывают новейшие устройства, ибо эти испытания можно провести на виртуальной модели. Это избавляет студентов и исследователей от необходимости строить физическую модель изучаемого устройства (выбирать необходимые устройства, их монтировать, заполнять необходимыми веществами, настраивать и эксплуатировать) и, следовательно, резко сокращает как материальные, так и временные затраты.

Математические функции

Логарифмическая функция	**log(x)**
Экспоненциальная функция	**exp(x)**
Квадратный корень	**sqrt(x)**
Степенная функция	**a^x**
Синус	**sin(x)**
Косинус	**cos(x)**
Десятичный логарифм	**log10(x)**
Абсолютное значение	**abs(x)**

Практическая часть

1 Задание.

а) Вычисление функций.

```
>> exp([1 2 3])

ans =

   2.7183    7.3891   20.0855

>> log([2 5 8])
ans =
   0.6931    1.6094    2.0794
```

2) Работа с матрицами.

```
>>A=[1 2 0;2 5 -1;4 10 -1]          построчно
A=
      1   2    0
      2   5   -1
      4  10  -1
>>B=A.'
B=
      1   2    4
      2   5   10                    транспонированная матрица
      0  -1   -1
>>C=A*B
C=
     5     12    24
    12     30    59                  умножение матриц
     24    59   117
>>C=A.*B
C=
     1     4    0          /.*  умножение соответствующих элементов
     4    25   -10                  матриц    A и  B/
     0    -10   1
>>X=inv(A)
X=
      5   2   -2
     -2   -1   1
      0    -2   1
```

Обратная матрица

```
>> I=inv(A)*A
I=
1   0   0
0   1   0
0   0   1
```
единичная матрица

```
>> eig(A)                                  функция определения собственных чисел
eigen(A)
ans=
        3.7321
        0.2679
        1.0000
S=svd(A)                                    сингулярные значения
ans=
     12.3171
      0.5149
      0.157

>> B=[1 2 3];
>> sum(B.')          вычисление суммы элементов матрицы

>>magic(3)
  ans =
            6
```
3. Построение графиков.

а)В декартовых координатах

```
>> x=-5: 0.05: 5;
>>y=sin(x.^2);
>>plot(x,y);
```
б) графики в виде гистограмм

```
>>x= -2.9:0.2:2.9;
>>bar(x,exp(x.*x)) ;
```
в) графики в полярных координатах

```
>>t=0:0.01:2*pi;
>>polar(t,abs(sin(2*t).*cos(2*t)));
```
г) трехмерные графики

```
>>t=0:pi/10:2^pi;
>>[X,Y,Z]=cylinder(4*cos(t));
>>subplot(2,2,1);
```

```
>>mesh(X)
```

Упражнение 1. (2-3 графика в одном в MATLAB, набираем):

```
>>f=0:pi/180:2*pi;
>>k1=0;
>>k2=-1;
>>i=0.25;
>>a=0;
>>l=1
>>y1=(1+k1*i)*cos(f)-l*cos((((1+k1*i)/i)*f)+k1*a);
>>y2=(1+k1*i)*sin(f)-k1*l*sin((((1+k1*i)/i)*f)+k1*a);
>>y3=(1+k2*i)*cos(f)-l*cos((((1+k2*i)/i)*f)+k2*a);
>>y4=(1+k2*i)*cos(f)-k2*l*cos((((1+k2*i)/i)*f)+k2*a);
>>plot(y1,y2,y3,y4,sin(f),cos(f))
```

Упражнение 2.

```
1)>>t=0:pi/10:2^pi;
>>[x,y,z]=cylinder(4*cos(t));
>>subplot(2,2,1);
>>mesh(x)
2)>>plot(y)  получится 2-х мерный график
3) >>bar(y)
>>xlabel('sample#')
>>ylabel('pounds')    получится    гистограмма
4)>>plot(y,'*')
>>axis([010010])    точечный  график
5)>>x=0:10;
>>a=tan(x);
>>plot(x,a)
```

Графики

```
1)>>x=0:0.05:5;
>>y=sin(x.^2);
>>plot(x,y) интервал функции график со сжатием
2)>>x=-2.9:0.2:2.9;
>>bar(x,exp(x.*x));  график в виде гистограмма
3)>>t=0:.01:2*pi;
>>polar(t,abs(sin(2*t).*cos(2*t))); полярный график (ромашка)
```

Работа с матрицами

```
>>A=[1 2 0;2 5 -1;4 10 -1]   построчно
A=
      1 2  0
      2 5 -1
      4 10 -1
>>B=A'
B=
     1 2 4
```

```
      2 5 10
      0 -1 -1    транспонированная матрица
>>C=A*B   Умножение матриц
C=
5 12 24
12 30 59
24 59 117
>>C=A.*B /.*оператор умножения соответствующих  элементов матриц A и
B /
C=
       1   4     0
       4   25  -10
       0   -10   1
>>exp([1 2 3])
ans=
      2.7183  7.3891  20.0855
>>log([2 5 8])
ans=
       0.6931  1.6094  2.0794
>>x=inv(A)  инверсная (обратная матрица)
X=
        5  2   -2
       -2  -1  1
        0  -2   1
I=inv(A)*A
I=
     1  0  0
     0  1  0
     0   0  1

>>eig(A)   функция определения eigen values(собственные числа)
ans=
        3.7321
        0.2679
        1.0000
>>svd(A)  функция сингулярных значений(единственных)
ans=
           12.3171
           0.5149
           0.1577
```

Исследование параметров конкретной системы(схемы устройства) с помощью процедур системы MatLab. Расчёт параметров и исследование функциональных зависимостей средствами системы. Используются 2 режима работы в системе MatLab. Режим командного программирования и режим моделирования с использованием пакетов приложений.

Выберем в пакете Simulink блок "Matlab function" и в параметрах пропишем путь к нашему файлу с уравнениями (файл называется Mathfuncs_Diode). Входным параметром будет напряжение, которое измеряется вольтметром, а выходным- ток.
Всю систему поместим в один блок - Subsystem. Назовем наш блок с диодом DiodeModel. Создадим маску диода
Для тестирования модели диода соберем схему Блок XY Graph строит ВАХ диода, Multimeter - графики напряжения и тока на диоде и резисторе.
Рассмотрим реакцию системы с единичной ООС и апериодическим звеном в прямой цепи на единичное воздействие. Для её набора нам понадобятся библиотеки модели: Source|Step, Sinks|Scope, Linear|Transfer Fcn и Linear|Sum. Перетянув эти элементарные модели в окно редактирования Simulink, изменяем исходные параметры в соответствии с нашей системой (двойной щелчок мышью) и соединяем их.
Чтобы запустить модель, вам достаточно набрать команду **sim** с необходимыми параметрами (наберите HELP SIM для подробной информации).

1) Выполнить следующую последовательность

a. Launch Pad ->Simulink->Library Browser
b. File->New->Model
c. В новой модели разместите следующие блоки:

- Sources->Signal Generator
- Sinks->Scope
- Math->Abs
- Sinks->Display

д. Соедините элементы в следующей последовательности и получите результат:

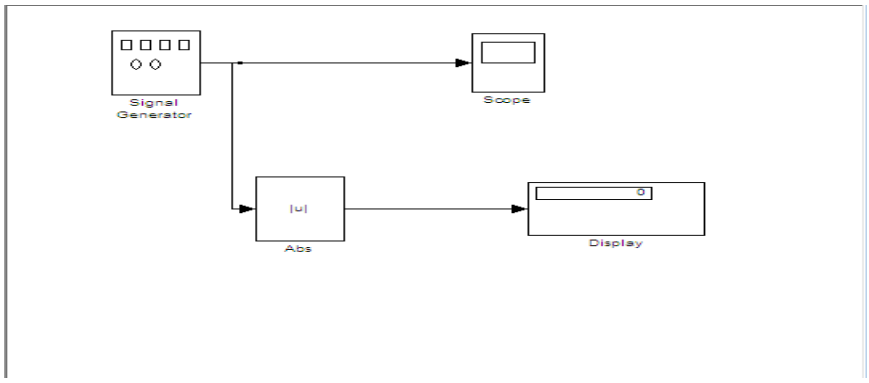


Рис.19

2) a. Launch Pad ->Simulink->Library Browser
b. File->New->Model
c. В новой модели разместите следующие блоки:

- Sources->Signal Generator
- Sinks->Scope

- Sinks->Display

д. Соедините элементы в следующей последовательности и получите результат:

Контрольные вопросы.

1. Из каких библиотек состоит пакет Simulink?
2. Как собрать модель в пакете Simulink?
3. Как изменить параметры моделирования?
4. Какие существуют способы визуализации процесса моделирования?
5. Перечислите возможности системы MatLab.

Практическая работа № 3
Разработка имитационных моделей энергетических задач при прикладных программах. (SIMULINK).

Цель работы: *Ознакомиться с возможностями пакета Simulink системы Matlab при решении задач моделирования систем управления технических систем*

Теоретическая часть

Моделирование электрических и электронных схем в настоящее время невозможно представить без использования вычислительной техники. Существует и разрабатывается множество программных средств для этой цели, такие как MultiSim, MicroCap, SIMetrix, CircuitMaker, ASIMEC, в том числе и программа Simulink программного комплекса Matlab. Использование программы Simulink по сравнению с большинством других программ позволяет моделировать как физические воздействия, так и сами схемные решения с использованием мощных средств моделирования самих приборов. Однако библиотека электрических и электронных компонентов практически не содержит полупроводниковых приборов, таких как диоды и биполярные транзисторы.

В настоящее время существуют математические модели, достаточно точно описывающие работу полупроводниковых диодов. Использование программы Simulink по сравнению с большинством других программ позволяет моделировать как физические воздействия, так и сами схемные решения с использованием мощных средств моделирования самих приборов. В настоящее время существуют математические модели, достаточно точно описывающие работу полупроводниковых диодов Однако библиотека электрических и электронных компонентов практически не содержит полупроводниковых приборов, таких как диоды и биполярные транзисторы. Программа Simulink является приложением к пакету MATLAB. При моделировании с использованием Simulink реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет

расчеты. При работе с Simulink пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков. После открытия основного окна программы MATLAB нужно запустить программу Simulink. Это можно сделать одним из трех способов:

1. Нажать кнопку (Simulink) на панели инструментов командного окна MATLAB.
2. В командной строке главного окна MATLAB напечатать Simulink и нажать клавишу Enter на клавиатуре.
3. Выполнить команду **Open**… в меню **File** и открыть файл модели (mdl - файл).

Последний вариант удобно использовать для запуска уже готовой и отлаженной модели, когда требуется лишь провести расчеты и не нужно добавлять новые блоки в модель.
Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки Simulink (рис.20).

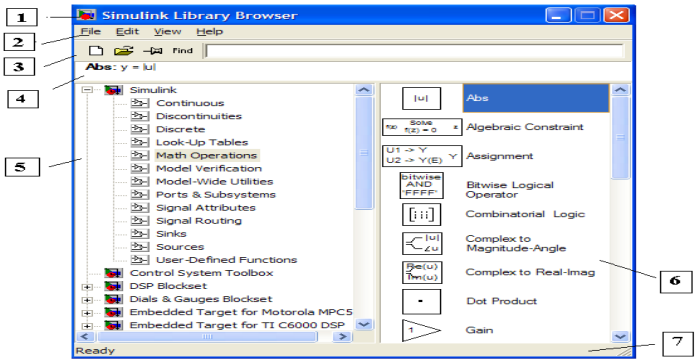


Рис.20 – Окно обозревателя разделов библиотеки Simulink

Окно обозревателя библиотеки блоков содержит следующие элементы:

1. Заголовок, с названием окна – Simulink Library Browser.
2. Меню, с командами **File**, **Edit**, **View**, **Help**.
3. Панель инструментов, с ярлыками наиболее часто используемых команд.
4. Окно комментария для вывода поясняющего сообщения о выбранном блоке.
5. Список разделов библиотеки, реализованный в виде дерева.
6. Окно содержимого раздела библиотеки (список вложенных разделов библиотеки или блоков)
7. Строка состояния, содержащая подсказку по выполняемому действию.

На рисунке 20 выделена основная библиотека Simulink (в левой части окна) и показаны ее разделы (в правой части окна).
Библиотека Simulink содержит следующие основные разделы:

1. **Continuous** – линейные блоки.
2. **Discrete** – дискретные блоки.
3. **User–Defined Functions** – функции и таблицы.
4. **Math Operations** – блоки математических операций.
5. **Discontinuities** – нелинейные блоки.
6. **Signals Attribute**, **Signals Routing** – сигналы и системы.
7. **Sinks** - регистрирующие устройства.
8. **Sources** — источники сигналов и воздействий.
9. **Ports & Subsystems** – порты и блоки подсистем.

Список разделов библиотеки Simulink представлен в виде дерева, и правила работы с ним являются общими для списков такого вида: Для создания модели в среде Simulink необходимо последовательно выполнить ряд действий:

1. Создать новый файл модели с помощью команды **File/New/Model**, Вновь созданное окно модели показано на рисунке 21.

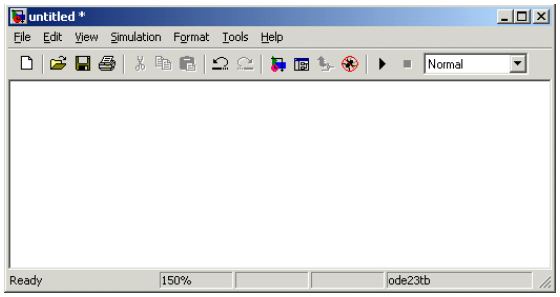


Рис.21 – Пустое окно модели

2. Расположить блоки в окне модели. Для этого необходимо открыть соответствующий раздел библиотеки (Например, **Sources** - Источники). Далее, указав курсором на требуемый блок и нажав на левую клавишу “мыши” - “перетащить” блок в созданное окно. **Клавишу мыши нужно держать нажатой**. На рисунке .22 показано окно модели, содержащее блоки.

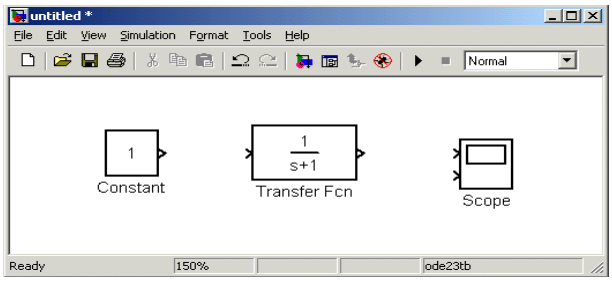


Рис.22 – Окно модели, содержащее блоки

Для изменения параметров блока необходимо дважды щелкнуть левой клавишей “мыши”, указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. При задании численных

параметров следует иметь в виду, что в качестве десятичного разделителя должна использоваться точка, а не запятая. После внесения изменений нужно закрыть окно кнопкой **OK**. После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы. Для соединения блоков необходимо указать курсором на “выход” блока, а затем, нажать и, не отпуская левую клавишу “мыши”, провести линию к входу другого блока. После чего отпустить клавишу. В случае правильного соединения изображение стрелки на входе блока изменяет цвет. Для создания точки разветвления в соединительной линии нужно подвести курсор к предполагаемому узлу и, нажав правую клавишу “мыши”, протянуть линию. Схема модели, в которой выполнены соединения между блоками, показана на рисунке 23.

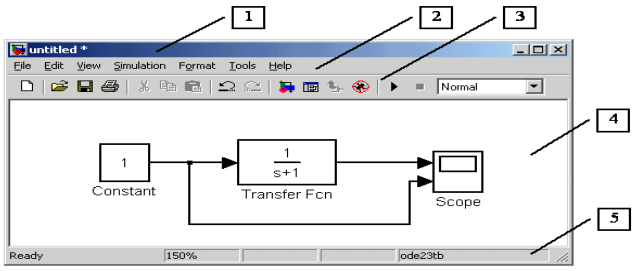


Рис. 23 – Схема модели

Окно модели содержит следующие элементы (см.рис.23):

1. Заголовок, с названием окна. Вновь созданному окну присваивается имя Untitled с соответствующим номером.
2. Меню с командами **File**, **Edit**, **View** и т.д.
3. Панель инструментов.
4. Окно для создания схемы модели.
5. Строка состояния, содержащая информацию о текущем состоянии модели.

Меню окна содержит команды для редактирования модели, ее настройки и управления процессом расчета, работы файлами и т.п.:

1. **File** (Файл) — Работа с файлами моделей.
2. **Edit** (Редактирование) — Изменение модели и поиск блоков.
3. **View** (Вид) — Управление показом элементов интерфейса.
4. **Simulation** (Моделирование) — Задание настроек для моделирования и управление процессом расчета.
5. **Forma**t (Форматирование) — Изменение внешнего вида блоков и модели в целом.
6. **Tools** (Инструментальные средства) — Применение специальных средств для работы с моделью (отладчик, линейный анализ и т.п.)
7. **Help** (Справка) — Вывод окон справочной системы.

Для повышения наглядности модели удобно использовать текстовые надписи. Для создания надписи нужно указать мышью место надписи и

дважды щелкнуть левой клавишей мыши. После этого появится прямоугольная рамка с курсором ввода. Аналогичным образом можно изменить и подписи к блоками моделей. Следует иметь в виду, что рассматриваемая версия программы (Simulink) не адаптирована к использованию кириллических шрифтов, и применение их может иметь самые разные последствия: отображение надписей в нечитаемом виде, обрезание надписей, сообщения об ошибках, а также невозможность открыть модель после ее сохранения. Поэтому применение надписей на русском языке для текущей версии Simulink крайне не желательно. Перед выполнением расчетов необходимо предварительно задать параметры расчета. Задание параметров расчета выполняется в панели управления меню **Simulation/Parameters**. Вид панели управления приведен на рис 24.

