O'ZBEKISTON RESPUBLIKASI

OLIY VA O'RTA MAXSUS TA'LIM VAZIRLIGI

ALISHER NAVOIY NOMIDAGI

TOSHKENT DAVLAT O'ZBEK TILI VA ADABIYOTI

DASTURLASH ASOSLARI VA AXBOROT TEXNOLOGIYALARI

fani bo'yicha

O'QUV-USLUBIY MAJMUA

Toshkent – 2018

ASOSIY QISM

Fanning nazariy mashg'ulotlari mazmuni

1-modul. Axborot va axborot jarayonlari

Axborot. Axborot o‘lchov birliklari. Axborotni kodlashtirish. Axborot almashinuvi. Axborot texnologiyalari. Axborot texnologiyalarining rivojlanish bosqichlari. Shaxsiy kompyuterlar. Kompyuter tarmoqlari. Lokal va global tarmoq. Axborot texnologiyalarining apparatli vositalari. Axborot texnologiyalarining dasturiy vositalari. Amaliy dasturlar.

2-modul. Filologiyada axborot texnologiyalari

Matnni avtomatik o‘qish tizimlari. Matnni “tarjima qilish” tushunchasi. Avtomatik tarjima vositalari. Tarjima bosqichlari: Mark Tuliy Sitseron, V-XVII asrlarda tarjimashunoslik, XX asr o‘rtalarida tarjima, lug‘atlar va maxsus kompyuter dasturlari. Mashina vositasida tarjima qilish. Matnni interpretatsiya qilish. So‘zlarni morfologik tahlil qilish. Gaplarni sintaktik tahlil qilish mexanizmlari. Mashina tarjimasidagi avtomatik lug‘atlar. Tarjima qilinadigan matnning sintaktik mosligini aniqlash tizimlari. Matnni bir tildan boshqasiga o‘tkazish algoritmlari. Lingvistik va filologik resurslar. Sun’iy intellekt.

3-modul. Tarmoqlar

Kompyuter tarmoqlari. Kompyuter tarmoqlari klassifikatsiyalari. Kompyuter tarmoqlari ierarxiyasi. Kompyuter tarmoqlari topologiyasi. Serverlar. Kompyuter tarmoqlari dasturiy vositalari. Tarmoq texnologiyalari. Internet global tarmog‘i. Elektron pochta. IP-manzillar. Domen. Internet tarmog‘i bilan ishlash vositalari. Izlash tizimlari. Izlash tizimlari xarakteristikalari. Elektron kutubxona va internet resurslar.

4-modul. Dasturlash tillari

Dasturlash tillarining umumiy xarakteristikasi. Dasturlash tillarida ma’lumotlarni ifodalashning vositalari. Ma’lumotlar va ularni kodlar shaklida ifodalash. Sanoq tizimlari. Pozitsion va pozitsion bo‘lmagan tizimlar. Operativ xotira va registr. Bir sanoq sistemasidan boshqasiga o‘tkazish.

Algoritm xususiyatlari. Algoritmni ifodalash usullari. Operatorli algoritmlar. Blok-sxema. Oddiy tuzilmali algoritmlarni ishlab chiqish. Oddiy va murakkab tiplar. Tiplarni o‘zaro o‘tkazish. Chiziqli dasturlar. Shart operatorlari. Tarmoq operatorlari.

Algoritm va abstrakt mashinalar. Funksiyalar. Massivlar. Satrlar. Fayllar ustida amallar. Rekursiv algoritmlar. Dinamik xotira va unda axborotni saqlash. Algoritmning murakkabligi. Algoritmlarni baholash. Massivlarda izlash va tartiblash. O‘rin almashtirishlar. «Ochko‘z» algoritmlar. Graflarda algoritmlar. Leksik va sintaktik tahlil. Matnlar ustida algoritmlar.

5-modul. Dasturlash texnologiyalarining zamonaviy yo‘nalishlari

Yuqori darajali dasturlash tillari va ularning asosiy xarakteristikalari. Protseduraga asoslangan dasturlash. Ob’ektga yo‘naltirilgan dasturlash. Integrallashgan dasturlash. Zamonaviy informatsion tizimlarda dasturlash. Modulli dasturlarni yaratish. Ob’ektli tiplar. Inkapsulyatsiya. Vorislash. Polimorfizm. Klasslar va ob’ektlar. Konstruktorlar va destruktorlar. Oynali operatsion tizimlarda dasturlash. Menyu. Dasturni sozlash va testdan o‘tkazish. Vizual dasturlash asoslari.

Amaliy mashg‘ulotlarni tashkil etish bo‘yicha ko‘rsatma va tavsiyalar

Amaliy mashg‘ulotlarning maqsadi nazariy olingan bilimlar asosida amaliy topshiriqlarni bajara olish ko‘nikmalarini hosil qilishdan iborat. Amaliy mashg‘ulotlarda tinglovchilar tarjimonlikda mavjud kompyuterli dasturiy ta’minotlar (ABBYY Lingvo, Multitran, Polyglossum, Promt) dan amaliyotda samarali foydalanish usullari hamda ulardan dars jarayonida foydalanishni o‘rganadilar.

Amaliy mashg’ulotlarning taxminiy mavzulari ro‘yxati

1. Windows OS. Masalalar panelini sozlash. Obyektlarni ochish, yig’ish, yopish, oyna o‘lchamlari va joylarini o‘zgartirish. Ish stolida fayl, papka va yorliqlar yaratish.
2. Zip va Rar formatli arxivlangan fayllarni hosil qilish va ularni arxivdan chiqarish.
3. Kompyuterda viruslar va mavjud antivirus dasturlaini ishlatish.
4. Matn muharriri yordamida hujjatlar yaratish va tahrirlash. Matn kiritish, tahrirlash va xotirada saqlash. Matn qismlarini ajratish usullari.
5. Matn muharririda bufer yordamida matnning ko‘chirish, nusxalash va o‘chirish. Matnni formatlash.
6. Matn muharririda obyektlar yordamida matnni turli ko‘rinishlarda yozish. Jadval yaratish va ularda o‘zgartirishlar qilish.
7. Matematik formulalar yozish. Snoska, kolontitul va gipermurojaatlar.
8. Taqdimot dasturida slayd, fon, shablon va animasiyalar. Tovush effektlari. Prezentasiyada slaydlar almashinishini qo‘lda va avtomatik tarzda tashkil qilish.
9. Elektron jadvalda ma’lumotlar bilan ishlash. Ma’lumotlarni formatlash. Funksiya va diagrammalar ustasi bilan ishlash. Tayyorlangan hujjatni bosmaga chiqarish. Mutaxassislikka oid masalalarni bajarish.

10. Berilganlar bazasini hosil qilish, saqlash, tahrirlash, so‘rovlar, shakllar, hisobotlar va sahifalar tashkil qilish.
11. Internet. Qidiruv tizimidan foydalanib zarur adabiyotlarni topish va olish.
12. Elektron pochta. EP ro‘yxatidan o‘tish. Xat yozish, jo‘natish va kelgan xatlarni o‘qish.
13. HTML tili. Sarlavha, sahifaga jadval, rasm va murojaatlar qo‘yish. Harakatlanuvchi obyektlar hosil qilish. Shrift ranglari va sahifa fonlari. Har xil atributlar. Sahifani ochib ko‘rish va uni tahrirlash. Sayt tushunchasi. Internetda saytni joylashtirish.
14. Tarjimon dasturlari bilan ishlash.
15. Ifodalar va operatorlar.
16. Arifmetik va mantiqiy operatorlar.
17. Tiplarni boshqa tipga o‘tkazish.
18. Konsoldan kiritish va chiqarish operatorlari.
19. Shart operatorlari. "if" va "if-else", "switch-case" shart operatori
20. Sikl operatori. For, Foreach, Do-While operatori.
21. Massivlar bilan ishlash.
22. Qism dasturlar.
23. Ma’lumotlarning abstrakt tiplari va inkapsulyasiya tuzilishi
24. Obyektga yo‘naltirilgan dasturlash.
25. Xatoliklar bilan ishlash.
26. Satrlar ustida amallar.
27. Fayllar ustida amallar.

Laboratoriya mashg‘ulotlarini tashkil etish bo‘yicha ko‘rsatma va tavsiyalar

Labaratoriya mashg‘ulotlaridan maqsad talabalar ma’lum mavzular bo‘yicha olgan nazariy va amaliy bilimlarini dasturlash orqali mustahkamlaydilar. Bunda talabalar mashg‘ulotlarda misol va masalalarni yechishda, yechimlarni tahlil qilishda qo‘llay olishlari nazarda tutiladi.

Laboratoriya mashg’ulotlarning taxminiy mavzulari ro‘yxati

1. MS Word muhitida hujjat yaratish.
2. MS Excel dasturida elektron jadvallar bilan ishlash.
3. MS Access muhitida ma’lumotlar bazasini yaratish
4. MS Power Point muhitida taqdimot yaratish
5. Matnni avtomatik o‘qish tizimlarini sozlash va foydalanish.
6. Avtomatik tarjima vositalari bilan ishlash.
7. Kompyuter tarmoqlaridan foydalanish.