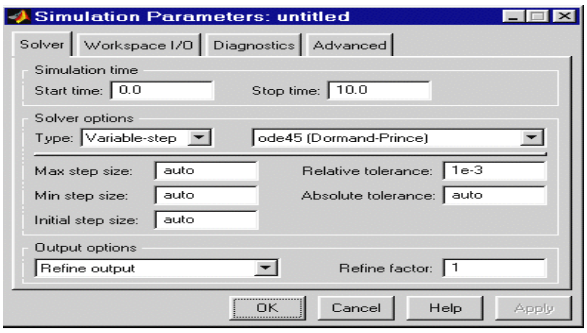


Рис 24. Панель управления

Упр 1. Собрать заданные схемы моделирования с использованием библиотеки.

1.Запустить Matlab и Simulink.

2. Настроить имя директории для хранения создаваемых файлов (команда: Home→Set Path→Add Folder1), указав свою папку на диске :

3. Создать новую модель с помощью верхнего меню окна Simulink Library Browser (команда: File→New→Model).

4.Добавить в окно модели блок Transfer Fcn (передаточная функция) из группы Continuous библиотеки Simulink, предназначенной для получения функциональных блок-схем различных устройств. Поместить блок в окне в соответствии со структурной схемой моделируемой системы.

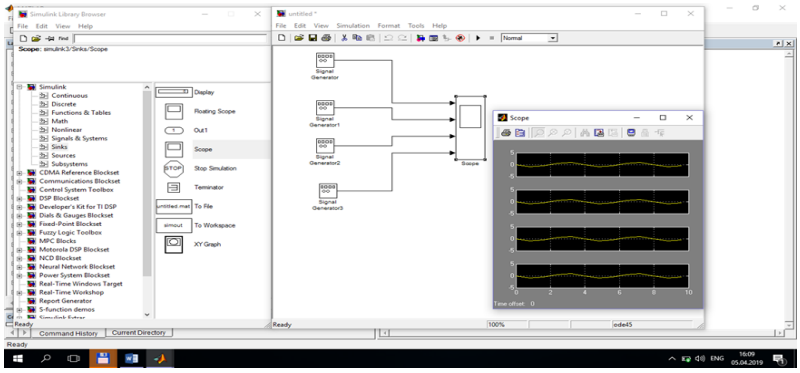


Рис 25.

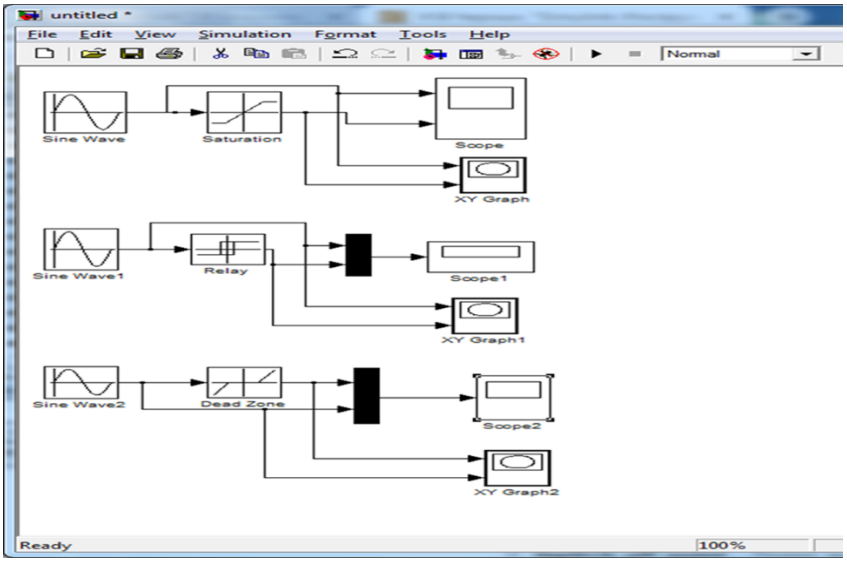


Рис. 26

Полученные результаты:

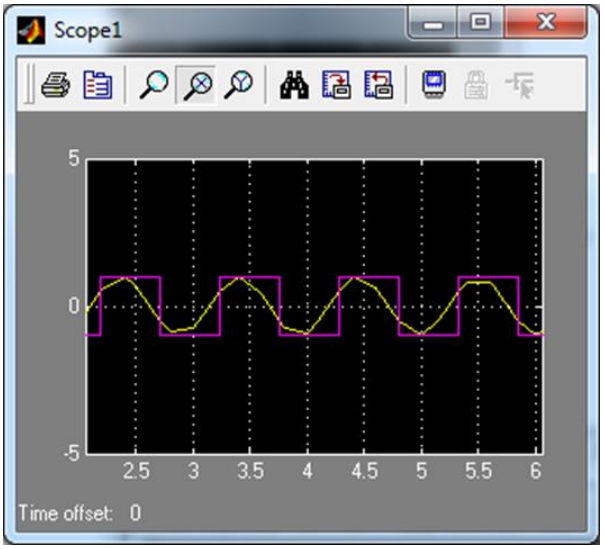


Рис. 27

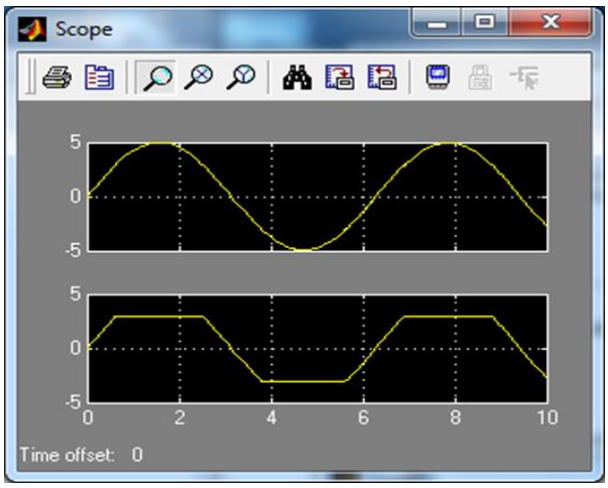


Рис. 28

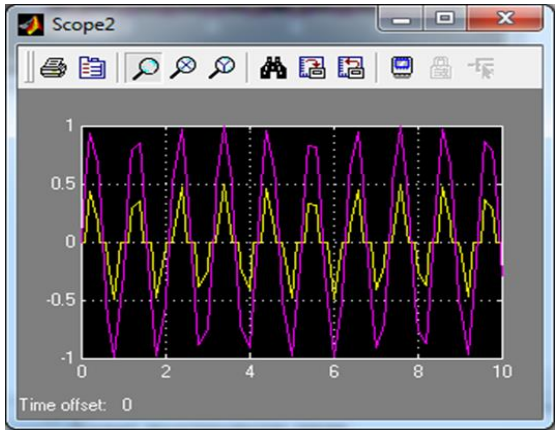


Рис.29

Упражнение 2. Построить схемы в системе Simulink по вариантам:

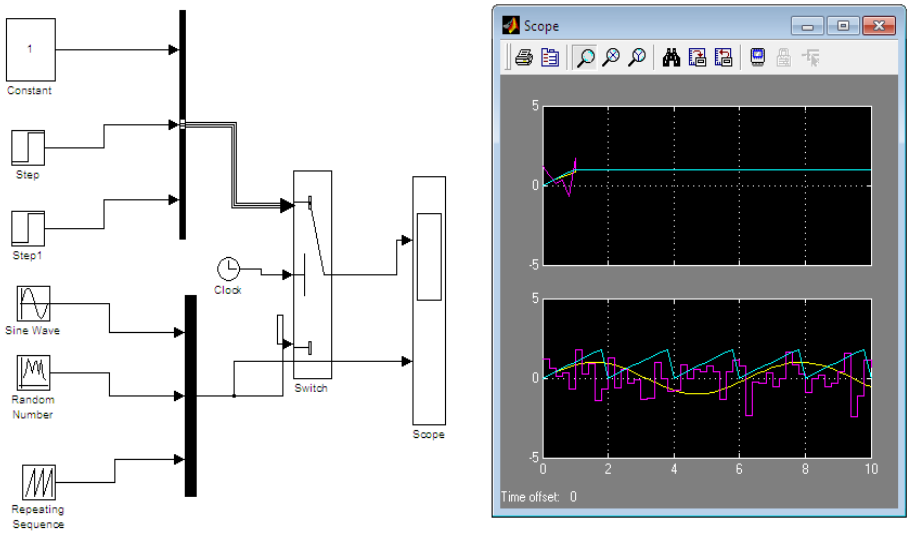


Рис.30

Упражнение 3.

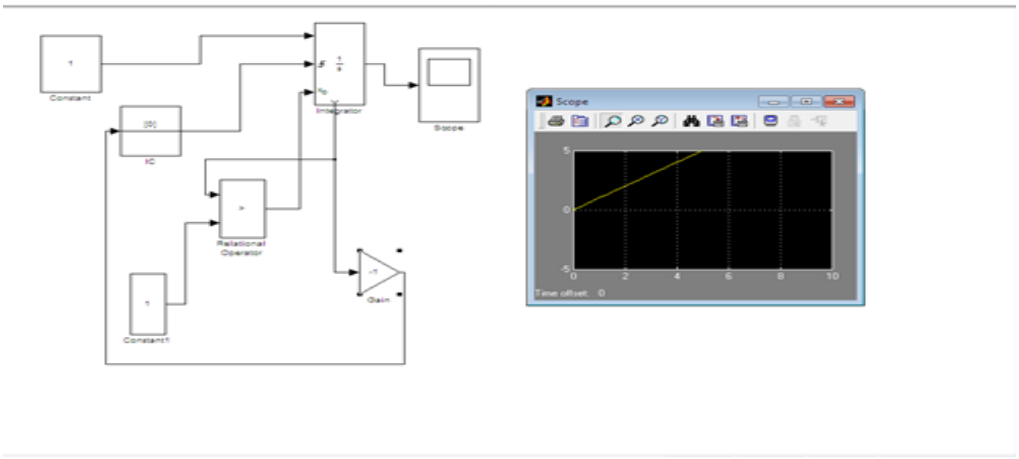


Рис. 31

Упражнение 4.

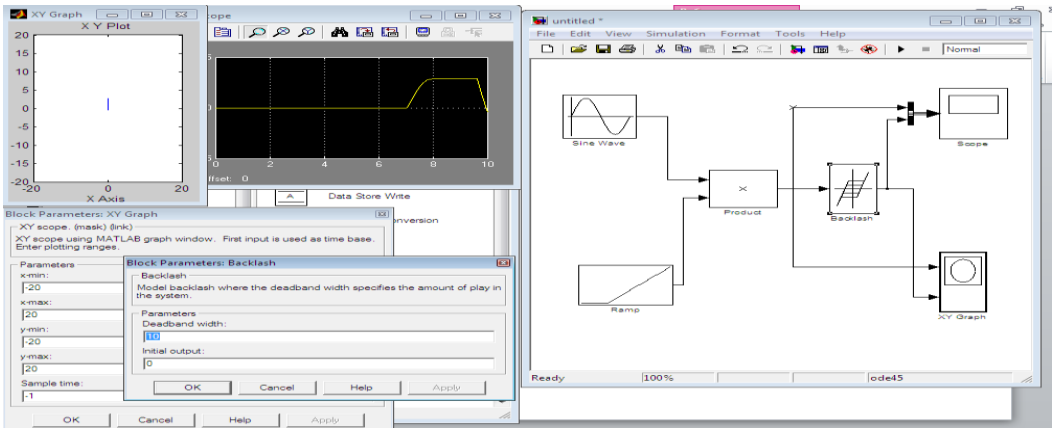


Рис.32

Упражнение 5

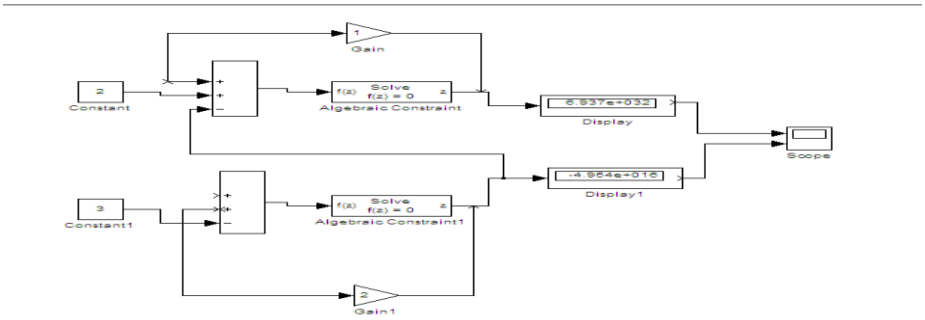


Рис.33

Контрольные вопросы

1.Какие средства автоматизации научной работы вы знаете?
2.Для каких процессов предназначена система Simulink?
3.Каким образом осуществляются простые расчеты в системе MATLAB?
4.С помощью какого инструмента строятся графики функций?
5.Перечислите возможности системы MATLAB.
6.Система визуального моделирования Simulink.

Практическое занятие № 4
Использование и визуализация графических возможностей приложений в процессе проектирования COMPAS 3D.

Цель работы: Применение навыков при работе с графическим редактором **COMPAS 3D**

Рабочее пространство КОМПАС 3D

Панель управления

Для вызова команд **КОМПАС** Вы можете обращаться не только к командам меню, но и к панелям, на которых расположены кнопки с изображенными на них пиктограммами. Каждая кнопка соответствует какой-либо команде системы.

При работе с **КОМПАС** на экране отображаются несколько различных панелей кнопок. Например, такие как ***Панели управления,*** ее состав различен для разных режимов работы и набор кнопок может быть изменен пользователем.

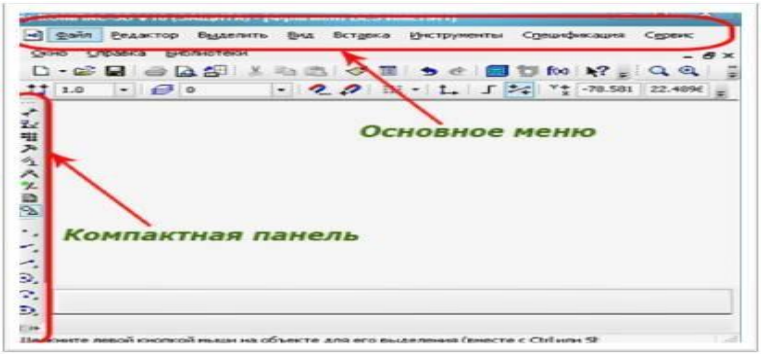


Рис.1

Панели переключения.

Помимо панели управления отображается ***Панель переключения.*** Её еще называют ***Компактная панель***.На ней находятся кнопки для переключения между страницами ***Инструментальной Панели.***

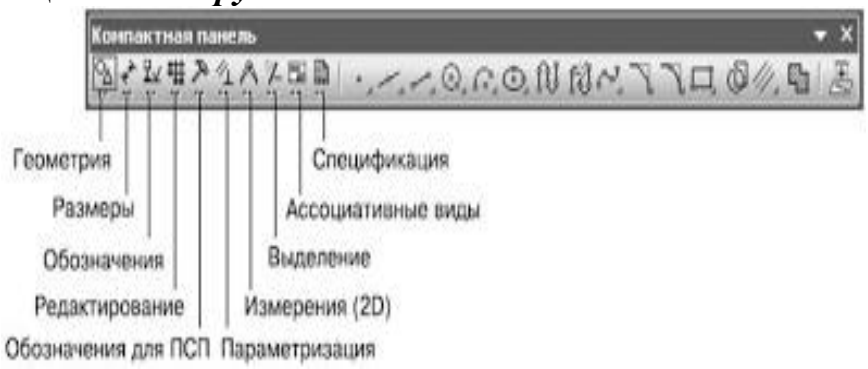


Рис.2

Инструментальная панель (она находится в левой части окна системы) к включает в себя: ***панель геометрии***, ***размеров***, ***редактирования***, ***измерений*** и ***панель селектирования*** **(выделения)**. Одновременно на экране отображается только одна страница панели.

Панели Специального управления

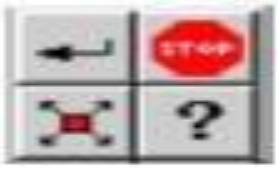


Рис.3

Панели **Специального управления** (она также появляется на экране только после вызова какой-либо команды) находятся кнопки, позволяющие контролировать процесс выполнения команды (ввод объекта, прерывание текущего действия и т.д.).

Для запуска команды с использованием соответствующей кнопки подведите курсор к этой кнопке и щелкните на ней левой кнопкой мыши.

Рис .4

Геометрические построения

В использовании команды ввода геометрических объектов поможет ***Панель геометрии.*** Чтобы вызвать расширенную панель команд, нужно нажать маленький черный треугольник в правом нижнем углу. Кнопки сгруппированы по типам объектов, ввод которых они вызывают , но на ***панели геометрии*** видна только одна кнопка из группы. Для того чтобы увидеть остальные кнопки группы и выбрать одну из них, нужно нажать на видимую кнопку группы и не отпускать клавишу мыши. *Прервать ввод объектов можно, нажав клавишу* <***Esc***> *или кнопку* ***Прервать команду*** *на* ***Панели специального управления***

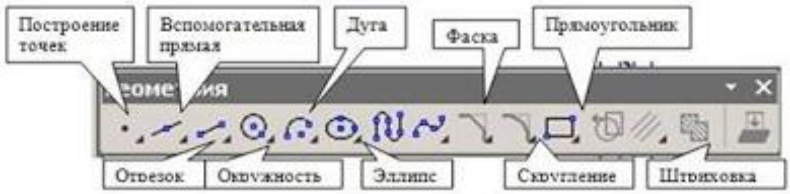


Рис.5

Команды и кнопки Инструментальной панели геометрии

Точка

Вычерчивание одной или нескольких точек.

При создании точек Вы можете явно указывать их положение, перемещая курсор по экрану мышью или клавишами. Можно также вводить значения координат точки в полях ***Строки параметров объектов*** и изменять стиль ее отрисовки.

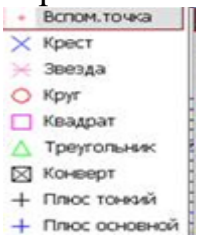


Рис.6

Команды и кнопки Инструментальной панели геометрии

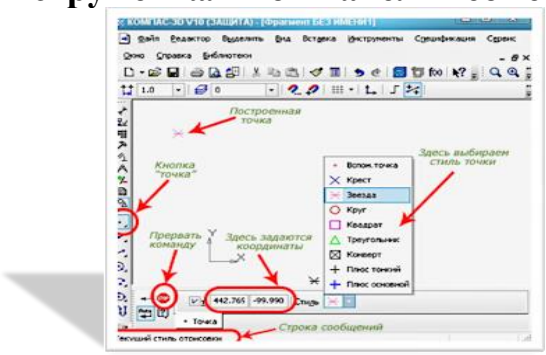
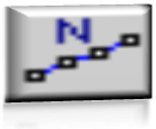


Рис.7

Команды и кнопки Инструментальной панели геометрии



Точки равномерно по кривой.

Для построения нескольких точек, разбивающих кривую на равные участки, предназначена команда "Точки по кривой", чтобы её вызвать нажмите кнопку "Точка" и удерживайте некоторое время. На панели свойств вводится количество участков, на которые требуется разбить кривую, мышью указывается сама кривая. Если кривая не замкнута, точки строятся сразу. Первая совпадает с начальной точкой кривой, последняя с конечной.

Для замкнутых кривых необходимо конкретно указывать положение первой точки.

Команды и кнопки Инструментальной панели геометрии

Точки пересечений

Команда "Точки пересечения двух кривых" служит для построения точек в местах пересечения кривых. Указывается кривая для поиска пересечений, а затем указываются пересекающиеся с ней кривые. Для выбора другой кривой для поиска пересечений нужно нажать кнопку "Указать заново" на панели свойств.

Непрерывный ввод

Для указания всех мест пересечений кривой с другими кривыми служит команда "Все точки пересечений кривой". Указывается кривая для поиска пересечений, после этого автоматически создаются точки в местах её пересечения с другими кривыми.

Команды и кнопки Инструментальной панели геометрии

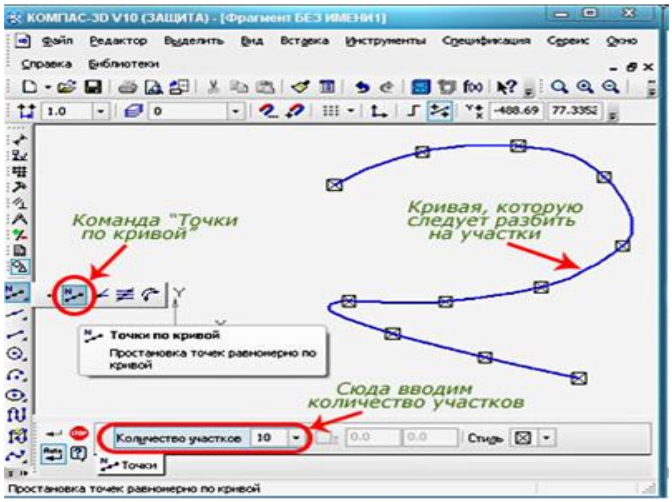


Рис.8

Команды и кнопки Инструментальной панели геометрии

Кнопка ***Отрезок***

Кнопка ***Дуга по 3 точкам***

Кнопка ***Кривая Безье***

Кнопка ***Перпендикулярный отрезок***

Кнопка ***Касательный к кривой отрезок из внешней точки***

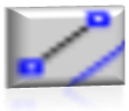
По умолчанию строится последовательность отрезков с концами в указываемых точках. Если нажать одну из кнопок в ***Строке параметров объектов*** для переключения на другой тип элемента, из развернувшейся панели можно выбрать другой вариант построения выбранного объекта.

Команды и кнопки Инструментальной панели геометрии

Кнопка ***Отрезок***

Отрезок – самый простой и наиболее используемый вариант построения отрезка. Создание возможно путем указания на чертеже двух

точек (начальной и конечной) или задания начальной точки, угла наклона и длины отрезка.



Кнопка ***Паралелльный отрезок***
Параллельный отрезок – после вызова команды вы должны указать любой прямолинейный объект, после чего зафиксировать первую точку отрезка. Далее вы можете перемещать указатель в любую сторону, но фантомное изображение отрезка будет строиться строго параллельно выбранному объекту. Зафиксировав вторую точку, вы получите отрезок, параллельный указанному прямолинейному объекту.
Команды и кнопки Инструментальной панели геометрии

Перпендикулярный отрезок – действие команды аналогично команде Параллельный отрезок, только отрезок строится перпендикулярно указанному объекту

Касательный отрезок через внешнюю точку – для построения отрезка нужно задать любой криволинейный объект и точку, не лежащую на этом объекте. Первой точкой созданного объекта будет внешняя точка, а второй – точка касания воображаемой прямой и указанного объекта.
Команды и кнопки Инструментальной панели геометрии

Вспомогательная прямая
Создание вспомогательной прямой, проходящей через две указанные точки. При создании прямых можно явно указывать положение характерных точек, перемещая курсор по экрану мышью или клавишами. Можно также вводить значения координат точек и другие параметры в полях ***Строки параметров объектов***.

Панель "Размеры"

Рис.9

КОМПАС 3D поддерживает все предусмотренные ЕСКД типы размеров: линейные, диаметральные, угловые и радиальные .

Инструментальная панель размеров и технологических обозначений. В КОМПАС команды простановки размеров и технологических обозначений доступны только через кнопки **Инструментальной панели**, для них не предусмотрен вызов через страницы Главного меню.

Кнопки команд простановки размеров и технологических обозначений расположены на **Инструментальной панели** размеров. Здесь же находятся команда ввода текста и команда простановки обозначения центра. Кнопки сгруппированы по типам размеров и обозначений, ввод которых они вызывают .

ТЕКСТ

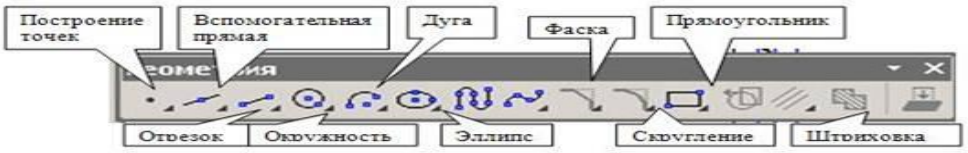


Рис.10

Введение одной или нескольких текстовых надписей.

Для ввода текста укажите курсором положение точки его привязки. Затем напечатайте в открывшейся рамке ввода нужное количество строк, заканчивая набор каждой из них нажатием клавиши Enter. Для перехода к созданию новой надписи просто переместите курсор за пределы рамки ввода и нажмите левую кнопку мыши. Предыдущая надпись будет зафиксирована, и откроется новое поле ввода текста в указанном Вами месте.

При необходимости Вы можете изменять внешний вид вводимой надписи или отдельных ее частей (тип шрифта, его высоту и сужение, цвет символов и т.д.) с помощью полей и кнопок в Строке параметров объектов.

Линейный размер

Простановка одного или нескольких линейных размеров.

Последовательно укажите две базовые точки, а затем положение размерной линии.

Значения координат характерных точек размера можно явно ввести в полях Строки параметров объектов.

Вы можете также автоматически привязать создаваемый размер к граничным точкам какого-либо геометрического объекта, нажав кнопку Выбор объекта на Панели специального управления, а затем указав курсором нужный элемент.

По умолчанию будет создаваться размер, параллельный базовому объекту или отрезку, соединяющему базовые точки.

Линейный размер с обрывом

Простановка одного или нескольких линейных размеров с обрывом.

Укажите курсором отрезок, который будет базовым для размера с обрывом. Выбранный объект будет подсвечен. Затем перемещайте курсор для достижения нужного положения размерной линии и надписи.

Построение линейного размера с обрывом имеет следующие особенности:

1. Если размерная надпись расположена на полке, то полка будет начинаться от середины размерной линии.

2. Если полки нет, то размерная линия автоматически строится такой длины, чтобы разместился текст размерной надписи.

3. Текст размерной надписи вводится только вручную.

Отказаться от ввода размеров с обрывом можно, нажав клавишу **Esc** и ли кнопку **Прервать команду** на **Панели специального управления**.

Угловой размер с обрывом.

Простановка одного или нескольких угловых размеров с обрывом.

Укажите курсором объект (отрезок или прямую), который будет базовым для углового размера с обрывом, а затем ось симметрии. Перемещайте курсор для достижения нужного положения размерной линии и надписи.

Следует заметить, что при построении углового размера с обрывом действуют некоторые допущения:

1. Если размерная надпись расположена на полке, то полка будет начинаться от середины размерной дуги.

2. Величина угла измеряется автоматически.

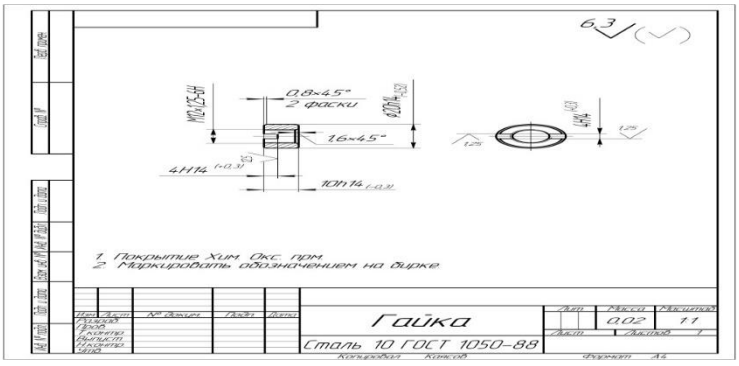


Рис.11 Редактирование готового объекта.

Практическая часть

Первым делом запускаем программу нажатием значка на рабочем столе компьютера или через меню **«ПУСК» - «Все программы»**, как Вам удобнее. На экране появится примерно следующее (в зависимости от версии продукта, в данном примере используется Компас-3D V10):

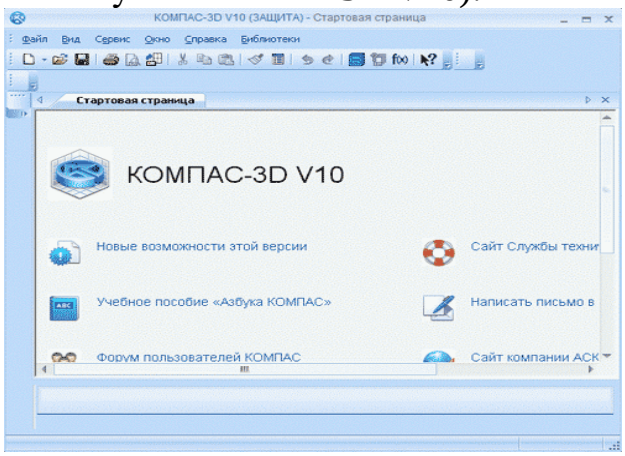


Рис.12

Далее заходим в верхнее меню навигации, нажимаем **«Файл»-«Создать»**

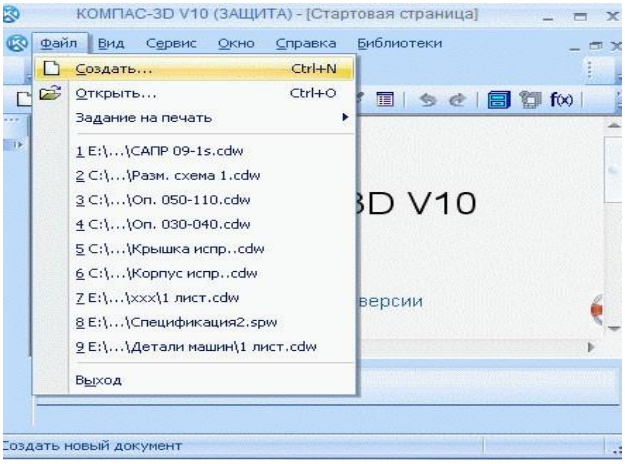


Рис.13

Или нажимаем соответствующий значок на верхней панели инструментов.

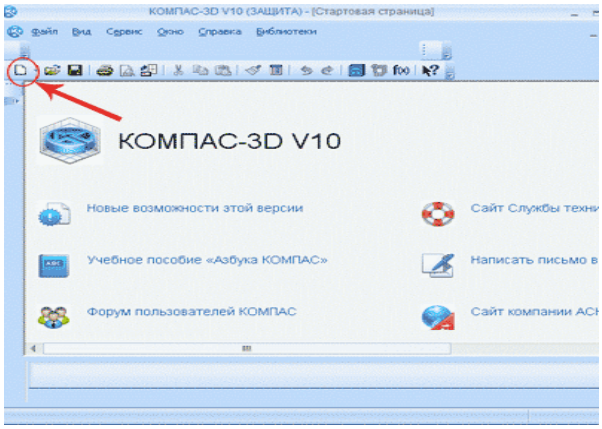


Рис.14

Появится диалоговое окно, в котором мы должны выбрать тип документа, с которым нам предстоит работать, это может быть чертеж, трехмерная

модель, сборка и т.д.. Предположим необходимо создать **«Чертеж»**, просто нажимаем на нужную иконку.

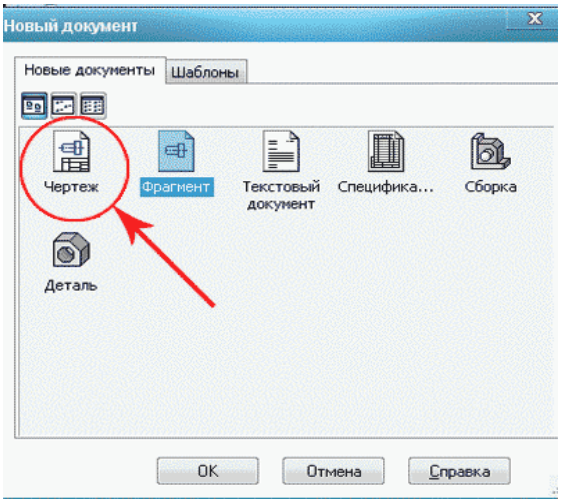


Рис.15

После этого откроется лист, на котором мы и будем создавать наш чертеж.

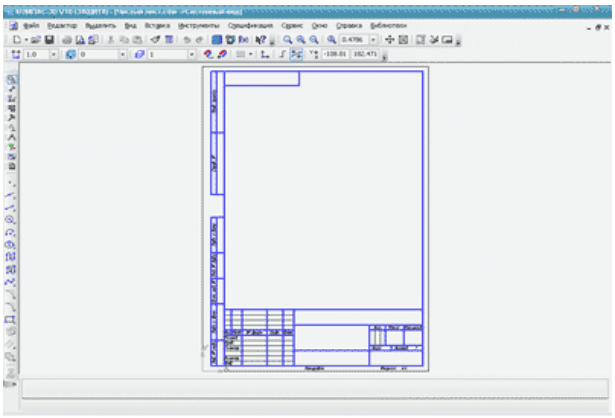


Рис.16

Урок №2. Открытие документа Компас. Сохранение документа Компас.

Продолжаем изучать программу Компас 3D, и сегодня поговорим о том, как открывать и сохранять документы Компас. На самом деле эти действия подобны тем, которые Вы выполняете при работе с другими документами. Если необходимо открыть существующий документ вызываем команды **Файл-Открыть**

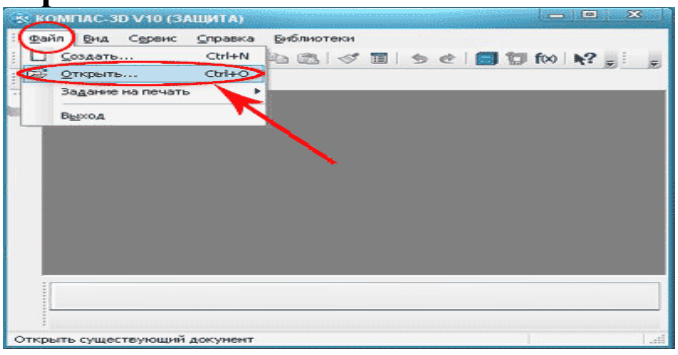


Рис.17

или нажимаем соответствующий значок на панели инструментов.

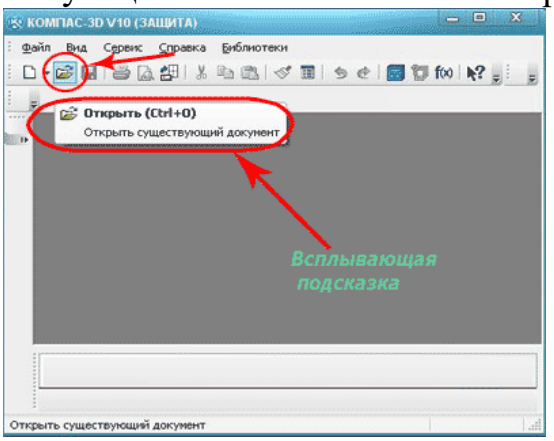


Рис.18

Появится диалоговое окно, в котором выбираем нужный документ и открываем его двойным щелчком или нажатием кнопки Открыть.