8. Elektron pochta bilan ishlash.
9. Kompyuterda viruslar va mavjud antivirus dasturlarini ishlatish.
10. Izlash tizimlaridan foydalanish.
11. Ma’lumotlarni himoyalash.
12. Web sahifa yaratish.
13. Oddiy va murakkab tiplar. Tiplarni o‘zaro o‘tkazish.
14. Shart operatorlari. Tarmoq operatorlari. Sikl operatorlari.
15. Funksiyalarni yaratish va qo‘llash.
16. Massivlar ustida amallar.
17. Satrlar bilan ishlash.
18. Fayllar ustida amallar.
19. Ob’yektga yo‘naltirilgan dasturlash.
20. Formali dastur ishlab chiqish.

Kurs ishini tashkil etish

Fan bo‘yicha kurs ishi namunaviy o‘quv rejada ko‘zda tutilmagan.

Mustaqil ishlarni tashkil etish shakli va mazmuni

Mustaqil ishning maqsadi olingan nazariy bilimlarni mustahkamlash, belgilangan mavzular asosida qo‘shimcha bilim olishdan iborat. Bunda ushbu ishlarni bajaradilar:

- amaliy mashg‘ulotlarga tayyorgarlik;
- nazariy tayyorgarlik ko‘rish;
- uy vazifalarni bajarish;
- o‘tilgan materiallar mavzularini qaytarish;
- mustaqil ish uchun mo‘ljallangan nazariy bilim mavzularini o‘zlashtirish.

Bunda talabalar ma‘ruzalarda olgan bilimlarini amaliy mashg‘ulotlarni bajarishlari bilan mustahkamlashi hamda statistikaning ba‘zi mavzularini tushunishi hamda ularga oid masalalarni yechishlari kerak.

Mustaqil ish mavzularini o‘zlashtirish ta‘lim jarayonida uzluksiz nazorat qilib boriladi va yozma hisobot sifatida topshiriladi.

Tavsiya etiladigan mustaqil ta’lim mavzulari

1. Kompyuter tarmog’i servislari.
2. Rasm fayllarini tahrirlovchi dasturlar.
3. Ovoz fayllarini tahrirlovchi dasturlar.
4. Video fayllarini tahrirlovchi dasturlar.

5. Respublikamizda davlat organlari tomonidan taqdim etilayotgan interaktiv xizmatlar va ularning ahamiyati.
6. Elektron ta’lim tizimlari.
7. Ijtimoiy tarmoqlar.
8. Yuqori bosqichli algoritmik tillar

Dasturning informatsion-uslubiy ta‘minoti

Fanni o‘qitish jarayonida mavjud nashr qilingan o‘quv qo‘llanmalari va elektron manbalar, Internet tizimidagi mos ta‘lim saytlari ma‘lumotlaridan http://www.zionet.uz va shunga o‘xshash saytlaridan foydalaniladi. Kompyuter texnikasini qo‘llash bilan bog‘liq zamonaviy pedagogik va informasion texnologiyalar asoslangan o‘qitish metodlari qo‘llaniladi.

Foydalaniladigan adabiyotlar ro‘yxati

Asosiy adabiyotlar

1. Svetin Yakov, Veselin Kolev & Co. Fundamentals of computer programming with C#. Sofia, 2013.
2. Endryu Troelsen. S# i platforma .NET. – SPb.: Piter, 2012g. - 796 s.
3. M.Aripov, M.Fayziyeva, S.Dottayev. Web texnologiyalar. O‘quv qo‘llanma. T.; “Faylasuflar jamiyati”. 2013. 350 bet.
4. M.Aripov, A.Madraximov. Informatika, informatsion texnologiyalar. Darslik. T: TDYuI., 2004.
5. D.Watson, H.Williams Computer Science. 2014 y. ISBN 978 1471809309, eISBN 978 1471809323

Qo‘shimcha adabiyotlar

1. **E**ndryu Troelsen. S# i platforma .NET. – SPb.: Piter, 2012g. - 796 s.:
2. Tom Archer. Osnovы S#. – M.: Izd.-torgovыy dom «Russkaya redaksiya», 2010 g.
3. Labor V. V. Si Sharp: Sozdanie prilojeniy dlya Windows/ V. V. Labor.— Mn.: Xarvest, 2013.
4. Usmonov A.I., Baxramov F.D. Kompyuter texnologiyalari asoslari. O‘quv qo‘llanma -T: “O‘zdavJTU”, 2010.
5. Aripov M.M., Kabiljanova F.A., Yuldashev Z.X. «Informatsionnыe Texnologii». Elektronnoe posobie dlya studentov VUZov (CD), Tashkent 2008, NUUz
6. M.Aripov, B.Begalov, U.Begimqulov, M.Mamarajabov. Axborot texnologiyalar. O‘quv qo‘llanma. T.: “Noshir”, 2009.
7. Yu. A. Alyaev, O.A. Kozlov. Algoritmizasiya i yazыki programmirovaniya – M: Finansы i statistika, 2007.

8. Tom Archer. Osnovы S#. – M.: Izd.-torgovыy dom «Russkaya redaksiya», 2010 g.

Internet saytlari

1. http://www.ctc.msiu.ru/materials/Book1,2/index1.html
2. http://www.ctc.msiu.ru/materials/CS_Book/A5_book.tgz

1-Mavzu. Dasturlash tillari konsepsiyasi

Reja

1. Dasturlash tillari tushunchalarini o‘rganish sabablari
2. Dasturlash domenlari
3. Til o‘rganish qoidalari
4. Dasturlash tillari evolyusiyasi
5. Sintaksis va semantika tavsifi

1. Dasturlash tillari tushunchalarini o‘rganish sabablari

Tabiiy xolki studentlar dasturlash tilini o‘rganishning asosiy qoidalaridan qanday foyda olishga ajablanishadi. Shunga qaramasdan ko‘plab kompyuter texnikasiga oid bilimlar jiddiy o‘rganishga munosib. Quyida dasturlash tilini o‘rganish koidalaridan foydalilarini keltiramiz.

- ***Tushunish qobiliyatining o‘sishi.***

Bu turlicha tushuniladiga biror narsani jiddiy o‘ylash kobiliyatida odamlar tilning ma’no kuchi orkali o‘ylaganlari ta’sirida ularni aloqaga kirgizadi. Bular tilni sayoz tushunish bu tilning qismlarini tushunishni chegaralaydi, odatda muxim mavxumlikni xam boshkacha qilib aytganda odamlar og‘zaki yoki yozma nutqda tuzilishlarni fikrlay olishmaydi va tasvirlab berisha olmaydi.

Dasturchilar fikrlari asosan dastur jarayonini rivojlantirishga qaratiladi. Dasturiy joylar chegaralarini rivojlantirish qaysi til nazorat tuzilmalari turlari, ma’lumotlar tuzilmalari, ma’lumotlardan foydalanish, ular qurish mumkin algoritmlarini shakllari xam shunday cheklangan . Dasturlash tili xususiyatlarini turli anglash, bunday cheklovlar dasturiy ta’minot ishlab chiqishni kamaytiradi. Dasturchilar til tuzulmalarini o‘rganish orkali dasturiy ta’minot ishlab chikish diapazonini oshirishadi. Buni ilgari surish mumkinki, boshka til imkoniyatlarini o‘rganish dasturchiga til imkoniyatidan foydalanishga xalaqit beradi. Chunki boshka tillar bir birini ko‘llab kuvvatlamaydi.Boshqacha kilib aytganda dasturlash tili asosiy koidalarini o‘rganish til xususiyatlari uchun qimmatli baxo beradi, xattoki til xususiyatlari va qurulmalari bir biriga to‘g‘ri bo‘lmasa xam dasturchilar foydalana oladi.

- ***To‘g‘ri til tanlay olish***

Ko‘plab moxir dasturchilar kompyuter ilmiga oid bilimlari yetarli bo‘lmagan, shunga karamasdan ular o‘zlarining dasturlash kobiliyatlarini mustakil o‘stirishgan. Bunday o‘kuv dasturlari ko‘pincha bu dastur loyixasiga oid bir yoki ikki tilni chegaralaydi. Ko‘plab boshka dasturchilar to‘g‘ri shug‘ullanish usullarini qabul qilishgan. Ular o‘rgangan tildan foydalanishmagan va ko‘plab xususiyatlari hozirda

ushbu tilni o‘rganishda kimmatli emas. Natijada ko‘plab dasturchilarga yangi loyixa berilsa ular sayoz tildan foydalanishadi. Agar ular ko‘plab tillarni bilishsa loyixaga eng ko‘p kaysi til mos kelishini bilishadi. Bir tilning ba’zi xususiyatlari boshka tilga mavxumligicha qoladi.