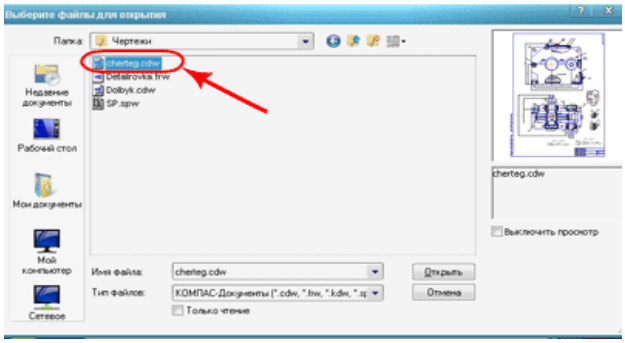


Рис.19

наш документ открыт.

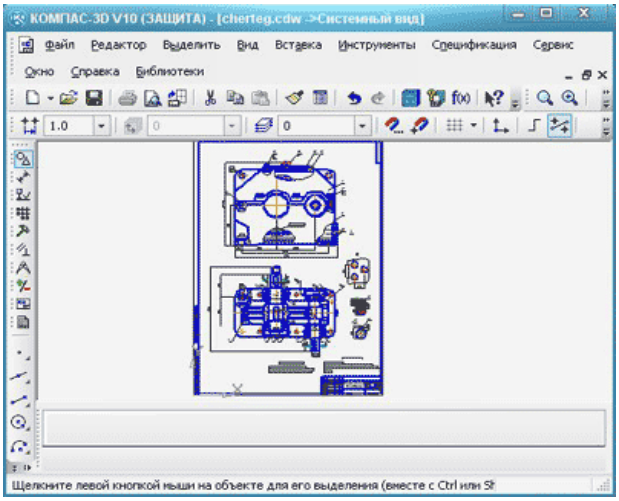


Рис.20

Само собой открыть документ Компас-3D можно с помощью проводника Windows, для этого достаточно выделить необходимый файл и щелкнуть на нем два раза левой клавишей мыши. Для сохранения документа на диск вызываем команды **Файл – Сохранить**

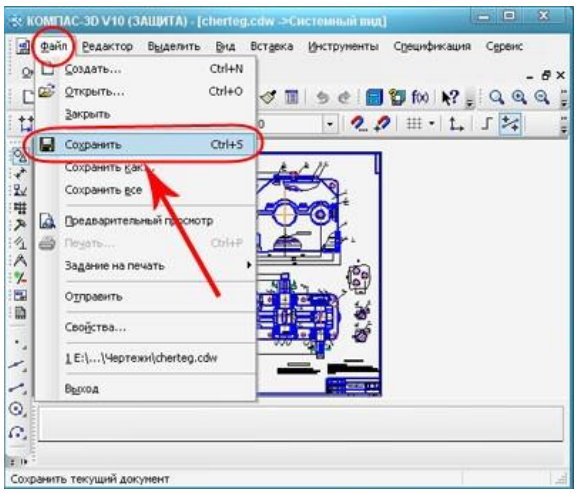


Рис.21

появится диалоговое окно, в котором выбираем папку для сохранения, вводим имя файла и нажимаем кнопку **Сохранить**. По умолчанию программа предложит расширение, которое соответствует типу документа. Изменять умолчательное расширение без крайней необходимости не следует, это затруднит поиск файла впоследствии.

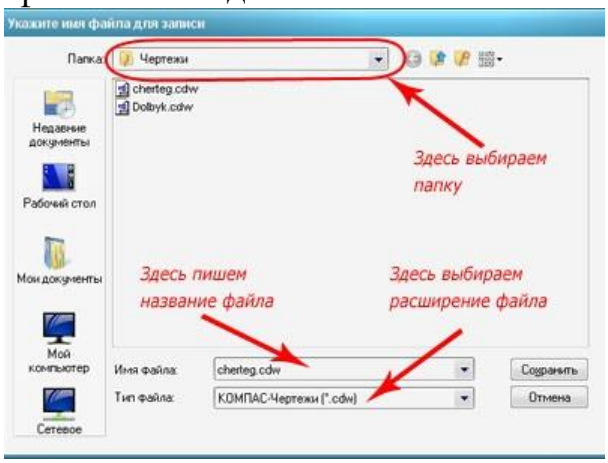


Рис.22

Если Вы отредактировали файл и хотите сохранить его под другим именем, не меняя старую редакцию файла, то вызываем команду **Файл** – **Сохранить как…**, и снова появится диалоговое окно, в котором указываем папку и имя файла.

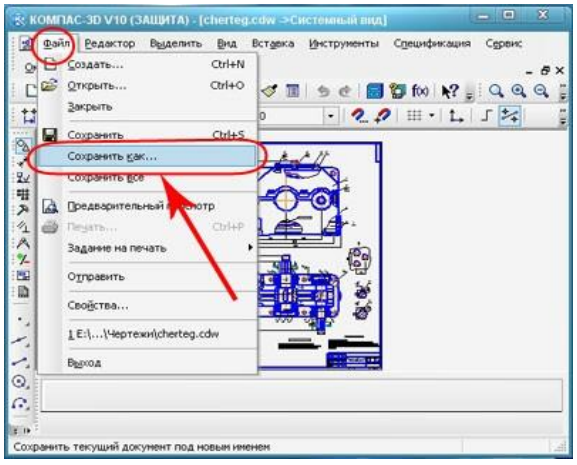


Рис.23

Есть ещё функция для сохранения всех открытых документов, команда **Файл** – **Сохранить все**, с ней Вы без труда разберетесь сами. Чтобы закрыть документ вызываем команду **Файл** – **Закрыть** (или просто нажимаем **X** в верхнем правом углу программы)

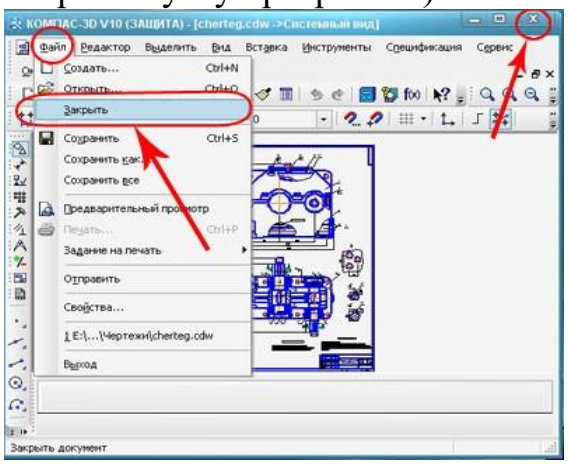


Рис.24

на экране появится запрос на подтверждение записи файла, нажимаем **ДА**, документ сохранен и закрыт.

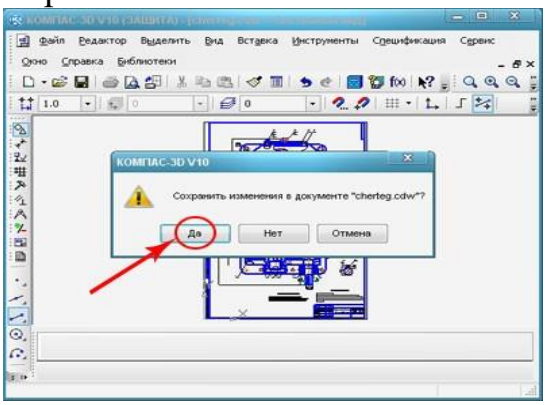


Рис.25

Пожалуй, это все, что необходимо знать об открытии, сохранении и закрытии документа. В следующем уроке начнем делать простейшие построения.

Построение точек в Компас 3D.

В Компас-3D есть несколько различных способов простановки точек, кроме того, имеется несколько стилей их оформления.

Прежде чем переходить к построению точек, давайте немного поговорим об интерфейсе системы (как помните, мы будем изучать его параллельно с изучением возможностей программы). Нам понадобятся: основное меню программы, компактная панель и панель инструментов **"Геометрия"**.Чтобы вызвать панель инструментов **"Геометрия"**на экран наберите команды **Вид – Панель инструментов – Геометрия.** Чтобы активировать её на компактной инструментальной панели необходимо нажать кнопку  - **Геометрия.**

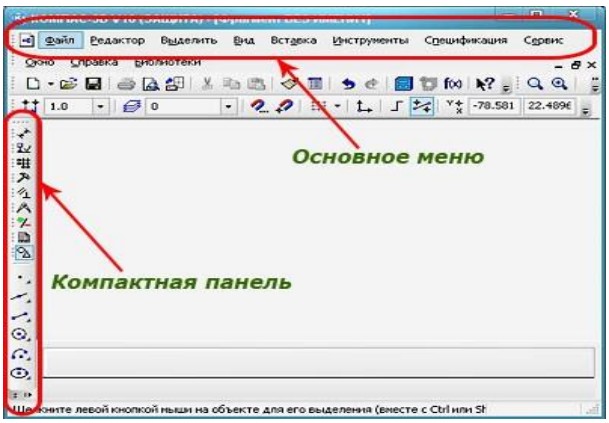


Рис.26

Обратите внимание, если инструментальная панель, входит в состав компактной, активировать её с помощью основного меню невозможно. Если Вам необходимо извлечь из компактной панели инструментальную панель, то нужно перетащить соответствующий ей маркер мышью за пределы компактной панели. Маркеры находятся рядом с кнопками переключения.

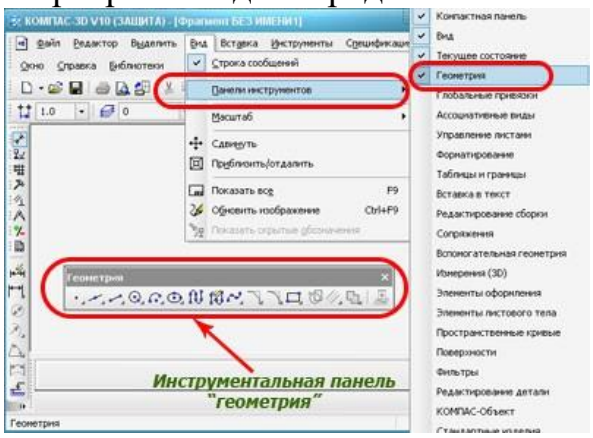
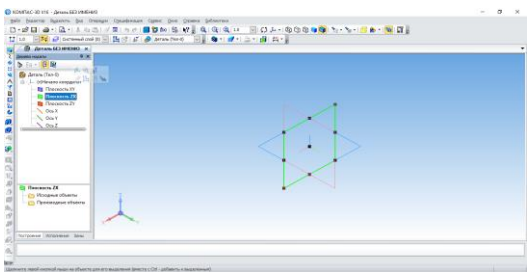


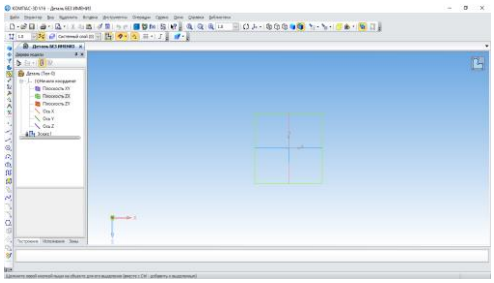
Рис.27

Если хотите вернуть или добавить в состав компактной панели инструментальную, перетащите её на компактную панель при нажатой клавише **Alt**. Когда рядом с курсором появится знак **"плюс"**, отпустите кнопку мыши и клавишу **Alt.**

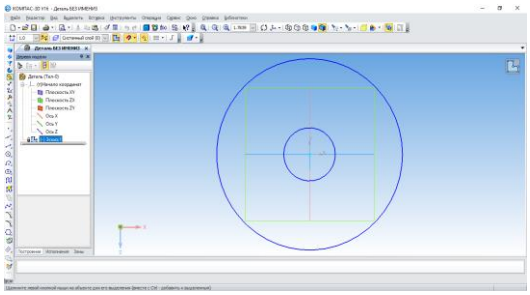
Шаг 1



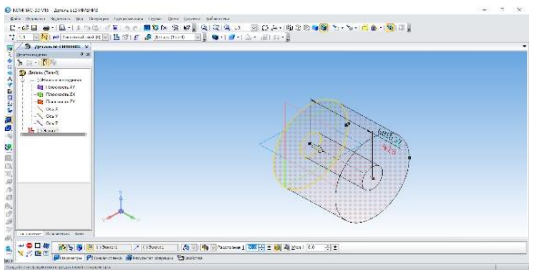
Шаг 2



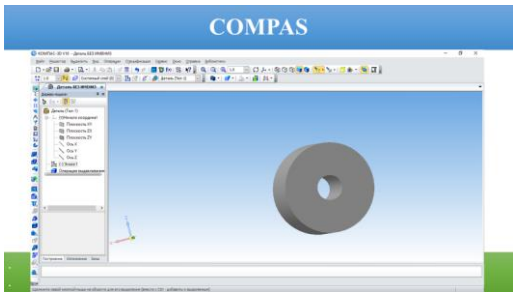
Шаг 3



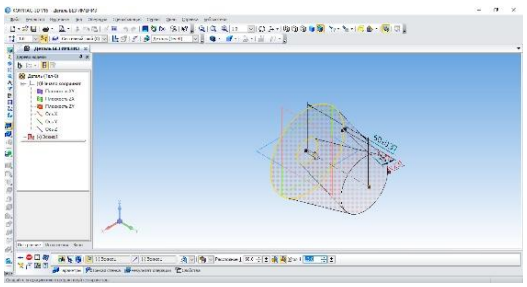
Шаг 4



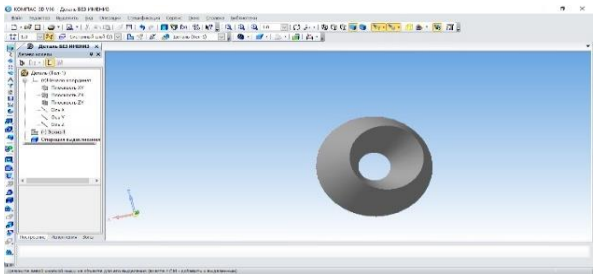
Шаг 5



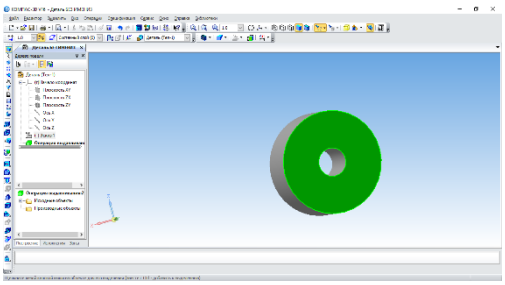
Шаг 6



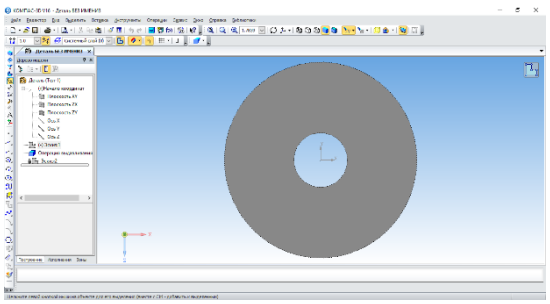
Шаг 7



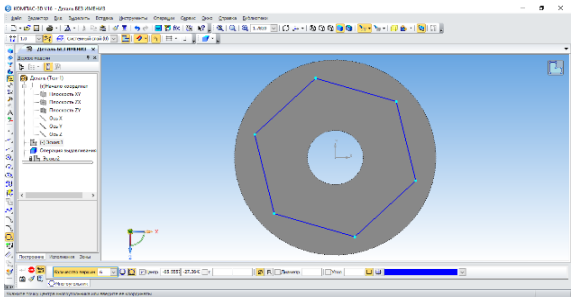
Шаг 8



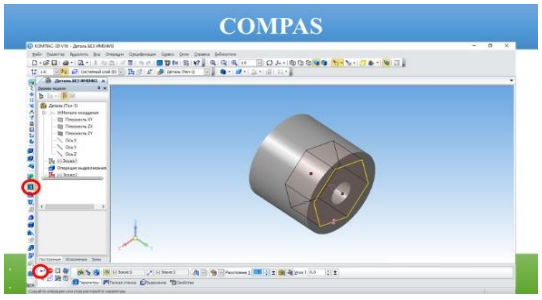
Шаг 9



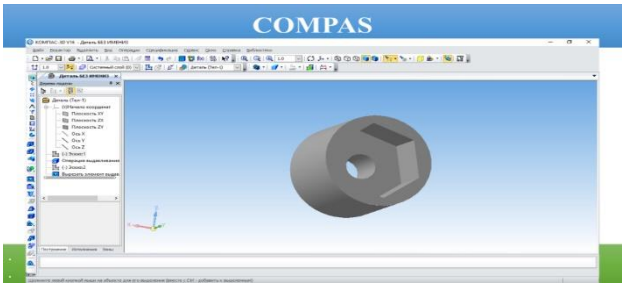
Шаг 10



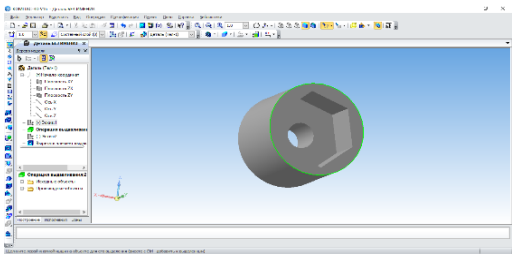
Шаг 11



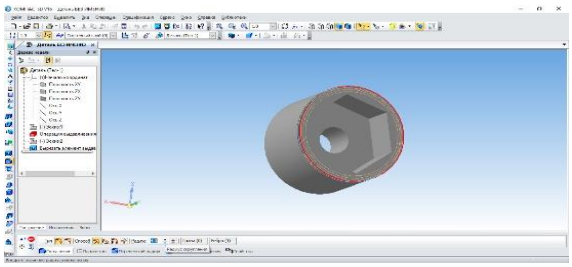
Шаг 12



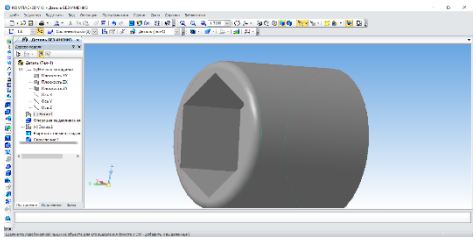
Шаг 13



Шаг 14



Шаг 15



Контрольные вопросы:

1.В чем заключается отличие COMPAS 3D других графических редакторов.
2.Какие параметры объектов: линии, отрезка прямой, отрезка кривой второго и третьего порядка нужно знать, чтобы описать объект в редакторе векторной графики?
3.Что такое кривая Безье, каковы особенности её построения?