- ***Yangi tillar o‘rganish kobiliyatining oshishi***

Kompyuter dasturlari xali yosh va uslubiy dizayn, dasturiy ta’minot ishlab chiqish vositalari uzluksiz rivojlanmokda. Bu dasturiy ta’minotni ishlab chiqish uzluksiz o‘rganishga muxtoj. Yangi dasturlash tilini o‘rganish jarayoni davomiy va qiyin bo‘lishi mumkin, ayniksa o‘zida bir yoki ikki tillarni tajriba qilmagan dasturchilarga tilning asosiy koidalarini yaxshi tushunish, bu qanday koidalar tarkibiga kirishini ko‘rish oson bo‘ladi. Misol uchun dasturchi obektga yo‘naltirilgan dastur tilini bilsa Javani oson o‘rgana oladi. Xozirda ko‘plab dasturlash tillaridan keng foydalanilmokda. So‘nggi ma’lumotlarga qaraganda dasturlash tillari tez o‘zgarmokda. Bu shuni ko‘rsatadiki dasturchilar bu tillarni o‘rganishga tayyor bo‘lishlari kerak. Oxir okibatda dasturchilar tilini va koidalarini bilishi orqali dasturni oson tushuna oladi.

- ***Bajarilayotgan ish moxiyatini yaxshi tushunish***

Dasturlash tili koidalarini o‘rganish bu jarayon natijasini ko‘rish uchun xam kerak xam qizikarli. Jarayonni tushunish nega bu til aynan shunday yaratilganligini tushunishga olib boradi. Bu esa tildan samarali foydalanishga yordam beradi.

Ayrim dastur nuksonlari o‘sha dastur xaqida ma’lumotga ega dasturchi tomonidan aniqlanadi. Amalga oshirish masalalarini tushunishning yana bir afzalligi, bizga tasavvur qilish imkonini beradi. Ba’zi xollarda, ba’zi bir
amalga oshirish masalalari ilmi bir dastur uchun tanlangan bo‘lishi mumkin, muqobil tuzilmalar nisbiy samaradorligi hakida maslaxatlar beradi. Misol uchun amalga oshirish murakkabligi haqida kam biladigan dasturchilar kichik dasturlarda samarasiz dastur yaratishadi.

- ***Bilgan tildan yaxshi foydalanish***

Ko‘plab mavjud dasturlash tillari katta va murakkab. Dasturchi dastur tili va xususiyatlarini bilishi ulardan foydalanishi uchun qulay. Bu qoidalarni o‘rganish orkali dasturchilar o‘zlari uchun oldindan noma’lum bo‘lgan til xususiyatlaridan foydalanishni boshlashadi.

- ***Ilg‘or xisoblash qadami***

Nixoyat, dasturlash tili tushunchalarini o‘rganish hisoblash texnikasida asosiy rol o‘ynaydi. Shunga qaramasdan nega aynan bir til mashxur bo‘lganini aniqlash mumkin. Lekin eng mashxur tillar xar doim eng yaxshi emas. Bunga sabab ba’zi hollarda dasturchilarning ko‘p tillarni bilmagani sababli o‘sha til qisman bo‘lsada foydalaniladi. Misol uchun ko‘p odamlar ALGOL 60 ni Fortran 2004 ga qaraganda yaxshi deb bilishadi. Biroq ko‘plab kompyuter foydalanuvchilari o‘zlari ishlatayotgan

til foydasini bilishmaydi. Umuman olganda keltirilgan tillardan to‘g‘risini tanlash oxir oqibat yomonini muomiladan chiqaradi

2 Dasturlash domenlari

Kompyuter turli yo‘llar uchun foydalaniladi, yadroviy qurollar va telefon o‘yinlari uchun foydalaniladi. Kompyuter foydalanishda turli xil dasturlash tillarida rivojlantirish uchun ishlatiladi. Bu bo‘limda biz kompyuterning bazi dasturlash tillarini o‘rganamiz.

2.1 Ilmiy dasturlar

Birinchi raqamli kompyuterlar 1940-1950 yildan kashf qilindi va ilmiy dasturlar uchun foydalana boshlandi. Odatda ilmiy dasturlar ma’lumotlarni bog‘lash uchun foydalaniladi, ammo matematik hisoblashlarni talab qiladi.

2.2 Biznes dasturlari

1950 yildan boshlab kompyuterlar biznes dasturlari uchun foydalanib boshlangan. Asosiy kompyuterlar faqat bir dasturlash tilida shu dastur uchun rivojlantirilgan. Birinchi muaffaqiyatga erishgan til COBOL (ISO/IEC,2002), buning dastlabki versiyasi 1960 yilda ishlab chiqilgan. Biznes dasturlari ichida u haliyam eng ko‘p foydalaniladi. Biznes dasturlari harakterlanadi. Qo‘shimcha imkoniyatlari puxta ishlab chiqilgani uchun u o‘nli sonlar va harflar bilan tasvirlanadi, o‘nlik matematik hisoblashlar uning harakterini belgilab beradi.

COBOL (ISO/IEC,2002) ning biznes dasturlash sohasida rivojlanishi uchun u yerda bazi sabablar bo‘lgan. Shu sababli bu kitobda COBOL (ISO/IEC,2002) haqida bazi ma’lumotlar berilgan.

2.3 Sun’iy intellekt

Simvollik yoki nomerik hisoblashlar sun’iy intellekt (SI)da kompyuter dasturlarini umumiy maydonlaridan foydalaniladi. Simvollik hisoblashlar asosan simvollardan, nomerikda nomerlardan foydalaniladi. Shuningdek, to‘plamlarga nisbatan simvollik hisoblashlar ma’lumotlarga bog‘lanishda ko‘proq qulayliklar yaratadi. Boshqa dasturlarga qaraganda bu turdagi dasturlar moslashuvchanlikni ko‘proq talab qiladi. Misol uchun, baza SI axborotlari kod bo‘limlarini bajarish davomida yaratishga va bajarishga qulaylik yaratadi.

SI dasturlari uchun birinchi keng foydalanilgan til LISP (Mc Carthy et al., 1965) u 1959 yilda ishlab chiqilgan. LISP tomonidan SI dasturlari 1990 yil davomida

rivojlangan. Ammo 1970 yil boshida Prolog (Clocksin va Mellish, 2003) mantiqiy dasturlaridan foydalanishga to‘g‘ri kelgan. Anchadan keyin SI tizim tillari sifatida yaratilgan, masalan C. Scheme (Dybvig 2003), LIPS shevasi, mos ravishda 15 va 16 bo‘limlarda Prolog tanishtiriladi.

2.4 Dasturlash tizimlari

Kompyuter tizimida operatsion tizim va dasturlar yig‘indisi dasturlash tizimlari deyiladi. Dasturlash tizimlar uzliksiz samarali foydalaniladi. Bundan tashqari, u tashqi qurilmalar uchun dasturiy ta’minot interfayslarini yozish imkonini ham beradi.

1960-1970 yillarda ba’zi kompyuter ishlab chiqaruvchilar misol uchun IBM, Digital va Burroughts (hozirgi UNISYS), maxsus dasturiy tizimlar uchun rivojlandirilgan dasturiy til. IBM kompyuterlari uchun PL\S tili, PL\I uchun maxsus, Digital uchun va BLISS uchun, hozigi yuqori darajadagi til; Burroughts uchun ALGOLni ifodalaydi. Ammo hozirgi paytda ko‘plab dasturlar S va S++ tillarida yaratiladi.

UNIX dasturiy tizimlari faqat S tilida yozilgan (ISO, 1999), u portga oson bog‘lanadi, yoki harakatlanadi. S ning ayrim xususiyatlari dasturlash uchun yaxshi tanlov. Yaxshi dasturchilar cheklovlarga ishonishmaydi. Ba’zi nopdatiy dasturchilar S tili juda xavfli deb hisoblashadi

2.5 Web dasturlar

World Wide Web elektronik tillar tomonidan qo‘llab quvvatlanadi, misol uchun HTML bu dasturlash tili emas, Java umumiy dasturlash tili. Chunki dinamik internet keng tarqalgan, ba’zi hisoblashlarni texnologiya o‘z ichiga oladi. Bu xususiyatlar HTML dasturlash tili tomonidan taqdim etiladi. Dasturlash tili ko‘p kodlardan ifodat, masalan JavaScript va PHP. Bazi tillar hujjatlarni nazorat qilish uchun konstruksiyalarni o‘z ichiga oladi, bular 1.5 va 2 bo‘limda yoritib beriladi.

3 Til o‘rganish qoidalari

Yuqorida aytib o‘tilganidek bu kitobning asosiy maqsadi dasturlash tilining asosiy tushunchalari va imkoniyatlarini tadbiq etishdan ifodat. Bundan tashqari bu qismlarni dasturiy ta’minot jarayoni o‘sishiga ta’sirini e’tiborga olgan holda o‘rganamiz. Buni bajarish uchun bizga qoidalar to‘plami kerak bo‘ladi. Bunga o‘xshgagan qoidalar ancha munozarali, chunki berilgan til xususiyatlari boshqalariga aloqador ekanligiga qo‘shiluvchi hattoki 2 ta komp’yuter olimini topish mushkul. Shunga qaramasdan ko‘pchilik quyidagi qoidalar muhimligini tan oladi.quyidagi jadvalda berilgan 4 ta qoidadan 3 tasi juda muhim (4-chisi qolgan 3 tasiga o‘zaro bo‘g‘liq) bu jadvaldagi har bir bo‘limga katta e’tibor bilan qarash kerak.