Практическое занятие № 5
Технология объектно - ориентированного программирования в языке C++ Bulder 6

Цель работы: Освоить методику представления основных объектов программы в языке C++Builder6. Приобрести навыки объектно-ориентированного программирования.

Теоретическая часть

Программирование в среде C++Builder 6 осуществляется в двух режимах: консольном и визуальном. Консольный режим ничем не отличается от режима реализации программ на языке C, C++. В визуальном режиме программа реализуется использованием компонент и специальных окон среды C++Builder 6. Как представлено на рис.1. среда C++Builder 6- есть окно
WINDOWS со всеми его атрибутами. В структуру среды включены 5 основных окон.

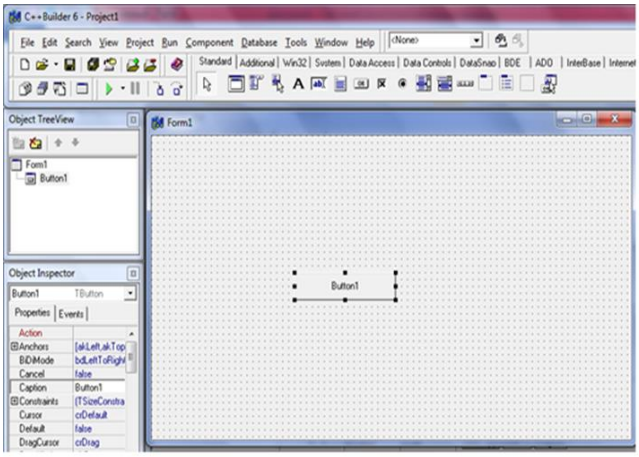


Рис.1

В консольном режиме используется окно, реализуемое командами:

File-->New-->Other--> Consol Wizard->OK

В этом окне записывается код программы.

Основными объектами программы в языке C++Builder 6 являются константы, переменные, функции, выражения и операторы. Константы могут быть числовые, символьные, логические и строковые.

Числовые константы :

56, -12, 526 -целого типа ,

0.43, -7.826, 0.2718e+1 - вещественного типа,

‘s’ ,’G’, ‘7’, ‘!’ – символьного типа,

&& , | |, !! – логические операции,

0, 1 – логические константы, означающие истина или ложь

‘Информационные технологии’ - строковая , π- константа.

Переменные

I, g, x, Y- простые переменные,

A[i], d[Ij] - переменные с индексом.

Каждая переменная имеет свое имя (идентификатор) и характеризуется типом.

Функции в языке C++Builder6.

Различают стандартные функции и функции пользователя. Стандартные функции задаются в специальной таблице. Эти функции составляют библиотеку стандартных функций с аргументами различного типа. В отличие от стандартных, функции пользователя задаются в программе самим пользователем с помощью средств языка и используются в конкретных задачах.

Прототип функции	Описание функции
char *strcpy(char	Копирует строку s2 в массив символов s1. Возвращает значение s1.

*sl, const char *s2i)	
char *strncpy(char *s1r const char *s2, size__t n)	Копирует не более n символов из строки s2 в массив символов s1. возвращает значение s1.
char *strcat(char *sl, conet char *s2)	Добавляет строку s2 к строке s1. Первый символ строки s2 записывается поверх завершающего нулевого символа строки s1.
char *strncat(char *sl, cons t char * s 2, size_t n)	Добавляет не более n символов строки s2 в строку s1. Первый символ из s2 записывается поверх завершающего нулевого символа в s1. Возврата значение s1.
int strcmp(conat char *sl, const char *s2)	Сравнивает строки s1 и s2. Функция возвращает значение 0, меньшее чем 0 или большее, чем 0, если s1 соответственно равна, меньше или больше, чем s2.
int strcmp(const char *sl, const char *s2, size_t n)	Сравнивает до n символов строки si со строкой s2. Функция возвращает значение 0, меньшее, чем 0 или большее, чем 0, если s1 соответственно равна, меньше или больше, чем s2
char *strtok (char *sl, const, char *s2)	Последовательность вызовов strtok разбивает строку s1 на «лексемы» - логические куски, такие как слова в строке текста - разделенные символами, содержащимися в строке s2. Первый вызов содержит в качестве первого аргумента s 1, а последующие вызовы для обработки той же строки, содержат в качестве первого аргумента NULL.
size t strlen (const_char *s)	Определяет длину строки s. Возвращает количество символов, предшествующих завершающему нулевому символу.

Стандартные функции для числовых переменных.

sinx	**sin(x)**	
cosx	**cos(x)**	
tgx	**tan(x)**	**sin(x)/ cos(x)**
lnx	**log(x)**	
expx	**exp(x)**	

x^2	x*x или pow(x,2)
|x|	abs(x)
$\sqrt{x}$	sqrt(x)
Cos ^{2}x ^{3}	pow(cos(pow(x,3),2))
X^4	pow(x,4)
A ^{2}exp lna	pow(a,x)*exp(x*log(a))
$(1+x)^2$	pow(1+x,2)
arctg(x)	atan(x)

Выражения

Выражение в языке C++Builder6 представляет собой совокупность операндов, связанных между собой соответствующими операциями. Каждый операнд представляется константой, переменной или функцией соответствующими типу выражения. Так выражения могут быть числовыми, символьными, логическими, строковыми. Используются арифметические, логические операции и операции отношения.

(x*x)+sin(a-x)-1 - числовое выражение.

x && y || ! z - логическое выражение.

Операторы

Операторы языка подразделяются на простые и структурированные.

Простые: оператор присваивания, перехода, ввода-вывода.

Структурированные: условный, оператор выбора, циклические операторы.

Программирование простейших алгоритмов связано с использованием основных конструкций языка в последовательности операторов в блоке программы. Единственным оператором, изменяющим значения переменных программы, является оператор присваивания. Y=a*x+b; Пример простейшей линейной программы в консольном режиме.

Задача 1.Найти расстояние между двумя точками A, B с заданными координатами X1, Y1 и X2, Y2.

```
//---------------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include <vcl.h>                                  // директивы препроцессора
#include<math.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
void main();
{
cin>> a,b,x1,x2,y1,y2,r;                          //программа оформляется в виде функции
float x1=4.6, x2=6.9, y1=7, y2=2.5, r;
r=pow(pow(x1-x2, 2)+pow(y1-y2,2),1/2);
```

```
cout<<"r= "<< r <<endl;
getch();
return 0;
}
//---------------------------------------------------------------------------
```

Практическая часть

Система C++ Builder6 – это сложный механизм, обеспечивающий эффективную работу программиста, используя язык C++. Визуальная среда реализуется 5 основными одновременно открытыми окнами (рис.1):

1)Главное окно;
2)Окно формы;
3)Окно кода программы;
4)Окно инспектора объектов;
5)Окно – дерево объектов.

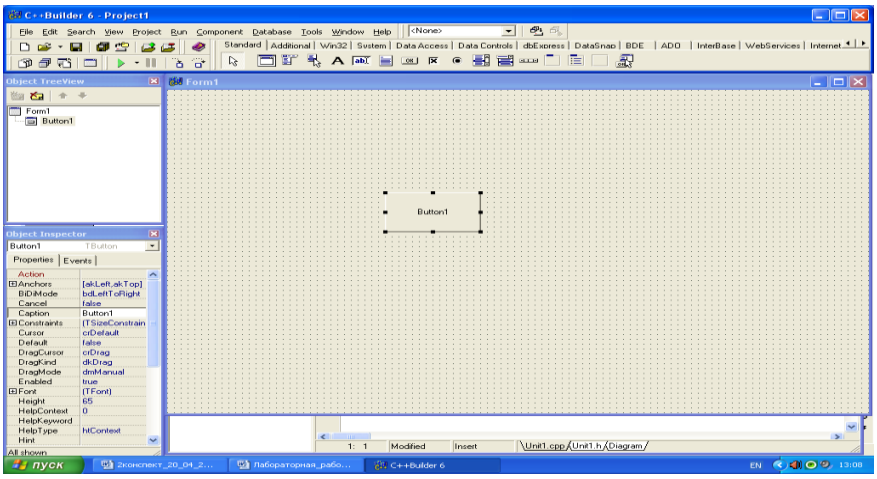


Рис.2. Окно приложения среды C++ Builder6

Главное окно - является окном приложения C++ Builder6 и присутствует всегда при работе с этой системой. Оно выполняет основные функции управления**,** благодаря совокупности команд предоставляемых меню. Кроме того, функции управления могут быть реализованы с помощью пиктографического меню, т.е. панелью инструментов, которые задаются в левом углу под главным меню. В правой части главного окна расположена палитра компонентов, которая имеет вкладки: BDE, Standart,, Additional и др.

Каждая страница этой палитры предоставляет программисту совокупность функциональных элементов, содержащихся в окне инспектора объектов - Свойства.

С помощью этих компонентов создается каркас программы, а именно: окна, кнопки, списки, которые устанавливаются в окне «Форма». В этом помогает специальный редактор системы.

Окно формы представляет собой проект Windows окна будущей программы. Вначале оно содержит только интерфейсные элементы. Вся рабочая область представляет собой координатную сетку для

упорядочивания размещаемых на форме компонентов. Таким образом создается интерфейс будущей программы. Именно этот процесс заполнения формы нужными компонентами и представляет собой визуальное программирование.

Окно кода программы предназначено для создания и редактирования текста программы. Этот текст описывает алгоритм функционирования программы. В системе C++ Builder6 используется язык Object Pascal. Причем система C++ Builder6 автоматически берет на себя многие рутинные аспекты программирования и знание языка Object Pascal является непременным условием для любого программиста. Код программы оформляется в виде модуля (Unit). Первоначально окно содержит минимальный исходный текст. Пользователь же оформляет исполняемую часть модуля, которая задает алгоритм решения задачи.

Окно инспектора объектов - связано с окном формы, т.к. компоненты, размещенные на форме, могут изменять не только свое положение и размеры непосредственно на форме, но также могут изменять свои свойства и реагировать на отдельные события, это и задается двумя вкладками окна инспектора объектов: свойства и события.

Окно дерево объектов – в этом окне отражаются все объекты программы в виде дерева, т.е. можно видеть структуру объекта.

В среде C++ Builder6 можно использовать два режима программирования:

1) профессиональный режим визуального программирования.

2) простейший консольный режим, который активизируется выбором меню « **Файл» -> NEW -> Other-> Concole Wizard->OK.**

Создание нового проекта приложения начинается с команды File | New Application. По этой команде открывается новый проект приложения с пустой формой. Сохранить на диске готовый проект или его заготовку можно командой File | Save Project As или File | Save All. Открыть ранее сохраненный проект можно командой File | Open или File | Open Project. Но если вы недавно работали с этим проектом, то удобнее воспользоваться командой File | Reopen. Эта команда дает вам возможность быстро выбрать проект из числа тех, с которыми вы работали в последнее время.

Задача 2 . Вычислить заданную функцию с конкретными значениями переменных.

$$y = \frac{ax^2 + bx + e^x}{\sqrt{1 + Cosx}} * x^a$$

В этой лабораторной работе используются компоненты Button, Lable, Edit. Число **Edit** зависит от того, сколько данных нужно ввести вручную пользователю. Button1 используется для вычисления Y.

Код программы:

```
//---------------------------------------------------------------------------
#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 __fastcall TForm1::Button1Click(TObject *Sender)
{
 float a,b,x,y;
 a= StrToFloat(Edit1->Text);
 b= StrToFloat(Edit2->Text);
 x= StrToFloat(Edit3->Text);
 y=((a*pow(x,2)+b*x+exp(x))/(sqrt(1+cos(x))))*pow(x,a);
 Label5->Caption = "Y="+FloatToStr(y);
}
//---------------------------------------------------------------------------
```

При выполнении программы будет создана Форма и получены следующие данные(рис.3).

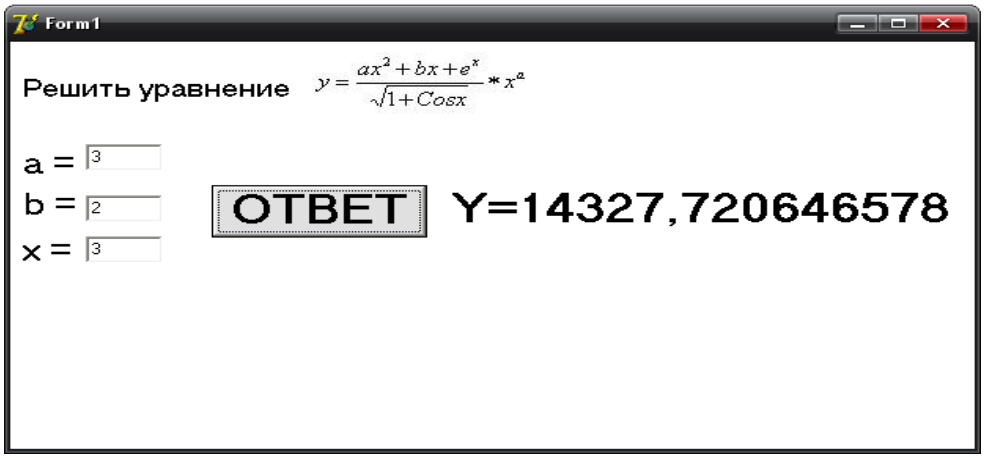


Рис.3. Созданная Форма и полученные результаты.

Контрольные вопросы:

1. В чем заключается назначение основных разделов программы в языке C++.
2. Какие команды включены в пункты основного меню интегрированной среды C++?
3. Как осуществляется загрузка программного файла в интегрированной среде?
4. Основные окна интегрированной среды C++ Builder6. и их назначение.

5. Последовательность выполнения программного файла в C++ Builder6..
6. Какие основные операторы используются при программировании линейного процесса?
7. Как осуществляется просмотр результатов программы?

Практическая работа №6
Технология логического программирования

Цель работы : Изучить методику создания программ с использованием циклических операторов C++Bulder6.

Задания:

1. Изучить теоретическую часть.
2. Загрузить систему C++ Builder6.
3. Реализовать программу своего варианта и получить результат.
4. Составить отчет о проделанной работе.

Теоретическая часть

Операторы цикла . Под оператором цикла понимается указание программе повторять некоторую последовательность операторов заданное количество раз или пока выполняется некоторое условие. Всего в C++ три вида циклов.

- с предусловием (while),
- с постусловием (do while),
- с параметром (for).

Любой цикл состоит из тела цикла, то есть тех операторов, которые выполняются несколько раз, начальных установок, модификации параметров цикла и проверки условия продолжения выполнения цикла.

Один проход цикла называется итерацией. Проверка условия выполняется на каждой итерации либо до тела цикла (цикл с предусловием), либо после тела цикла (цикл с постусловием). Отличие этих двух способов в том, что в цикл с предусловием возможно не выполнится ни разу, если условие с самого начала окажется ложным, а цикл с постусловием гарантировано выполнится хотя бы один раз.

Цикл завершается, если не выполнится условие его продолжения. При необходимости возможно принудительное завершение всего цикла оператором **break** или текущей итерации оператором **continue.**

Цикл while (с предусловием).

Формат:

while (выражение) оператор

Выполнение оператора начинается с проверки условия в скобках после while. Если условие истинно, выполняется оператор цикла, иначе управление

передается оператору, следующему за циклом. Если при первой же проверке условие не выполняется (равно false), то цикл не выполнится ни разу.

Потенциальный источник ошибок если в теле цикла условие не изменяется, но оно является истинным, цикл будет бесконечным. Это также можно использовать в своих целях, например, объявить цикл:

while (true) { . . . } ; и организовать выход изнутри цикла с помощью **break.**

Цикл do while (с постусловием).

Формат:

do оператор **while** выражение;

Цикл **do while** отличается от цикла **while** только тем, что в нем проверка условия происходит после цикла, то есть он достоверно выполнится хоть раз.

Алгоритм данного цикла следующий: вначале выполняется простой или составной оператор, являющейся телом цикла, затем проверяется выражение. Если оно не равно **false,** тело цикла выполняется ещё раз. Цикл завершается, когда выражение становится равным **false**, или если происходит выход из цикла при помощи оператора передачи управления.

Одним из приложений цикла **do while** является ситуация, когда ввод какой-то величины должен продолжаться пока она не примет определенное значение, например:

```
do {
   cout << “Введите число от 1 до 100 (0 – выход) : “;
   cin >> num;
   . . . // действия счислом
while (num ! = 0);
   }
```

Цикл for (с параметром).

Формат:

for (инициализация; выражение; модификации) оператор;

Цикл с параметром – один из мощнейших инструментов языка C++BULDER6. Он позволяет не только повторять последовательность операторов заданное число раз, но и создавать сложные циклы с условиями выхода и прочими продвинутыми возможностями.

Схема работы цикла **for** следующая:

- Выполняется инициализация. Она производится лишь однажды перед началом цикла. Обычно инициализация содержит установку переменной-счетчику начального значения.
- Как и в цикле **while**, проверяется истинность выражения, и если оно равно **true,** выполняется тело цикла, иначе оно пропускается.
- В заключение производятся операции, описанные в секции модификаций, обычно это изменение управляющей переменной.

Рассмотрим на примерах, какие циклы можно создать, используя **for**:

for (**int** n = 1; n<= 10; n++) { **. . .** }

Это наиболее очевидное применение for – переменная n принимает все значения от 1 до 10, то есть тело цикла выполнится 10 раз (если n не меняется в теле цикла).

for (**int** i = 1, j = 100; i != j; i ++, j --) {**. . .**}

Это более сложный пример условия. Здесь в секции инициализации начальные значения присваиваются двум переменным, после каждой итерации одна из них увеличивается на единицу, а вторая уменьшается на единицу. Вывод из цикла происходит, когда значение обеих переменных равны, то есть всего будет 100/2 = 50 итераций.

for (int k = 1; ; k++) { **. . .** }

В данном случае опущено условие выхода из цикла – на его месте стоит *пустой оператор,* состоящий только из знака; (точка с запятой). Значит тело цикла будет выполняться, пока внутри не встретится оператор передачи управления, но в конце каждой итерации значение переменной-счетчика k увеличивается на единицу.

for (int i = 1; i <= 10; a[i] = i, i ++);

В этом, последнем примере, рассматривается случай, когда необходимо проинициализировать массив из десяти элементов некоторыми значениями. Этот цикл не имеет даже тела – все необходимые операции выполняются уже в заголовке, в секции модификаций. После очередной итерации сначала происходит присваивание значения очередному элементу массива, а затем собственно приращение счетчика.

Оператор do while удобен, когда цикл обязательно должен быть выполнен хотя бы раз (например, в нем есть ввод данных с последующей проверкой правильности ввода).

Оператор for идеален для организации цикла, управляемого счетчиком, но также удобен во многих других случаях.

В остальных циклах можно применять while, особенно если число итераций неизвестно заранее, очевидных параметров цикла не имеется или их удобно модифицировать в конце цикла.

Решение одного варианта:

Задача 1. Вычислить значение функций f1 и f2 при значении x, изменяющегося в интервале от a до b с шагом h.

$f1=x^2+1*e^{-x}$ f2=1+2sinx

//----------

```
 #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<<"Vvedite znachenie 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;
}
//---------------------------------------------------------
```

Задача 2. Реализовать задачу вычисления функции F в интервале **x a,b** с шагом **h.** Для организации цикла используем оператор с постусловием**.**

В этой работе применяются компоненты Memo, Label, Edit, Button

```
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Button1: TButton;
Label6: TLabel;
Memo1: TMemo;
```

Button1 используется для вычисления F. Memo1 применяются для отображения всех F на диапазоне [A,B].

Код программы:

```
//---------------------------------------------------------------------------
    #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));
    x=x+h; }
  while (x<=b);
}
//---------------------------------------------------------
```

При выполнении программы результаты будут получены в следующей Форме (рис.6):

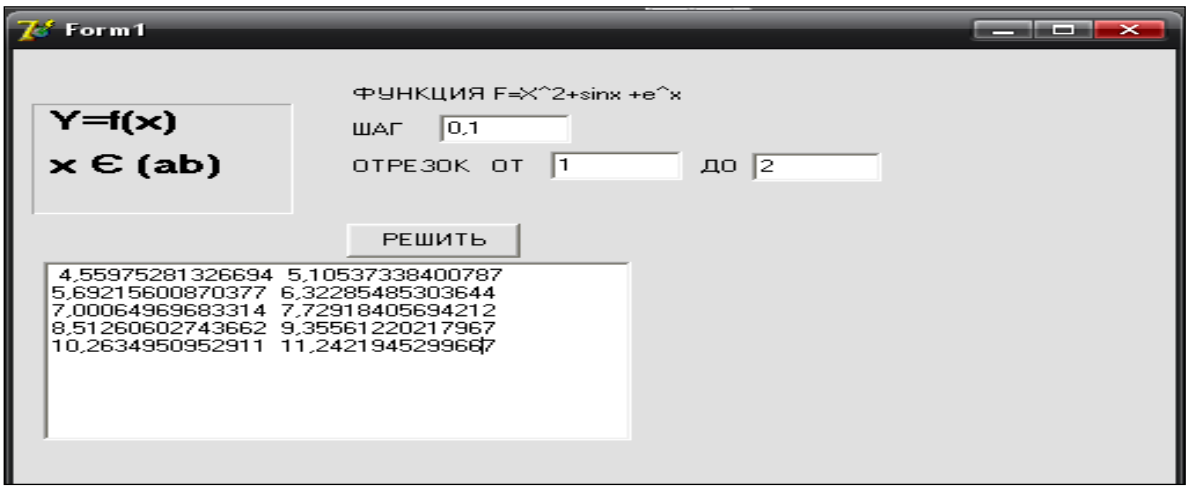


Рис.1. Форма с полученными результатами

Варианты заданий для индивидуального выполнения:

Для каждого ***x***, изменяющегося от ***a*** до ***b*** с шагом ***h***, найти значения функции $Y(x)$, суммы $S(x)$ и $|Y(x)–S(x)|$ и вывести в виде таблицы. Значения a, b, h и n вводятся с клавиатуры. Так как значение $S(x)$ является рядом разложения функции $Y(x)$, при правильном решении значения S и Y для заданного аргумента x (для тестовых значений исходных данных) должны совпадать в целой части и в первых двух-четырех позициях после десятичной точки. Работу программы проверить для $a = 0{,}1$; $b = 1{,}0$; $h = 0{,}1$; значение параметра n выбрать в зависимости от задания.

1. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$, $Y(x)=\sin(x)$.

2. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}$, $Y(x)=x\cdot arctg(x)-\ln\sqrt{1+x^2}$.

3. $S(x)=\sum_{k=0}^{n}\frac{\cos(k\pi/4)}{k!}x^k$, $Y(x)=e^{x\cos\frac{\pi}{4}}\cos(x\sin(\pi/4))$.

4. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k}}{(2k)!}$, $Y(x)=\cos(x)$.

5. $S(x)=\sum_{k=0}^{n}\frac{\cos(kx)}{k!}$, $Y(x)=e^{\cos x}\cos(\sin(x))$.

6. $S(x)=\sum_{k=0}^{n}\frac{2k+1}{k!}x^{2k}$, $Y(x)=(1+2x^2)e^{x^2}$.

7. $S(x)=\sum_{k=1}^{n}\frac{x^k\cos(k\pi/3)}{k}$, $Y(x)=-\frac{1}{2}\ln(1-2x\cos\frac{\pi}{3}+x^2)$.

8. $S(x)=\sum_{k=0}^{n}\frac{(2x)^k}{k!}$, $Y(x)=e^{2x}$.

9. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k+1}}{4k^2-1}$, $Y(x)=\frac{1+x^2}{2}arctg(x)-x/2$.

10. $S(x)=\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!}$, $Y(x)=\frac{e^x+e^{-x}}{2}$.

11. $S(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!}(x/2)^k$, $Y(x)=(x^2/4+x/2+1)e^{x/2}$.

12. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{2k^2+1}{(2k)!}x^{2k}$, $Y(x)=(1-\frac{x^2}{2})\cos(x)-\frac{x}{2}\sin(x)$

.

13. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}$, $Y(x)=2(\cos^2 x-1)$.

14. $S(x)=\sum_{k=0}^{n}\frac{x^{2k+1}}{(2k+1)!}$, $Y(x)=\frac{e^x-e^{-x}}{2}$.

15. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}$, $Y(x)=-\ln\sqrt{1+x^2}+x\,arctg(x)$.

16. $S(x)=\sum_{k=0}^{n}\frac{\cos(k\pi/4)}{k!}x^k$, $Y(x)=\cos[x\cdot\sin(\pi/4)]e^{x\cos\frac{\pi}{4}}$.

Контрольные вопросы

1. Что такое циклический вычислительный процесс?
2. Какие операторы цикла вы знаете?
3. Какие типы переменных используются в качестве параметра в операторе цикла с параметром?
4. Различие между операторами цикла WHILE и DO WHILE.
5. Какие компоненты C++ используются при реализации циклических программ?
6. Какие процедуры применяются для обработки строк в C++?

Практическая работа № 7
Технология создания приложений в системе программирования C++ Builder 6

Цель работы: Изучить правила создания и обработки данных структурного типа с использованием файлов; правила работы с компонентами ***OpenDialog*** **и** ***SaveDialog*****;** написать и отладить программу по созданию файлов.

Теоретическая часть

Структура объединяет логически связанные данные разных типов.
Структурный тип данных определяется описанием ***шаблона***:

```
struct Person  {
char Fio[30];
double sball;
};
```

Объявление переменных созданного структурного типа:

```
Person Stud, *p_Stud;
```

Обращение к элементам структур производится посредством:

1) операции принадлежности (**.**) в виде:

ID_структуры **.** *ID_поля* или (**указатель*) **.** *ID_поля*

2) операции косвенной адресации (**–>**) в виде:

указатель **–>** *ID_поля* или &(*ID_структуры*) **.** *ID_поля*

Для приведенного выше примера

1) Stud.Fio = “Иванов А.И.”; //Инициализация данных
   Stud.sball = 5.75;
2) p_Stud –> Fio = “Иванов А.И.”;
   p_Stud –> sball =5.75;

В языке C/C++ файл рассматривается как поток (*stream*), представляющий собой последовательность считываемых или записываемых байт. При этом последовательность записи определяется самой программой.

Работа с файлами

Файл – это набор данных, размещенный на внешнем носителе и рассматриваемый в процессе обработки и пересылке как единое целое. Прототипы большинства функций по обработке файлов описаны в библиотеках ***stdio.h*** и ***io.h***.

Прежде чем работать с файлом, его нужно открыть для доступа, т.е. создать и инициализировать область данных, которая содержит информацию о файле: имя, путь и т.д. В языке C/C++ это выполняет функция ***fopen*()**, которая связывает физический файл на носителе с логическим именем в программе. Логическое имя – это указатель на файл, т.е. на область памяти, где хранится информация о файле. Указатели на файлы необходимо декларировать:

FILE **указатель на файл*;

Формат функции

fopen(“*строка* 1” , “*строка* 2”);

в *строке* 1 указывается место, в которое мы собираемся поместить файл, например: “d:\\work\\sved.txt” – файл с именем sved.txt, который будет находиться на диске *d*, в папке *work*; если путь к файлу не указывать, то он будет размещен в рабочей папке проекта.

В *строке* 2 указывается код открытия файла:

w – для записи, если файла с заданным именем нет, то он будет создан, если же такой файл существует, то перед открытием прежняя информация уничтожается;

r – для чтения; если файла нет, то возникает ошибка;

a – для добавления новой информации в конец;

r+, w+ – возможны чтение и запись информации;

a+ – то же, что и для *a*, только запись можно выполнять в любое место файла, доступно и чтение файла. По умолчанию файл открывается в текстовом режиме (***t***), указав ***b*** – файл открывается в двоичном режиме. Если

при открытии файла произошла ошибка, функция ***fopen*** возвращает значение ***NULL***. После работы доступ к файлу необходимо закрыть с помощью функции ***fclose***(*указатель файла*), например, *fclose* (*f*);
Для закрытия нескольких файлов введена функция: void ***fcloseall***(void);
Приведем пример минимального набора операторов, необходимых для корректной работы с файлом:

```
#include <stdio.h>
        . . .
    FILE *f_my;
        if( ! ( f_my = fopen(“rez.txt”, “r+t” ) ) ) {
            puts(“\n Ошибка открытия файла!”);
// В оконном режиме – ShowMessage("Ошибка открытия файла");
            return;
        }
        . . .                    //    Работа с файлом
    fclose(f_my);
        . . .
```

Для работы с текстовыми файлами в консольном приложении удобнее всего пользоваться функциями ***fprintf***() и ***fscanf***(), параметры и выполняемые действия аналогичны функциям *printf*() и *scanf*(),только первым параметром добавлен указатель файла, к которому применяется данная функция.

Функции работы с текстовыми файлами удобны при создании результирующих файлов для отчетов по лабораторным и курсовым работам. Для создания баз данных удобнее пользоваться функциями работы с бинарными файлами. Рассмотрим некоторые из них, обозначив указатель файла – ***fp*** (FILE ****fp***;):

1) int ***fread***(void ****ptv***, int ***size***, int ***n***, ***fp***) – считывает ***n*** блоков по ***size*** байт каждый из файла ***fp*** в область памяти, на которую указывает ***ptv*** (необходимо заранее отвести память под считываемый блок);
2) int ***fwrite***(void ****ptv***, int ***size***, int ***n***, ***fp***) – записывает ***n*** блоков по ***size*** байт каждый из области памяти, на которую указывает ***ptv*** в файл ***fp***;
3) int ***fileno***(***fp***) – возвращает значение ***дескриптора*** файла ***fp*** (дескриптор – число, определяющее номер файла);
4) long ***filelength***(int *дескриптор*) – возвращает длину файла в байтах;
5) int ***chsize***(int *дескриптор*, long ***pos***) – выполняет изменение размера файла ***fp***, признак конца файла устанавливается после байта с номером ***pos***;
6) int ***fseek***(***fp***, long ***size***, int ***kod***) – выполняет смещение указателя на ***size*** байт в направлении признака ***kod***: 0 – от начала файла; 1 – от текущей позиции; 2 – от конца файла;
7) long ***ftell***(***fp***) – возвращает значение указателя на текущую позицию файла ***fp*** (-1 – ошибка);
8) int ***feof***(*указатель файла*) – возвращает ненулевое значение при правильной записи признака конца файла;

9) int ***fgetpos***(*указатель файла*, long ****pos***) – определяет значение текущей позиции ***pos*** файла; при успешном завершении возвращает значение 0.

Создание оконного приложения

Компоненты OpenDialog и SaveDialog

Компоненты *OpenDialog* и *SaveDialog* находятся на странице *Dialogs*. Все компоненты этой страницы невизуальны, т.е. не видны при работе программы, поэтому их размещают в любом месте формы. Обе компоненты имеют идентичные свойства.

После вызова компоненты появляется стандартное диалоговое окно, с помощью которого выбирается имя программы и путь к ней. В случае успешного завершения диалога имя выбранного файла и его размещение содержатся в *FileName*. Для выбора файлов, отображаемых в окне просмотра, используется свойство *Filter*, а для изменения заголовка окна – используется свойство *Title*.

Задача1. Написать программу для блокнота с возможностями набора текста, сохранения(save), открытия(open) , закрытия (close) и вызова справки(help).

Для начала создадим дизайн формы. На форму ставим компоненты ***Memo1*** и ***MainMenu1***(вкладка Standard), ***SaveDialog1, OpenDialog1***(вкладкаDialogs). Дважды кликнем на компоненту ***MainMenu*** и в открывшемся окне записываем ***File-Open-Save - Close*** и рядом ***Help***. В окне ***Object Inspektor*** меняем свойства компоненты ***Memo Align - alclient***. Затем, кликнув на каждую из этих команд, записываем код программы.

```
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Soxranit1Click(TObject *Sender)
{
  if(Form1->SaveDialog1->Execute())
Form1->Memo1->Lines->SaveToFile(Form1->SaveDialog1->FileName+"txt");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Otkrit1Click(TObject *Sender)
{
 If (Form1->OpenDialog1->Execute())
```

```
Form1->Memo1->Lines->LoadFromFile(Form1->OpenDialog1->FileName);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Help1Click(TObject *Sender)
{
  ShowMessage("do you have question");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Close1Click(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------------
```

Создание оконного приложения
Настройка компонент OpenDialog и SaveDialog

На странице *Dialogs* выбрать пиктограммы  ,  для установки компонент *OpenDialog* и *SaveDialog* соответственно. Для выбора нужных файлов установить фильтры следующим образом: выбрав компоненту, дважды щелкнуть кнопкой мыши по правой части свойства *Filter* инспектора объектов, и в появившемся окне *Filter Editor*, в левой части записать текст, характеризующий выбор, в правой части – *маску*. Для *OpenDialog*1 установить значения *маски*, как показано на рис. 1. Формат **.dat* означает, что будут видны все файлы с расширением *dat*, а формат **.** – будут видны все файлы (с любыми именами и расширениями).

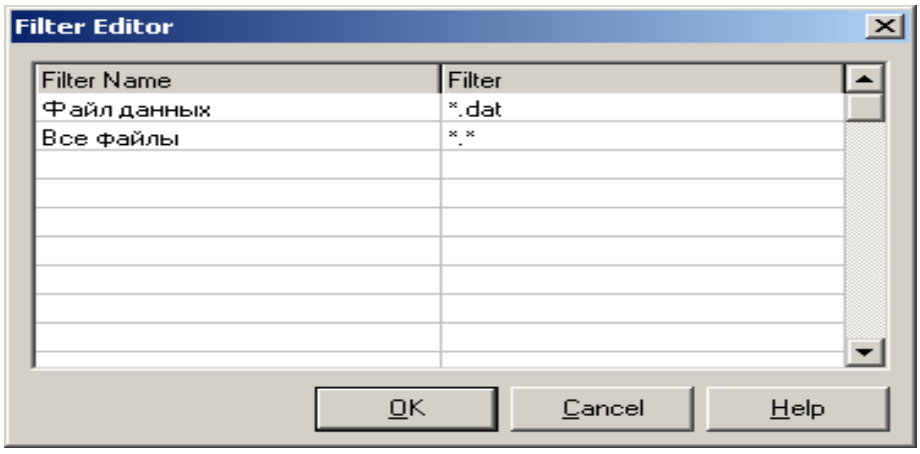


Рис.1.

Для того чтобы файл автоматически записывался с расширением *dat*, в свойстве *DefaultExt* записать требуемое расширение –. *dat.*
Аналогичным образом настраивается *SaveDialog*1 для текстового файла, который будет иметь расширение *.txt*.
Работа с программой Форма может иметь вид, представленный на рис. 8.2.