Xususiyatlar	O‘qishga qulaylik	Yozishga qulaylik	Ishonuvchanlik
Soddalik	•	•	•

Imlo qoidadari	●	●	●
Ma’lumot turlari	●	●	●
Model sintaksisi	●	●	●
Mavhumlikni ochish		●	●
Ifodalash		●	●
Tekshirish turlari			●
Istisno holatlardan qoydalanish			●
Cheklovlar			●

3.1 O‘qishga qulaylik

Dastur uning qanchalik o‘qishga osonligi va tushunuvchanligi bilan baholanadi. 1970-yildan oldin dasturiy ta’minot tashkilotlari yozuvni ifodalashdagi kodlarga katta ahamiyat berishgan. Dasturlash tilidagi boshlang‘ich ijobiy xususiyat samarali bo‘lgan. Kompyuter qurilmalari undan foydalanuvchilarga qaraganda ko‘proq loyihalashtirildi. 1970-yiga kelib dasturiy ta’minot tushunchalari rivojlantirildi. Kodlash juda kichik rolni o‘ynab qoldi. Xizmat ko‘rsatish asosiy qism bo‘lib qoldi. Chunki xizmat ko‘rsatishning yengillanishi dasturni o‘qiy olishda muhim qism bo‘ldi. O‘qishga osonlik dastur va dastur tilining eng muhim qismi hisoblanadi. Dasturlash tilining rivojida eng muhim ulashdir. Bunda mashina yo‘nalishida va insoniyat yo‘nalishida alohida o‘zgarish bo‘ldi. Dasturning muammoli sohasi e’tiborga olinishi kerak. Misol uchun agar dastur hisoblashni ifodalasa ishlatishga mo‘ljallanmagan tilda yoziladi va bu notabiiy va murakkab bo‘lib, o‘qishga qiyinchilik qiladi. Quyidagi xususiyatlar ham dastur o‘qilishida o‘z hissasini qo‘shadi.

3.1.1 Umumiy soddalik

Umumiy soddalik dastur o‘qilishida muhim ta’sir ko‘rsatadi. Qurilmalari ko‘p bo‘lgan tilni qurilmalari kam bo‘lgan tilga qaraganda o‘rganish qiyin. Dasturchilar katta tilni va tillar guruhini o‘rganishi kerak va uning qismlarini e’tiborsiz qoldirmasligi kerak. Bu o‘rganish namunasi katta til qurulmalarini tushuntirish uchun foydalaniladi. Lekin bu argument mavjud emas.

Tildagi 2-muhim xususiyatlardan biri asosiy miqdordir. Bu ma’lum operatsiyani bir necha yo‘llar bilan muvaffaqiyatli bajarishdir. Misol uchun javada sonlarni 4 xil usul bilan qo‘shishi mumkin.

Son=son+1

Son+=1

Son++

++son

Endi 3-muhim muammo bu operatorning birdan ko‘p ma’nolari bo‘lishi. Lekin bu ko‘pincha foydali agar dasturchilarga ham operatorga yangi ma’no qo‘shish imkoniyati berilgan bo‘lsa. Soddalik tilni uzoqqa olib bora oladi. Ko‘plab guruh tillar model va mazmuni soda ko‘rinishga keltiriladi. Shunday soddalik guruh til dasturlari tushunishga ko‘p vaqt talab qilmaydi. Chunki ular boshqaruv tizimida murakkablikni kamaytiradi. Dastur qurulmasida kam aniqmaslik qoladi. Chunki operatorlar juda sodada.

3.1.2 Orthogonal tushunchasi

Dasturlash tilida nazoratni va ma’lumot tuzilmalarini qo‘lga olish uchun nisbatan kichik sodda qurulmaga ega bo‘lgan operatorlardan foydalaniladi. Bundan tashqari har bir birikma to‘g‘ri va mazmunga ega. Ortogonal tilining ma’nosi dasturda ko‘rinishi mustaqil.(ortogonal so‘zi matematikadan kelgan so‘z ya’ni vektorlarning perpendikulyarligi). Ortogonallik bazaga aloqadordir. Ortoganalning yetishmasligi tildagi qoidalarga istisnolik olib keladi. Biz ortogonallikdan foydalanishni dizayn loyihasi sifatida IBM kompyuterlari va VAX minikompyuterlaridagi tillar bilan solishtirish orqali bezak bera olamiz. Buni bir oddiy misol sifatida ko‘raylik. 2 ta 32 bitlik butun sonni qo‘shish uchun xotirada 2 sonning biri hosil bo‘lgan yig‘indi bilan almashtiriladi. IBMda bu 2 xil usulda amalga oshiriladi.

A Reg1, memory_cell

AR Reg1, Reg2

Reg1 ←contents(Reg1) + contents(memory_cell)

Reg1 ←contents(Reg1) + contents(Reg2)

32 bitli butun qiymat uchun VAX qo‘shimcha instruksiya bo‘ladi.

ADDL operand_1, operand_2

operand_2 ← contents(operand_1) + contents(operand_2)

VAXdagi dizayn qo‘llanmalar ortogonaldir. Chunki 1 ko‘rsatma ob’ekt sifatida xotiradan yoki jurnaldan foydalanish mumkin. Bu yerda ob’ektni aniq o‘rnatish uchun 2 xil yo‘l bor ya’ni barcha yo‘llar bilan birlashtirilishi mumkin. Lekin IBM loyihasi ortogonal emas. Chunki biz 2 sonni qo‘shib natijani xotirada saqlay olmaymiz. Bundan tashqari IBM loyihasi juda qiyin va o‘rganish ham noqulay.

Ortogonallik oddiylikka juda yaqin. Tilda ortogonallikning ko‘pligi so‘rovlarda juda kam istisno holatlarga olib keladi. ma’lumki kam istisnolar til darajasini yuqori holatga ko‘taradi. Bu o‘rganish va tushunish osonligini bildiradi. Misol uchun S tilida ortogonallik kam. Garchi S tilida 2 xil ko‘rinishda qurulgan ma’lumot turlari bor. Massiv va satr (satr funksiyadan qaytarilishi mumkin lekin massiv emas). Strukturaning har bir a’zosi bo‘sh va o‘ziga tegishli tipdan tashqari ma’lumot a’zosi bo‘lishi mumkin. Massivdagi element bo‘sh va funksiyadan tashqari ixtiyoriy ma’lumot turi bo‘lishi mumkin.

Ko‘p ortogonallik ham muammolar olib keladi. Eng ko‘p ortogonallangan til bu ALGOL68. Har bir til ALGOL68 ni tip sifatida quradi va bu yerda hech qanaqa cheklovlar yo‘q. Bundan tashqari ko‘plab qurulmalar qiymat chiqaradi. Ba’zan ortogonalning ortiqcha formalari kerakmas va murakkablikni olib keladi. Chunki til orqaga qaytib ishlashini va ortogonalning yuqori daraja natijalarini so‘raydi.

3.1.3 Ma’lumot turlari

Ma’lumot turlarini va ma’lumot tuzulmalarini aniqlash uchun yetarli imkoniyat mavjudligi o‘qiy olish uchun yana bir yordam hisoblanadi. Misol uchun aytaylik raqamlangan ma’lumot ko‘rsatchik uchun ishlatiladi, chunki tilda mantiqiy tipi yo‘q. Shunga o‘xshagan tillarda bizda quyidagicha faqat 1 topshiriq bo‘lishi mumkin.

timeOut = 1

Bu yerda bu holat ma’nosi aniq emas. Agar tilda mantiqiy tipi bo‘lsa biz quyidagiga ega bo‘lamiz.

Timeout=true

Bu esa ancha tushunarlidir.

3.1.4 Dizayn sintaksisi

Dastur o‘qilishida tildagi elementlar sintaksis va forma muhim ta’sir ko‘rsatadi. Sintaksis loyihalar:

- ***Muhim so‘zlar.*** Dastur ko‘rinishi va dastur o‘qilishida tildagi muhim ya’ni kalit so‘zlar kuchli ta’sir ko‘rsatadi. Ayniqsa eng muhumi qurulma nazoratida shakl berish usuli turadi. Ba’zi tillar guruhlarini tuzish uchun maxsus so‘z yoki belgilar juftlaridan foydalangan. S va uning avlodlari murakkab holatlarni belgilash uchun qo‘shtirnoqdan foydalanilgan. Har bir til qiynaladi, chunki sinf holatlari 1 xil yo‘l bilan tugatiladi va bu esa qaysi guruh qachon tugatilganini aniqlashga qiyinchilik tug‘diradi. Fortran95 va Ada tillarida bu nisbatan

osonroq chunki har bir guruh yopilishida alohida sintaksis belgidan foydalaniladi.

Boshqa bir muhim narsa bu dasturda o‘zgaruvchi sifatida til maxsus kalit so‘zlaridan foydalanish imkoniyati mavjudmi yoki yo‘q. Bunda dastur natijasi adashtiruvchi bo‘lishi mumkin.

- ***Forma va ma’no***. Dasturni loyihalashtirish (dizaynlash) maqsadi ularning ko‘rinishida o‘qish osonligiga yordam berishdan ifodat. Bunda sitaksis ma’nosi dasturga to‘g‘ridan to‘g‘ri amal qilishi kerak. Ba’zi hollarda bu qoidalar o‘xshash yoki bir xil til qurulmalari tomonidan buziladi. Bunda ko‘rinish bir xil bo‘lishi mumkin lekin ma’no jihatdan farq qiladi.

  Asosiy shikoyatlardan biri bu UNIXning qobiq buyruqlari ko‘rinishlari o‘zlarining dastur funksiyalariga mos kelmaydi.

3.2 Yozilish osonligi

Osonlik bilan yozish bu tanlangan dastur loyihasi uchun qaysi til munosib ekanligini tanlashdir. Ko‘plab til xususiyatlari o‘qilishga va yozilishga ham ta’sir ko‘rsatadi. Bu dasturchiga yozilgan dastur qismini qayta o‘qish imkonini beradi. Dasturda o‘qilishi bilan birga dasturga oid muammoni yozishda til imkoniyatini o‘ylash kerak. Ma’lum dastur uchun 2 til yozilish usullarini solishtirish oqilona ish emas.

Quyida biz yozilish osonligiga nimalar ta’sir ko‘rsatadi bilib olamiz.

3.2.1 Ortogonallik va oddiylik

Agar tilda juda ham ko‘p qurulmalar bo‘lsa, dasturchilar u til bilan yaqin bo‘lmaydilar. Bu vaziyat ba’zi xususiyatlarni noto‘g‘ri ishlatish va ba’zilarini qo‘llamaslik ko‘proq nafis bo‘lishi yoki ko‘proq natijali bo‘lishi mumkin. Ba’zan esa tasodifan bilmagan xususiyatlarimizdan foydalanish ajoyib natija olib kelishi mumkin. Dasturchi muayyan murakkab muammo uchun o‘zining loyiha rejasini sodda qurulmalarni o‘rgangandan keyin javob topishi mumkin. Boshqacha qilib aytganda juda ko‘p ortogonallik dasturning yozilishiga zarar keltirishi mumkin.

3.2.2 Mavhumlikni anglash

Qisqacha qilib aytganda mavhumlik bu aniqlash qobiliyati va undan foydalanish dasturdagi murakkab strukturalar va detallar e’tiborsiz qoldiriladi. Mavhumlik dasturlash tili loyihasida zamonaviy qoida kaliti hisoblanadi.

1.4. Dasturlash tillari evolyusiyasi
Susning Plankalkul tili
Psevdokod

IBM 704 kompyuteri va Fortran tili
Funksional dasturlash: LISP tili
ALGOL 60 tilini mukammallashtirishga birinchi qadam
SOBOL tilida savdo sotiq yozuvlarini kompyuterlashtirish
BASIC tilining dastlabki vaqtdagi bosqichlari
PL /1 tili ([1] 68-71-betlar)

Bu mavzu ostida dasturlash tillari kolleksiyasining rivojlanishi ko‘rib chiqiladi. Bunda har bir tilning yaratilgan muhiti o‘rganiladi va tilning imkoniyatlari va rivojlanishi uchun asoslariga e’tibor qaratiladi. Tilning umumiy ta’riflari keltirilmagan, buning o‘rniga, har bir tilning ba’zi yangi o‘ziga hos hususiyatlarini ko‘rib chiqilgan. Kelgusidagi dasturlash tillariga yoki kompyuter ilmiga ta’sir qilgan hususiyatlarga alohida e’tibor qaratilgan.

Bu mavzuda hech bir dasturlash tilining hususiyatlari va konsepsiyalari chuqur muhokama qilinmaydi; bu keyingi mavzularga qoldiriladi. Bu mavzuda hamma uchun ham birdek tushunarli bo‘lmagan dasturlash tillari va ularning konsepsiyalarining xilma-xilligi haqida gap boradi. Bu mavzuni toki kitobni o‘rganib bo‘lmaguncha takroran o‘qish mumkin.

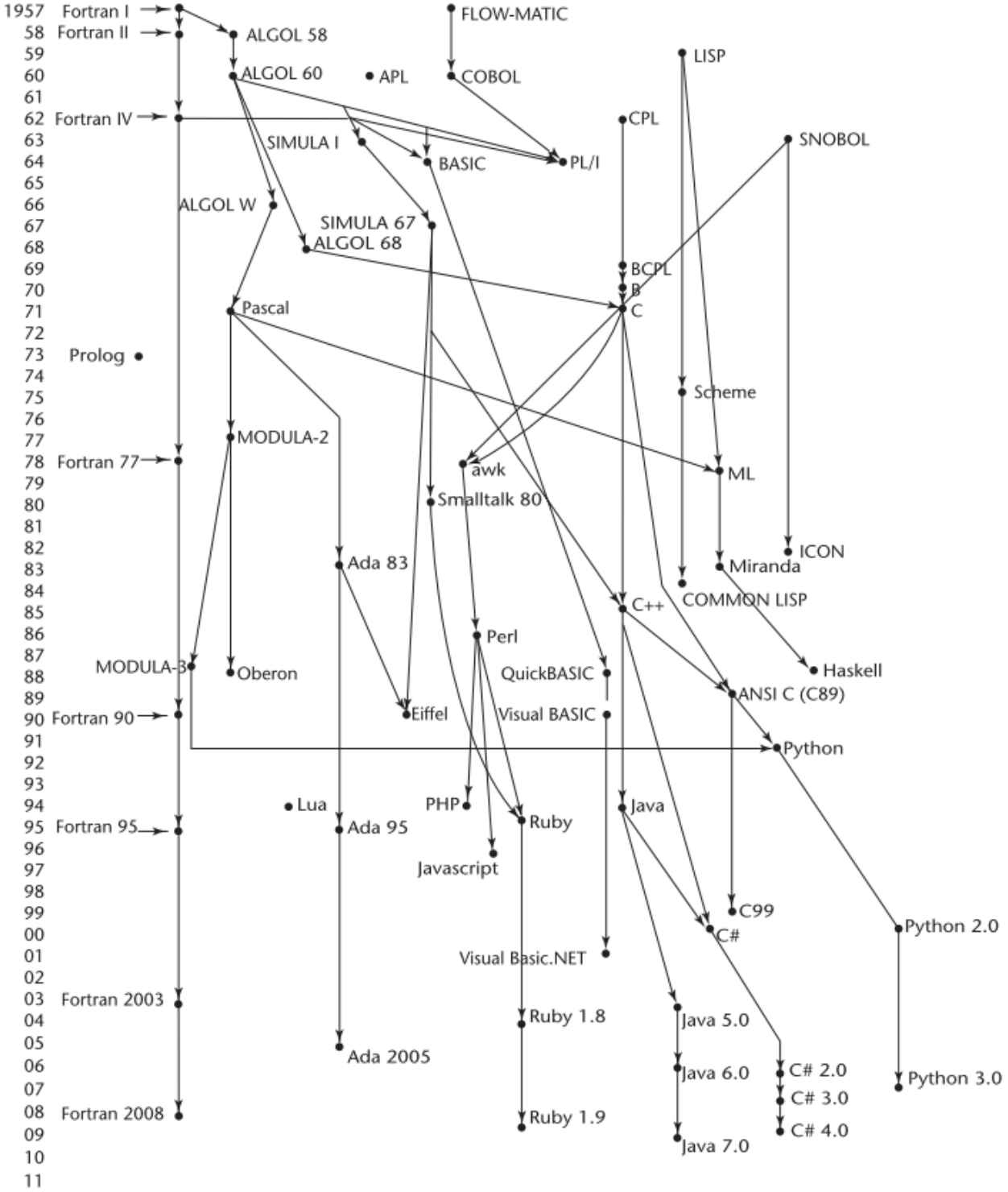
Bu yerda muhokama qilinadigan dasturlash tillari ham sub’ektiv tanlab olingan va ayrimlar uchun sevimli bo‘lgan tillar keltirilmagan bo‘lishi ham mumkin. Bu yerda dasturlash tillarining va kompyuter olamining rivojlanishiga katta hissa qo‘shgan deb hisoblangan tillar tanlab olingan.