Кнопку «***Создать***» нажимаем только при первом запуске программы или, если захотим заменить прежнюю информацию на новую, в окне *Memo*1 отображается путь и имя созданного файла. Заполнив оба поля информацией, нажимаем кнопку «***Добавить***», после чего введенная информация отображается в окне *Memo*1.
Для работы с уже созданным файлом нажимаем кнопку «***Открыть***» – в *Memo*1 выводится содержимое всего файла, после чего можно добавлять новые данные в конец этого файла, не уничтожая предыдущие.
При нажатии кнопки «***Сортировать***» в *Memo*1 выводятся записи, сортированные по возрастанию рейтинга.
При нажатии кнопки «***Сохранить результаты***» создается текстовый файл, в котором сохранится информация, выведенная в *Memo*1. Этот файл можно просмотреть в любом текстовым редакторе (блокноте, *Word*).
В текст программы включена пользовательская функция void ***Out***(*TZap*, *TMemo**); – для вывода в *Memo*1 одной записи.
Для создания результирующего текстового файла используется функция, ***SaveToFile***(*FileNameRez*); позволяющая записать все содержимое *Memo*1 в файл с указанным именем.

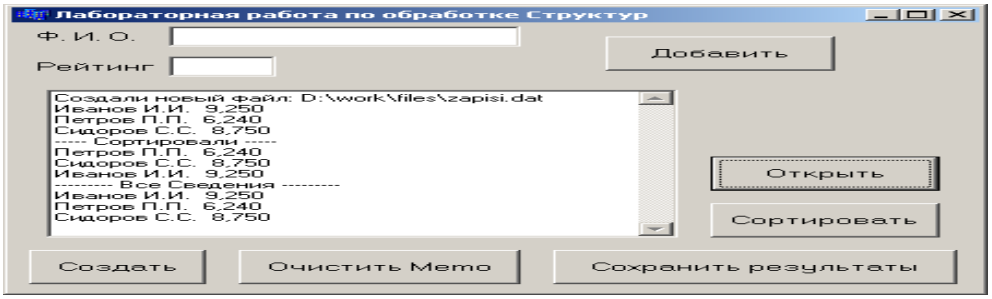


Рис.2

Текст программы может иметь следующий вид:

```
        . .
#include <stdio.h>
#include <io.h>
                . . .
//---------------------------------------------------------------------------
struct TZap{
 char FIO[30];
 double s_b;
    } Zap;
int size = sizeof(TZap);
FILE *Fz;
AnsiString File_Zap;
void Out(TZap, TMemo*);
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
                Edit1->Text="";       Edit2->Text="";
Memo1->Clear();
}
//----------------- Создать-----------------------------------------------
```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
      OpenDialog1->Title="Создать новый файл";
      if (OpenDialog1->Execute()){
            File_Zap = OpenDialog1->FileName;
            if ((Fz=fopen(File_Zap.c_str(),"wb"))==NULL)  {
                  ShowMessage("Ошибка создания ФАЙЛА!");
                  return;
            }
      }
      Memo1->Lines->Add("Создали новый файл: "+AnsiString(File_Zap));
      fclose(Fz);
                        }
//----------------- Добавить-----------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
     Fz = fopen(File_Zap.c_str(),"ab");
      strcpy(Zap.FIO,  Edit1 -> Text.c_str());
      Zap.s_b = StrToFloat(Edit2->Text);
      Out(Zap, Memo1);
      fwrite(&Zap, size, 1, Fz);
      Edit1->Text=""; Edit2->Text="";
      fclose(Fz);
                        }
//--------------------- Сортировать -------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
      TZap st, *mas_Z;
      Fz = fopen(File_Zap.c_str(),"rb");
      int D_f = fileno(Fz);                              // Находим дескриптор файла
      int len = filelength(D_f);                  // Находим размер файла
      int i, j, kol;
      kol = len/size;                                    //Количество записей в файле
      mas_Z = new TZap[kol];
// Считываем записи из файла в динамический массив
      for (i=0; i < kol; i++)
                  fread((mas_Z+i), size, 1, Fz);
      fclose(Fz);
                  Memo1->Lines->Add("Сортированные сведения");
      for (i=0; i < kol-1; i++)
            for (j=i+1; j < kol; j++)
                  if (mas_Z[i].s_b > mas_Z[j].s_b) {
                    st = mas_Z[i];
                    mas_Z[i] = mas_Z[j];
                    mas_Z[j] = st;
```

```
                }
        for (i=0; i<kol; i++)
                        Out(mas_Z[i], Memo1);
        delete []mas_Z;
}
//------------------ Сохранить ----------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
      SaveDialog1->Title="Сохранить файл результатов";
      if (SaveDialog1->Execute()) {
            AnsiString FileNameRez = SaveDialog1->FileName;
            Memo1->Lines->SaveToFile(FileNameRez);
      }
    }
//-------------------- Открыть ----------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  OpenDialog1->Title="Открыть файл";
  if (OpenDialog1->Execute()) {
          File_Zap = OpenDialog1->FileName;
          if ((Fz=fopen(File_Zap.c_str(),"rb"))==NULL)  {
               ShowMessage("Ошибка открытия ФАЙЛА!");
               return;              }
                  }
  Memo1->Lines->Add("--------- Все сведения ---------");
  while(1)              {    if(!fread(&Zap,size,1,Fz)) break;
       Out(Zap, Memo1);
  }
         fclose(Fz);              }
//----------------- Очистка Memo ----------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
      Memo1->Clear();
}
//----------------- Функция вывода одной записи ----------------------
void Out(TZap z, TMemo *Memo1)
{
   Memo1->Lines->Add(AnsiString(z.FIO)+ "  "+FloatToStrF(z.s_b,
ffFixed,6,3));
}
```

Для работы с файлами на форме можно использовать компоненты с созданием меню и подменю, с помощью которых можно организовать выполнение различных операций над файлами, например, с командами сохранить, открыть и т.д., установив соответствующие кнопки для приложения.

- ***TMainMenu*** – используется для создания главного меню. Меню создается в следующей последовательности
- 1)Установка на форме компоненты *TMainMenu*;

2)В инспекторе объектов с помощью свойства *Items* выводим дизайн меню;

3) В дизайне меню создаем вкладки.

OpenDialog – компонента для открытия нужного файла ***SaveDialogs*** – компонента для сохранения нужного файла.

Эти компоненты находятся на странице ***Dialog***. Познакомимся с некоторыми из них.

Задание.2 Открываем нужную форму. Выполняем следующую последовательность действий:

a) Устанавливаем на форме из страницы Standart компоненту *MainMenu*. Компонента *MainMenu* устанавливается для создания меню. Двойной клик на пиктограмме активизирует компоненту и позволяет вводить пункты меню задав в C***aption*** имена вкладок.

b) Устанавливаем из страницы Standard компоненту ***Memo***. В инспекторе объектов в свойстве ***Lines*** удаляем ***Memo1.,*** а свойство ***align*** меняем на ***alclient***.

d) На странице D***ialogs*** выбираем из меню компоненту ***OpenDialogs*** и устанавливаем на форме. В инспекторе объектов в свойстве ***Filter*** **вводим** ****.txt***.

e) Из страницы D***ialogs*** выбираем пункт меню ***SaveDialogs*** и устанавливаем на форме. В инспекторе объектов в свойстве ***Filter*** **вводим** ****.txt***.

Код программы:

```
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm1::Save1Click(TObject *Sender)
{
  if(Form1->SaveDialog1->Execute())
Form1->Memo1->Lines->SaveToFile(Form1->SaveDialog1-
>FileName+"txt");
}
//---------------------------------------------------------------------
void __fastcall TForm1::Open2Click(TObject *Sender)
{
```