Bu mavzuning tashkillashtirilishi quyidagicha: tillarning dastlabki versiyalari hronologik tartibda muhokama qilinadi. Biroq, keyingi versiyalar keyingi qismlarda emas, balki, dastlabki versiyalari bilan bir vaqtda ko‘rib chiqiladi. Masalan, Fortran 2003 tili Fortran I (1956 yildagi) versiyasi bilan birgalikda ko‘rib chiqiladi. Shu bilan birga ayrim joylarda, ikkinchi darajali deb hisoblangan tillar ham agar boshqa tilga aloqador bo‘lsa shu mavzuda ko‘rib chiqiladi.

Bu mavzu har biri turli tillarda yozilgan 14 ta namuna dasturlar keltirilgan. Ushbu mavzudagi dasturlar to‘liqligicha tushuntirib berilmagan, faqatgina dasturlash tillaridagi ko‘rinishini tasvirlab berilgan. Umumiy buyruqlar sintaksisga ega bo‘lgan tillar bilan tanish bo‘lgan dasturchilar bu dasturlarning kodlarini oson o‘qiydilar va tushuna oladilar, faqat bundan LISP, COBOL va Smalltalk tillarida yozilgan kodlar mustasno (LISPda keltirilgan namunaga o‘xshash sxema funksiya 15- mavzuda muhokama qilinadi). Shu masalaning o‘zi Fortran, ALGOL 60, PL/I, BASIC, Pascal, C,

Perl, Ada , Java, JavaScript va C# dasturlarida ham tuzilgan. E'tibor bering bu mavzudagi zamonaviy dasturlash tillarining ko'pchiligi dinamik massivlar bilan ishlash imkoniyatiga ega, biroq namunaviy masalalarning osonligi sababli, bu yerda dinamik massivlardan foydalanilmagan.

Quyidagi sxemada bu mavzuda muhokama qilingan yuqori darajali dasturlash tillarining kelib chiqishi ko'rsatilgan.



Susning “ Plankalkul ” tili

Bu mavzuda gaplashadigan birinchi dasturlash tili g‘ayritabiiy bo‘lib tuyulishi mumkin. Birinchidan, u hech qachon ishlatilmagan. Undan tashqari, 1945- yilda ishlab chiqilganiga qaramay, u haqidagi ma’lumotlar 1972-yilgacha chop etilmagan. Chunki juda ozchilik bu til bilan tanish bo‘lgan, uning ko‘pchilik qobiliyatlari 15 yilgacha boshqa tillarda namoyon bo‘lmagan.

Tarixi

1936-1945 yillar oralig‘ida, nemis olimi Konrad Sus elektromexanik relelardan kompleks va murakkab kompyuterlarni seriyasini qurgan. 1945 yilning boshida, Ittifoqdoshlarining bombardimoni tufayli uning oxirgi 34 modelidan boshqa hamma modeli yo‘q bo‘lib ketgan. Shundan so‘ng u chekka Hintershteyndagi Bavariya qishlog‘iga ko‘chib ketadi, uning ilmiy tadqiqot jamoasi esa tarqab ketadi.

Sus yolg‘iz ishlab boshlaganidan keyin, u 34 modelida hisoblashni bajaradigan ifodalarni kiritish uchun til yasashga kirishib ketadi. Bu uning 1943-yilda boshlagan uning doktorlik ishi loyihasi edi. U bu tilni “Plankalkul” deb nomladi, ya’ni hisoblash dasturi. Qo‘l yozmalar 1945 yilga borib taqalsa ham, Sus uni 1972 yilda e’lon qilgan va turli tuman masalalarni yechish algoritmlarini shu tilda yozgan.

Umumiy tushunchalar

Plankalkul nihoyatda to‘liq, ayniqsa ma’lumotlar strukturasi sohasidagi eng yangi o‘ziga hosliklarga ega edi. Plankalkuldagi eng sodda ma’lumot tipi – bit edi. Butun va haqiqiy sonlar tiplari bit tipidan qurilgan.

Odatiy skalyar tiplardan tashqari, Plankalkulda massivlar va rekordlar (S ga asoslangan tillarda strukt deb ataladi) ham bor. Rekordlar o‘z ichiga ichki rekordlarni ham olishi mumkin.

Bu til aniq bir **“goto”** operatoriga ega bo‘lmasa ham, Ada tilidagi **for** operatoriga o‘xshash iteratsiya operatoriga ega. Undan tashqari bu tilda, siklning yakuniga yetganligi yoki yangi sikl ochilishi uchun **Fin** operatori ham bor. Plankalkulda tanlash operatori ham bor lekin **else** punktiga ega emas.

Sus dasturining eng qiziqarli hususiyatlaridan biri bu, matematik amallarni dasturdagi o‘zgaruvchilarning bir-biri bilan bog‘liqligini ko‘rsatgan holda biriktirib olishidadir

Glossariy

- **while –** цикл оператори томонидан тестлаштириш жараёни;
- **Идентификатор –** катта ва кичик лотин ҳарфлари, рақамлар ва таг чизиқ („_“) белгиларидан ташкил топган ва рақамдан бошланмайдиган белгилар кетма–кетлиги тушунилади;
- **Тил генератори –** тилнинг гапларини яралишига сабаб бўлувчи мосламадир;
- **Метатили -** бошқа тилни тасвирлаш учун ишлатиладиган тилдир;
- **If –** шарт оператори бўлиб, қандайдир шартни ростликка текшириш натижасига кўра программада тармоқланишни амалга оширади;
- **Старт рамзи -** маҳсус грамматика билан бошланувчи тил жумлалари қоидаларнинг кетма-кетлиги асосида бошқарилади
- **Синтактик тармоқ –** белгили семантикадаги тармоқ
- **Тилнинг синтаксис қоидаси –** тилдаги боғланишни белгилайдиган қисми
- **BNF (Backus-Naur формаси) –** бошқа тилни тасвирлаш учун ишлатиладиган метатил
- **Жумла –** терминал белгилардан иборат форма
- Цуснинг “ **Планкалкул** ” тили-1943-йилда Цус томонидан ишлаб чиқилган
- **Fin**- Ада тилидаги циклнинг якунига етганлиги ёки янги цикл очилишини ифодаловчи оператор
- **goto** - шартсиз ўтиш оператори
- **For** - такрорлаш оператори
- **Бўш операция** - дастурнинг умимий таркиби бузмасдан ўчирилган буйриқ ўрнига қўйиш мумкин.
- **Short code тили** - 1949 йили BINAC камютери учун ДЖонМочли томонидан яратилган
- **Speedcoding тизими** - IBM 701 компьютери учун Джон Бекус томонидан яратилган
- **IF** - оператори қандайдир шартни ростликка текшириш натижасига кўра программада тармоқланишни амалга оширади
- **DO** - **FORTRAN I** тилининг цикл оператори

- **ALLOCATABLE** - **FORTRAN** тилида ишлатилиб, массивлар сифатида эълон қилинган ва улар буйруқ бўйича динамик ҳолда хотирага жойлашади ва ўчади
- **CASE** - кўп вариантли тармоқланиш оператори
- **EXIT** - циклдан чиқиб кетиши оператори
- **CYCLE** - узатиш бошқаруви циклдан туриб унинг бошига ўтувчи бошқарув оператори
- **PRIVATE** - маълумотлар ва кичик дастурлар ҳақида маълумотларни ўз ичига олувчи ва ташқи алоқани бошқарувчи ёпиқ спецификатор
- **PUBLIC** - маълумотлар ва кичик дастурлар ҳақида маълумотларни ўз ичига олувчи ва ташқи алоқани бошқарувчи очиқ спецификатор
- **Scheme тили** - 1970- йилнинг ўрталарида MIT институтида ишлаб чиқилган
- **Компиляция** – яратилган дастур компьютер томонидан тестлаштириш жараёни
- **IAL тил** - **ALGOL** тилини тарихдаги эски номи
- **integer** – бутун сон типи
- **array** – массивни эълон қилиш
- **reading** – бутун типдаги ўзгарувчиларни ўқиш учун ишлатилади
- **begin** – дастурни бошланишини ифодалайди ва блок ўрнида ишлатилади
- **end** - дастур тугаганлигини англатади
- **printstring** - сатрни экранга чиқариш учун ишлатлади
- **printint** – бутун сон типдаги ўзгарувчиларни экранга чиқариш учун ишлатлади
- **COBOL тили** - 1- дастур тили бўлиб, АҚШ ҳукумати томонидан ишлаб чиқилган
- **BAL-FWD-FILE файл** - махсулот рўйхатига эга ва қайта буюртма талаб қилади
- **BASIC тили** - **COBOL** тилининг олдинги версияси ҳисобланиб, 1979-йилнинг охири 1980- йилнинг бошларида вужудга келган

Ушбу таянч ифодалар айнан шу мавзуда фойдаланилган. Қавслар ичида улар биринчи маротаба учрайдиган китобдаги шу мавзу бетлари келтирилган.