```
 if(Form1->OpenDialog1->Execute())
 Form1->Memo1->Lines->LoadFromFile(Form1->OpenDialog1-
>FileName);
}
//---------------------------------------------------------------------
void __fastcall TForm1::Help1Click(TObject *Sender)
{
 ShowMessage("do you have question";
}
//---------------------------------------------------------------------
void __fastcall TForm1::Close1Click(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------
```

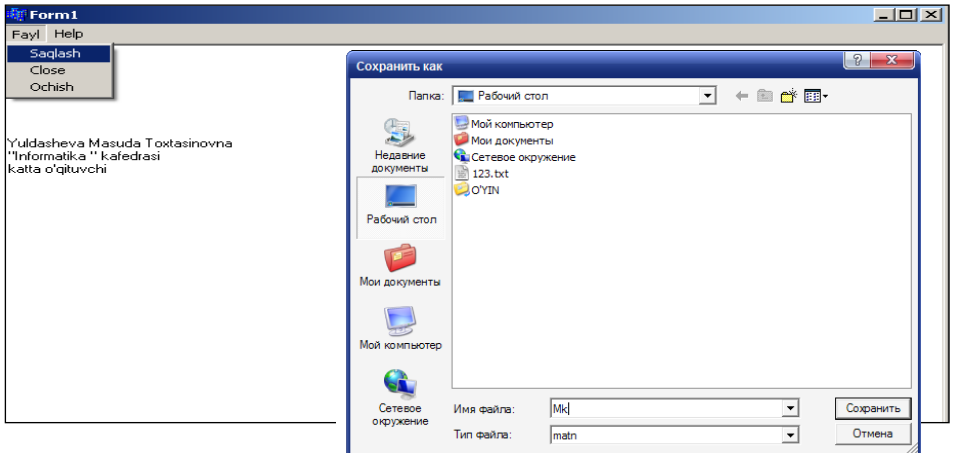


Рис.3. Окно созданного меню.

Индивидуальные задания

Написать программу обработки файла типа запись, содержащую следующие пункты меню: «Создание», «Просмотр», «Коррекция» (добавление новых данных или редактирование старых), «Решение индивидуального задания». Каждая запись должна содержать следующую информацию о студентах:

– фамилия и инициалы;
– год рождения;
– номер группы;
– оценки за семестр: по физике, математике, информатике, химии;
– средний балл.

Организовать ввод исходных данных, средний балл рассчитать по введенным оценкам.

Содержимое всего файла и результаты решения индивидуального задания записать в текстовый файл.

1. Распечатать анкетные данные студентов, сдавших сессию на 3, 4 и 5.

2. Распечатать анкетные данные студентов-отличников, фамилии которых начинаются с интересующей вас буквы.

3. Распечатать анкетные данные студентов-отличников из интересующей вас группы.

4. Распечатать анкетные данные студентов, фамилии которых начинаются с буквы *А*, и сдавших математику на 2 или 5.

5. Распечатать анкетные данные студентов, имеющих оценки 4 или 5 по физике и оценку больше 3 по остальным предметам.

6. Распечатать анкетные данные студентов интересующей вас группы. Фамилии студентов начинаются с букв *В, Г* и *Д*.

7. Распечатать анкетные данные студентов, не имеющих оценок меньше 4 по информатике и математике.

8. Вычислить общий средний балл всех студентов и распечатать список студентов со средним баллом выше общего среднего балла.(55)

9. Вычислить общий средний балл всех студентов и распечатать список студентов интересующей вас группы, имеющих средний балл выше общего среднего балла.

10. Распечатать анкетные данные студентов интересующей вас группы, имеющих неудовлетворительную оценку (меньше 4).

11. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 3,4 или 5 по информатике.

12. Распечатать анкетные данные студентов, имеющих оценки 3 или 4 по физике и оценки 4 или 5 по высшей математике.

13. Вычислить общий средний балл студентов интересующей вас группы и распечатать список студентов этой группы, имеющих средний балл выше общего.

14. Распечатать анкетные данные студентов-отличников интересующей вас группы.

15. Распечатать анкетные данные студентов интересующей вас группы, имеющих средний балл выше введенного с клавиатуры.

16. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 3 по физике и оценку 3 по высшей математике.

Практическое занятие № 8
Создание базы данных в Интернете с помощью программных средств C++ Builder6

Цель работы: Создание базы данных (MS Access) в Интернете с помощью программных средств C++ Builder6.

Теоретическая часть

Соединение с базой данной с помощью технологии ADO (от англ. ActiveX Data Objects — «объекты данных ActiveX») — интерфейс программирования приложений для доступа к данным, разработанный компанией Microsoft.

Для создания формы использовать компоненты:

Label – для подписей

Button – для инициирования действий

Edit – для вывода количества полей (колонок) и записей (строк) таблицы

ADOConnection –компонент для подключения к базе данных

ADOTable – компонент для работы со структурой и данными таблицы базы данных

DataSource – компонент для передачи данных компоненту DBGrid и DBNavigator.

DBGrid – компонент для визуализации таблицы из БД

DBNavigator – компонент для редактирования записей подключенной таблицы БД

Рекомендуемая компоновка формы программы представлена на рисунке 1.

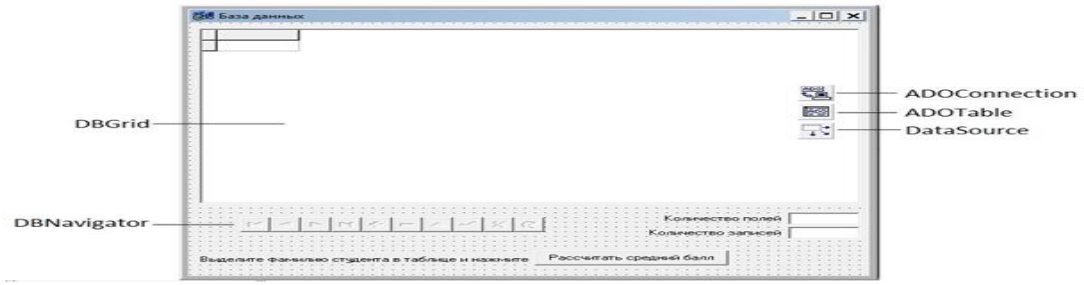


Рис.1 Рекомендуемая компоновка формы

Задание 1. Написать программу для работы с базой данных MS Access, содержащей оценки студентов по изучаемым дисциплинам. Программа должна подключаться к базе данных, содержащей оценки студентов по изучаемым дисциплинам. База данных должна быть создана в C Программа должна позволять пользователю создавать, редактировать и удалять записи. Кроме этого, программа должна подсчитывать средний балл для выбранного в таблице студента.

Рекомендации для выполнения практической работы:

1) Создать базу данных «Ведомость» в MS Access. Запустить Microsoft Office Access. В появившемся окне выбрать пункт «Новая база данных» и указать путь для сохранения базы данных, для этого нажать на кнопку напротив поля «Имя файла».

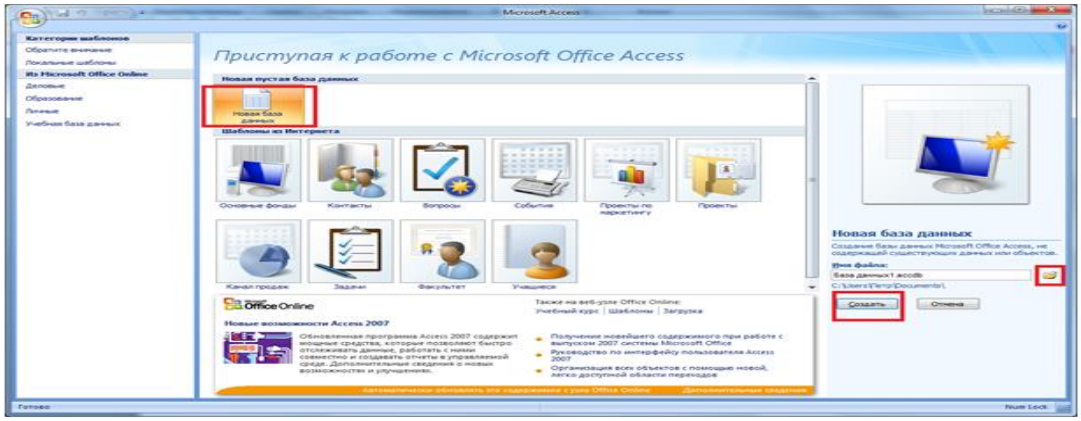


Рис.2 Создание БД

2) В появившемся окне выбрать расположение создаваемого файла БД и указать тип «Базы данных Microsoft Office Access 2002-2003 (*.mdb)» как показано на рисунке 2.

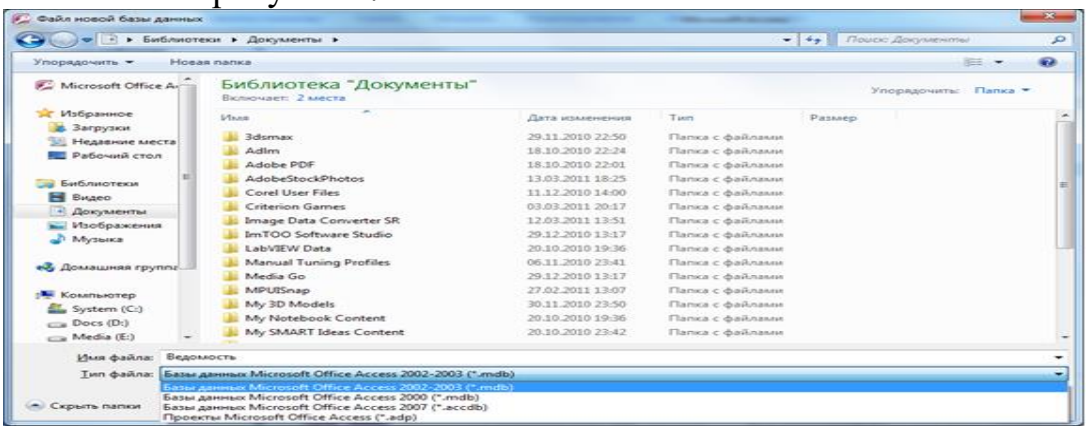


Рис. 3. Выбор типа файла БД

3) Нажать кнопку «Создать» (рисунок 2).

4) В появившемся окне нажать кнопку режим (рис.3) и задать имя таблицы «Студенты»

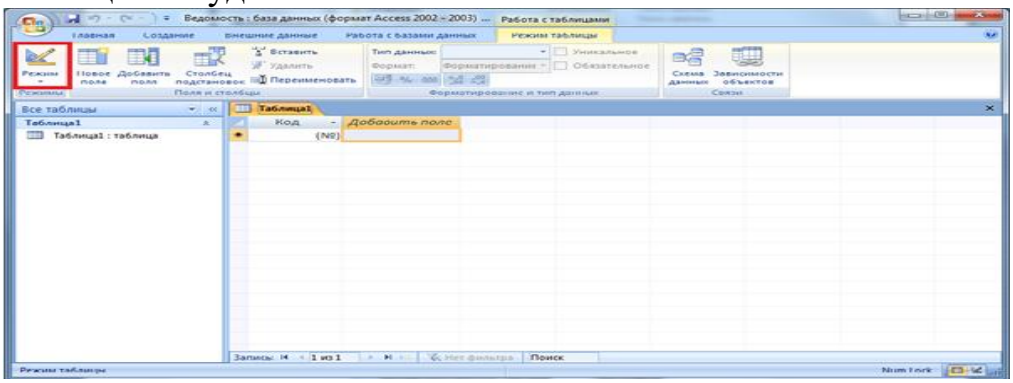


Рис.4 Переключение режима «Конструктор»

1) В БД создать таблицу «Студенты», структура которой показана на рисунке 5.

Студенты

Имя поля	Тип данных
Номер зачетки	Числовой
ФИО	Текстовый
Информатика	Числовой
Геодезия	Числовой
Математика	Числовой
Физика	Числовой
Моделирование систем	Числовой

Рис.5. Поля таблицы «Студенты»

Поле «Номер зачетки» сделать ключевым (уникальным), для этого выделить это поле и нажать кнопку на панели инструментов:

Для поля ФИО указать длину поля 50 символов:

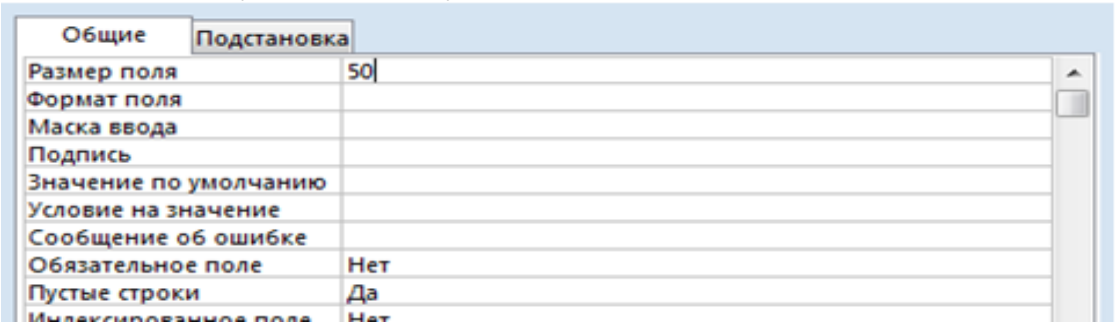


Рис. 6. Настройка размера текстового поля (количества символов)

Остальные поля должны быть числовыми. Они будут содержать оценку за соответствующую дисциплину.

6) Сменить режим на «Таблица», нажав кнопку «Режим» на панели инструментов

7) В таблицу ввести несколько записей.

Например:

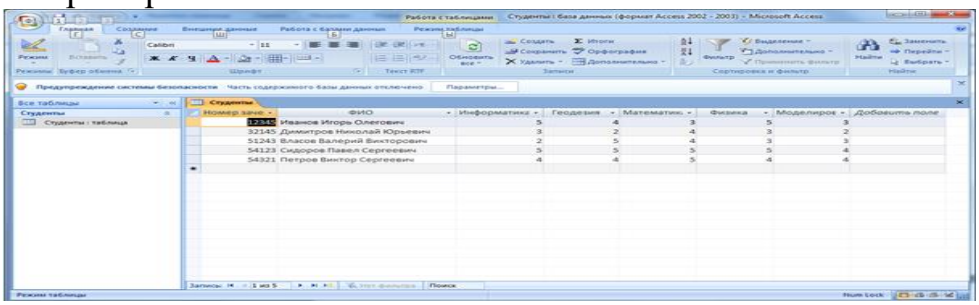


Рис.7. Пример заполнения таблицы

8) Сохранить и закрыть базу данных. Переместить файл базы данных в папку будущей программы.

9) Запустить C++ Builder. При запуске автоматически создается новый проект. Окно C++ Builder показано на рисунке 14. Для создания нового проекта, в случае если он не создался автоматически или вы его закрыли, выполнить команду меню File / New /Application.

10) Сохранить проект в свою рабочую папку, выполнив команду меню File / Save Project As. Будет сохранено несколько файлов проекта.

11) Расположить на форме требуемое количество объектов (см. рис.14).

Вкладка Standard: Label, Button, Edit.

Вкладка ADO: ADOConnection, ADOTable.

Вкладка DataAccess: DataSource.

Вкладка DataControls: DBGrid, DBNavigator.

12) Изменить подписи объектов Label и пользовательской формы Form1. Для этого необходимо у перечисленных объектов отредактировать свойство Caption в соответствии с рисунком 12.

13) У объектов Edit и ComboBox очистить поле свойства Text.

14) Поскольку объекты Edit используются только для вывода, то необходимо присвоить свойству ReadOnly для этих объектов значение true.

15) Настройка подключения к БД осуществляется за несколько шагов:

1. Выделить компонент ADOConnection1. Установить значение false для свойств Connected (соединение) и LoginPromt (вход с паролем) Сформировать строку подключения ConnectionString (строка параметров подключения к базе данных), нажав на кнопку с тремя точками.

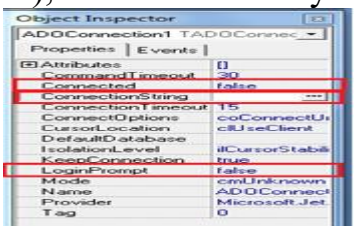


Рис.8. Настройка свойств объекта ADOConnection1

В появившемся окне выбрать пункт «Use Connection String» и нажать на кнопку Build»:

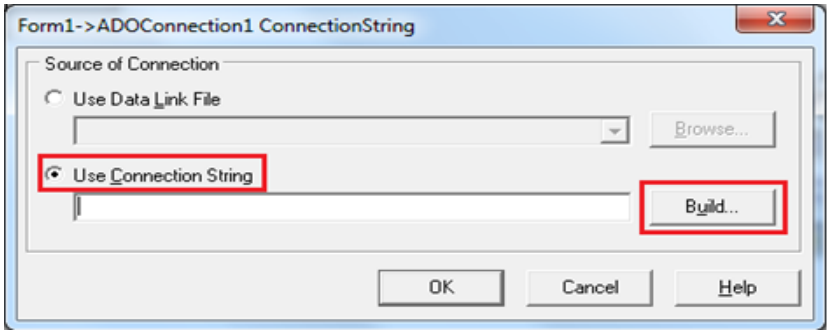


Рис.9. Окно настройки подключения

Далее необходимо выбрать поставщика данных и нажать на кнопку далее:

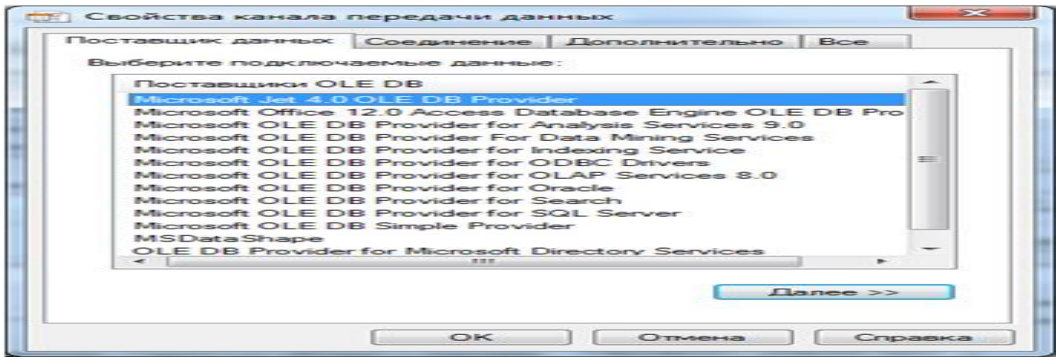


Рис.10. Выбор поставщика данных

Указать путь к базе данных и проверить соединение.

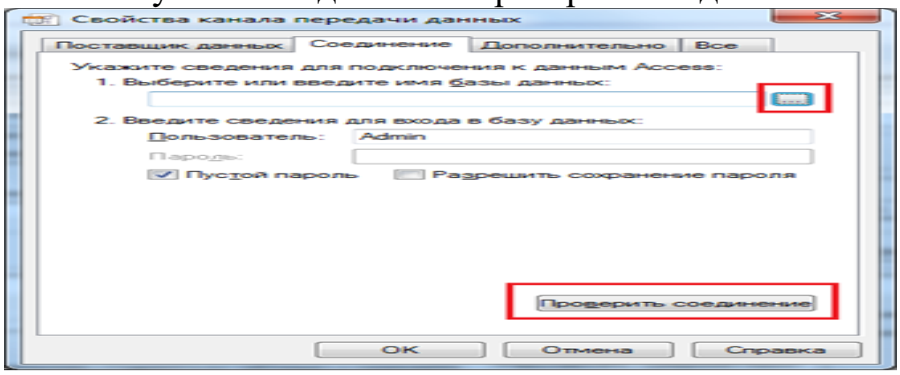


Рис.11. Выбор файла базы данных

Применить все изменения и поменять значение свойства Connected на true.

Важное примечание: Строка подключения представляет собой обычную строку, в которой перечислены параметры подключения программы к базе данных.

Пример строки подключения:

Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\ALL_WORK\ПРИКЛАДНАЯ ИНФОРМАТИКА\Прикладная информатика\Лабораторные работы\5-БД\Студенты.mdb;Persist Security Info=False

Как видно из примера, строка подключения содержит путь к базе данных, который при необходимости можно заменять программно. Это необходимо для подключения различных баз данных одного типа к программе.

2. Выделить объект ADOTable и в окне «Object Inspector» в поле Connection выбрать объект ADO Connection1, а в поле Table Name выбрать таблицу из базы данных. Если при выборе таблицы возникает ошибка, то соединение с базой данных не было установлено. В этом

случае следует проверить строку подключения в объекте ADOConnection1. В самую последнюю очередь установить переключатель Active в положение true.

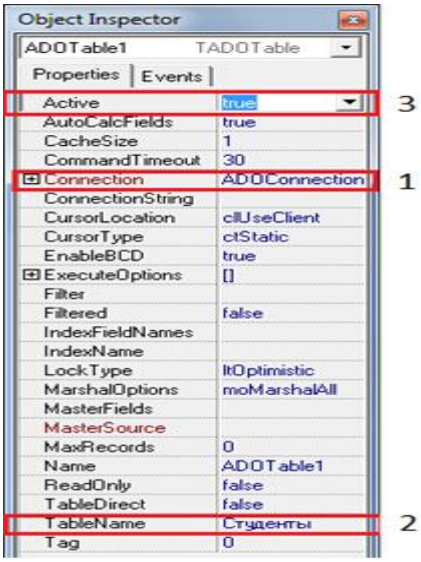


Рис.12. Настройка объекта ADOTable1

3. Выделить объект DataSource1 и в списке свойств в поле DataSet выбрать объект ADOTable1.

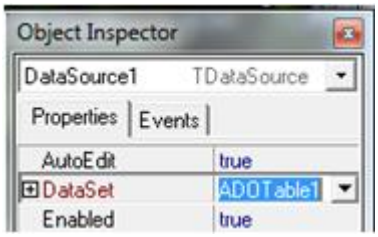


Рис.13. Настройка объекта DataSource1

4. Для объектов DBGrid и DBNavigator в поле свойства DataSource выбрать объект DataSource1:

1. При правильном выполнении всех вышеперечисленных операций в объект DBGrid должна быть загружена таблица из базы данных (рис.14).

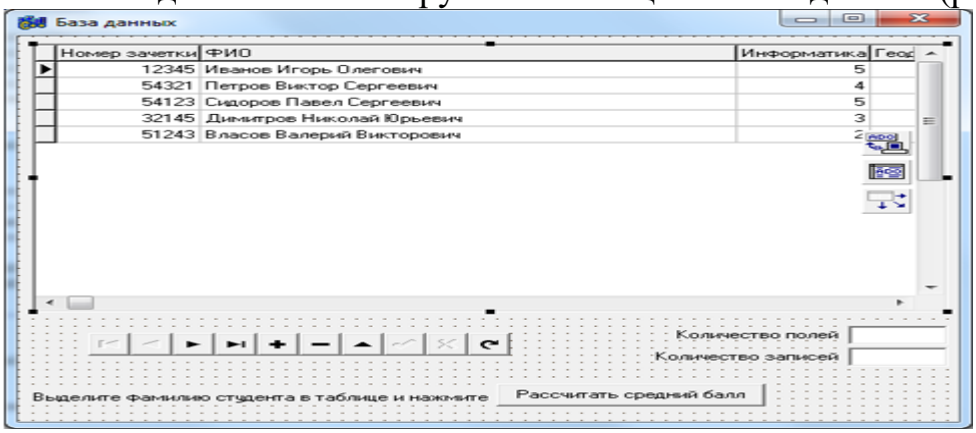


Рис.14. Загруженная таблица

Настройка подключения таблицы базы данных к программе завершена. Теперь необходимо написать код для расчета количества полей и записей таблицы для последующего их вывода в объекты Edit1 и Edit2.

6) Перерасчет количества полей и записей таблицы необходимо производить каждый раз при запуске программы, при добавлении или удалении записей, то есть при каждом изменении данных в таблице.

При изменении данных в таблице генерируется событие OnDataChange объекта DataSource. Для обработки этого события необходимо выделить объект DataSource1 и в списке событий Events дважды щелкнуть левой кнопкой мыши по полю OnDataChange. В созданной заготовке функции следует написать следующий программный код:

7) Подсчет среднего балла студента должен происходить по нажатию кнопки «Рассчитать средний балл». Для обработки нажатия этой кнопки дважды щелкнуть по ней и в заготовке функции написать следующий код:

8) Сохранить проект нажатием кнопки на панели инструментов.

9) Провести отладку и тестирование программы

10) Изучить назначение кнопок объекта DBNavigator1 самостоятельно.

11) Отредактировать таблицу средствами программы. Отредактировать уже существующие записи в таблице, добавить 4 и удалить 2 записи.

Задание 2. Дополнить программу «База данных». По желанию пользователя программа должна подсчитывать средний балл по выбранной дисциплине. Примерная компоновка формы измененной программы:

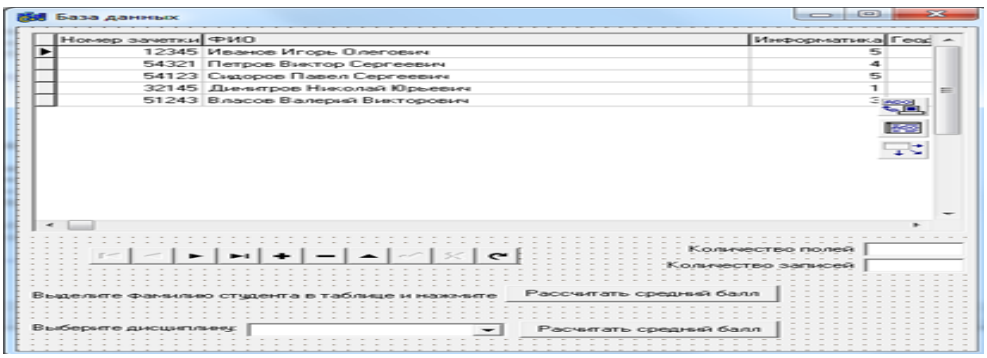


Рис.15.

Контрольные вопросы:

1. Что такое база данных?
2. Какие визуальные компоненты используются в приложении?
3. Какие вы знаете не визуальные компоненты?
4. Какие типы данных введены в программу?
5. Какие режимы работы с базой данных в системе C++Builder?

Литература

1. Информатика. Базовый курс. 2-е издание./ Под ред. Симонович С.В.- СПб.,Питер,2005-640с.ил. Учебник для вузов.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. Учебник для вузов.- СПб.,Питер,2003-464 с.ил
3. Kadirov M.M. Axborot texnologiyalari. O‘quv qo‘llanma, 1-qism. –T.: “Fan va texnologiya”, 2018. -316 b.
4. Kadirov M.M. Texnik tizimlarda axborot texnologiyalari. Darslik, 2-qism.
5. –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., Raxmanov Q.S. C va C++ tili. Darslik. –T.: “Voris”, 2013. -488 b.
12. Дейтел Х.М., Дейтел П. Дж. Как программировать на С++: Пятое малое издание. -М.:ООО «Бином- Пресс», 2007г.
13. Павловская Т.А. С/С++. Программирование на языке высокого уровня. -Спб. Питер, 2003.
14. Синицын А.К., Навроцкий А.А., и др.Программирование алгоритмов в среде builder c++. 1,2- часть. – Минск 2004
15. Шилд, Г. Программирование на *Borland C*++ / Г. Шилд. – Минск: ПОПУРРИ, 1999.
16. Специальная информатика: Учеб.пособ. / С.В. Симонович, Г.А.Евсеев, А.Г. Алексеев. - М.: АСТ ПРЕСС КНИГА, 2005.
17. Эллайн А. От ламера до программера. -СП6: Питер,2015

СОДЕРЖАНИЕ

Редактор Ахметжанова Г,М.