- **Назорат тузилмалари турлари**;
- **Маълумотлар тузилмалари;**
- **Маълумотлардан фойдаланиш;**
- **Тил тузулмалари;**
- **Java дастурлаш тили** – юқори савияли дастурлаш тили ҳисобланади;

- **ALGOL 60 дастурлаш тили** – мураккаб ҳисоблашлар кўп бўлган дастурлар тузишда қўлланилади;
- **Fortran дастурлаш тили** - универсал тиллардан ҳисобланади (1950-1958 , John Backus, IBM);
- **COBOL** – биринчи муаффақиятга эришган тил;
- **LISP** – СИ дастурлари учун биринчи кенг фойдаланилган тил;
- **UNIX** – операцион тизим;
- **Ортогонал** – математикадан келган сўз яъни векторларнинг перпендикулярлиги.

Амалий машғулот 1.
C# тили синтаксиси ва унинг лексик асоси

Ишдан мақсад: C# дастурлаш тили дастур тузиш кўникмалари билан танишиш. Содда масалалар алгоритмини блок-схема кўринишида тасвирлаш, Visual Studio 2013 дастурлаш мухитида C# тили орқали масаланинг дастурини тузиш.

Масаланинг қўйилиши: Тингловчи вариант бўйича берилган масалани алгоритмини блок-схема кўринишида тасвирлаши, C# дастурлаш тилида ишлаши ва керакли натижа олиши лозим.

Ишни бажариш учун намуна

Мисол: Берилган x, y ва z сонлари учун формуланинг натижаси топилсин.

Берилганлар: x=14.26, y=-1.22, $z=3.5\times10^{-2}$

$$t=\frac{2\cos\left(x-\frac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\frac{z^2}{3-z^2/5}\right)$$

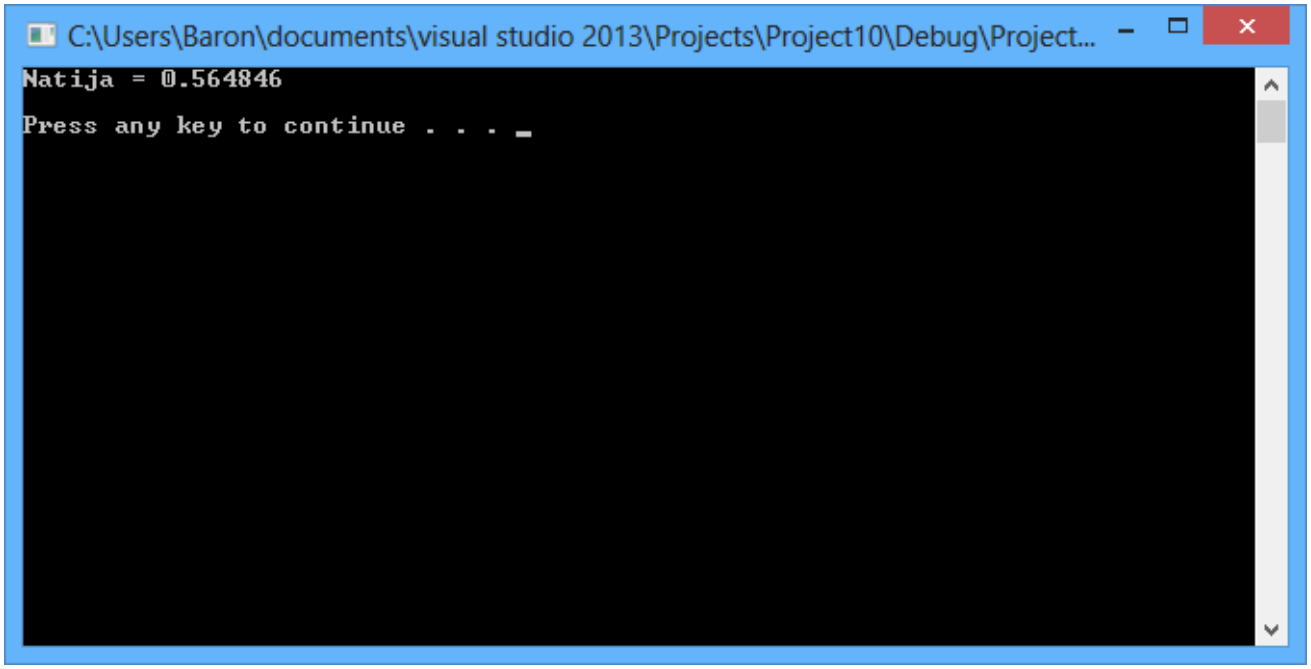
Натижа: t=0.564849

Дастур коди:

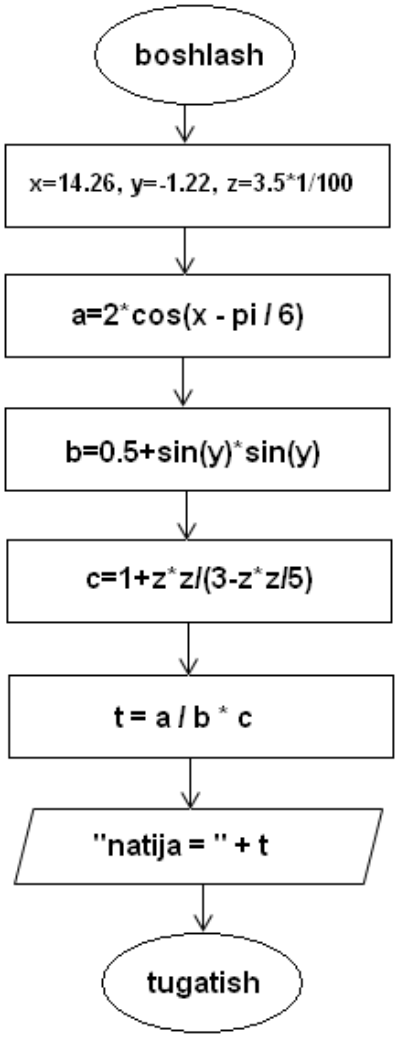
dastur.cs файли:

```
#define _USE_MATH_DEFINES
#include<iostream>
#include<cmath>
using namespace std;
int main(){
     double x,y,z,a,b,c,t;
     x=14.26;
     y=-1.22;
     z=3.5*1/100.;
     a=2*cos(x-M_PI/6);
     b=0.5+sin(y)*sin(y);
     c=1+z*z/(3-z*z/5);
     t=a/b*c;
     cout << "Natija = " << t << endl;
     system("pause");
     return 0;
     }
```

Дастур ишлаши натижаси:



Масала алгоритми блок схемаси:



Амалий топшириқлар

Берилган x, y ва z сонлари учун формуланинг натижаси топилсин.

1. x=14.26, y=-1.22, z=3.5×10^{-2},

$$t = \frac{2\cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y}\left(1 + \frac{z^2}{3 - z^2/5}\right)$$

Натижа: t=0.564849

2. x=-4.5, y=0.75x10^{-4}, z=0.845x10^{2},

$$u = \frac{\sqrt[3]{8 + |x-y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|}\left(tg^2 z + 1\right)^x.$$

Натижа: u=-55.6848.

3. x=-15.246, y=4.642x10^{-2}, z=20.001x10^{2},

$$\alpha = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \frac{y}{2}\right) + \sin^2 arctg(z).$$

Натижа: α=-182.036

4. x=0.1722, y=6.33, z=3.25x10^{-4} ,

$$\gamma = 5arctgx - \frac{1}{4}\arccos x \frac{x + 3|x-y| + x^2}{|x-y|z + x^2}.$$

Натижа: γ=-172.025

5. x=1.825x10^{2}, y=18.225, z=-3.298x10^{-2},

$$\psi = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right| + (y-x)\frac{\cos y - \dfrac{z}{(y-x)}}{1+(y-x)^2}.$$

Натижа: ψ =1.2131

6. x=6.251, y=0.827, z=25.001,

$$b = y^{\sqrt[3]{|x|}} + \cos^3 y \frac{|x-y|\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|} + x/2}.$$

Натижа: b=0.7121

7. x=17.421, y=10.365x10^{-3}, z=0.828x10^{5},

$$f = \frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|(\sin^2 z + tgz)}.$$

Натижа: f=0.33056

8. x=2.444, y=0.869x10^{-2}, z=-0.13x10^{3},

$$h = \frac{x^{y+1} + e^{y-1}}{1+x|y-tgz|}(1+|y-x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}.$$

Натижа: h=-0.49871

9. x=1, y=1, z=3

$$a = (1+y)\frac{x+y/(x^2+4)}{e^{-x-2}+1/(x^2+4)};$$

$$b = \frac{1+\cos(y-2)}{x^4/2+\sin^2 z}.$$

Натижа: a=9.608184; b=2.962605

10. x=3, y=4, z=5,

$$a = \frac{1+\sin^2(x+y)}{2+\left|x+2x/(1+x^2y^2)\right|} + x;$$

$$b = \cos^2(arctg\,\frac{1}{z}).$$

Натижа: a=3.288716; b=0.9615385

Amaliyot topshiriqlari

1. Konsolda ismingiz va familiyangizni chop qiluvchi dastur tuzing.

Тестлар

1. Биринчи дастурлаш тили қачон яратилган?

a) 1945 йил
b) 1972 йил
c) 1965 йил
d) 1970 йил

2. Қачондан дастурлаш тили ишлатила бошланган?

a) 1945 йил
b) 1972 йил
c) 1980 йил
d) 1945 йил

3. IPL тилиниг ривожланиши қачонгача давом етди?

a) 1961 йил
b) 1955 йил
c) 1960 йил
d) 1965 йил

4. Биринчи дастурлаш тили нима деб номланган?

a) Short Code
b) Pevdo Code
c) Plankalkul
d) Fortran

5. FORTRAN тилини тузишга сабабчи бўлган компьютер номи нима?

a) IBM 704
b) UNIX
c) AGAT
d) Linex

6. Рўйхатлар тузиш консепцияси ким томонидан ишлаб чиқилган?

a) Tsuze
b) Plankalkul
c) Allen Hyuel DJ.Show
d) Von Neiman

7. Қайси тил ҳеч қачон тадбиқ етилмаган?

a) IRL I
b) IRL II
c) IRL V
d) ALGOL

8. Теоремаларни механик исботлашда қайси тилдан фойдаланилади?

a) FLTR
b) FORTRAN
c) IRL
d) FORTRAN I

9. Қайси тилда икки хил структура бор?

a) LiSP
b) FORTRAN
c) ALGOL
d) IRL

10.Цюрихеда учрашувида қандай тил яратилган?

a) FORTRAN
b) IAL
c) ALGOL 58
d) LISP

11.FORTRAN тилининг давомчиси қайси?

a) ALGOL 58
b) IAL
c) LiSP
d) IRL

12.1959 йил ALGOL 58 тили қайси давлатларда кенг тарқалди?

a) Европада
b) АҚШда
c) Европада ва АҚШда
d) Россияда

13.Қайси тил SOBOL тилининг яратилишига сабаб бўлди?

a) FORTRAN
b) FLOW_MATIC
c) LiSP
d) ALGOL

14.Олдин тил турлари нечта категоряга ажратилган?

a) 3 та
b) 2 та
c) 5 та
d) 4 та

15.BASIC тили қачон яратилган?

a) 1977-1978 йиллар
b) 1979- 1980 йиллар
c) 1981-1982 йиллар
d) 1971-1978 йиллар

16.ВАРМАР нечта параметрнинг вазифаси була олади?

a) 2
b) 4
c) 1
d) 3

17.Бошкарувчилар кандай синволларга эга?

a) - , +
b) * , +
c) : , *
d) + , +

18.If операторига нечта шарт бериш мумкин?

a) 1
b) 2
c) 3
d) 4

19.while ва do цщиле фарки нимада?

a) Фарки ёк.
b) Иккаласи бирга ишлатилмайди
c) Иккаласи фа=ат бирга ишлатилади
d) Цикл биринчи текширалади до цикл эса шарт бажарилади.

20.goto операторини хамма жойда ишлатиш мумкинни?

a) Ха
b) йук
c) Хамма жойда ишлатиб булмайди.
d) goto оператори куп кулланилса хатоликка олиб келади.

21. Аксиоматик семантикада фойдаланиладиган мантикий иборалар нима деб аталади.

a) иборалар
b) тасдиклар
c) матинлар
d) сатрлар

22.Хулоса коидасининг энг юкори кисми нима деб аталади

a) утмиш
b) келажак
c) чакирув

d) келишув

23.Чизикдан тепадаги биринчи мантикий вазият кандай гапларни кўрсатади?

a) then
b) when
c) c) this
d) else

24.Аксиоматик семантикада нечта аник кулланиши бор?

a) 3 та
b) 2 та
c) 6 та
d) 4 та

25.Аксиоматик семантика нимага асосланган учун шундай номланган?

a) предметларга
b) сатрларга
c) математик мантикка
d) геометрик мантикка

26.Дастурлаш тилини ўрганиш қоидалари тўғри берилган жавобни топинг

a) Тушуниш қобилиятини ўсиши
b) Тўғри тилни танлай олиш
c) Янги тиллар ўрганиш қобилиятини ошиши
d) Барча жавоблар тўғри

27.Биринчи рақамли компьютерлар нечанчи йилда кашф қилинди?

a) 1890-1900 йилларда
b) 1940- 1950 йилларда
c) 1840-1850 йилларда
d) 1920-1930 йилларда

28.Нечанчи йилдан бошлаб компьютерлар бизнес дастурлари учун фойдалана бошланган?

a) 1940-йилдан
b) 1930-йилдан
c) 1950-йилдан

d) 1890-йилдан

29. Биринчи муаффақиятга эришган тил қайси?

a) C++
b) СИ
c) COBOL
d) LISP

30. СИ дастурлари учун биринчи кенг фойдаланилган тил қайси?

a) C++
b) СИ
c) COBOL
d) LISP

31. UNIX дастурий тизимлари қайси тилда ёзилган?

a) C++
b) СИ
c) Java
d) PHP

32. Ортогонал қандай тушунча?

a) Ортогонал сўзи математикадан келган сўз яъни векторларнинг перпендикулярлиги
b) Ортогонал сўзи математикадан келган сўз яъни векторларнинг паралеллиги
c) Ортогонал сўзи имло қоидалари деган маънони беради
d) Тўғри жавоб берилмаган

2-mavzu.O‘zgaruvchilar va tiplar

Reja

1. Kirish

2. Ma’lumotlarning elementar tiplari

3. Belgili satrlar

4. Foydalanuvchi tomonidan aniqlanuvchi ketma-ket tiplar

Ushbu mavzu boshlanishida ma’lumotlar tipi konsepsiyasi va ma’lumotlarning asosiy tiplari xossalari keltirilgan. Keyin sanaluvchi va cheklangan tiplar strukturasi ko‘rib chiqilgan. Bundan keyin ma’lumotlarning strukturali tiplari o‘rganib chiqilgan bo‘lib, asosiy e’tibor massivlar, yozuvlar va umumlashmalarga qaratilgan. Xulosada ko‘p turdagi va ssыlkali tiplar ko‘rib chiqilgan.

Tiplarning har bir kategoriyasi uchun ishlab chiqarish muammolari ko‘rib chiqilgan va kerakli tillar ishlab chiqaruvchilari tomonidan qabul qilinuvchi konstrukturaviy qarorlar tushuntirilgan. So‘ngra, ushbu tillarning strukturalariga baho berilgan.

Tiplar strukturasiga ularni amalga oshirish metodlari katta ta’sir ko‘rsatadi. Ushbu sababga ko‘ra ushbu mavzuga yana bir ahamiyatli bo‘lim qo‘shilgan bo‘lib, u ma’lumotlar, asosan massivlarni amalga oshirish muammolariga yo‘naltirilgandir.

Kirish

Ma’lumotlarni qayta ishlash orqali kompyuter natijalarga erishadi. Ushbu jarayonni bajarishning osonligi ma’lumotlar tipi ushbu masalaga qanchalik muvofiqligi bilan o‘lchanadi. Buning natijasida, mos tiplarning xilma-xilligi va ma’lumotlar strukturalarini qo‘llab-quvvatlash tilda oldindan ko‘rib chiqilishi juda muhim.

Ma’lumotlar tiplarining zamonaviy konsepsiyalari oxirgi 40 yillardan beri rivojlanib kelmoqda. Boshlang‘ich dasturlash tillarida aniq bir masalalarga mos keluvchi barcha ma’lumotlarning strukturalari ushbu tilda qo‘llaniluvchi ma’lumotlarning asosiy strukturalarining kichik miqdori bilan shakllantirilgan. Masalan, FORTRAN 90 tiligacha ishlab chiqilgan FORTRAN tili versiyalarida aloqa ro‘yxati va ikkilik daraxtlar, odatda, massivlar yordamida modellashtirilgan.

FORTRAN I tilida qo‘llanilgan modeldan tashqariga qo‘yilgan birinchi qadam COBOL tilidagi ma’lumotlar bazasi strukturasini ishlab chiqaruvchilari tomonidan

amalga oshirilgan bo‘lib, bu dasturchilarga o‘nlik sanoq sistemasidagi sonlar aniqligini o‘rnatish va axborot saqlovchi yozuvlarni taqdim etish uchun ma’lumotlarning strukturaviy tiplarini qo‘llash imkonini bergan. PL/I tilida butun sonli qiymatlar va haqiqiy sonlar aniqligini o‘rnatish imkoniyati yanada mukammallashtirildi. Keyinchalik, ushbu imkoniyatni taqdim etuvchi vositalar Ada va FORTRAN 90 tillariga kiritildi. Ilovalar sohasini kengaytirish maqsadida, PL/I tiliga turli tipdagi ma’lumotlar kiritildi. Fikrimizcha, dasturchiga asosiy tipdagi opreatorlar hamda ma’lumotlarni bergan yaxshi edi, bu esa foydalanuvchi tomonidan aniqlanuvchi ma’lumotlar tipini yaratishga imkon berardi, ularga struktura biriktirilar edi, xuddi bu ALGOL 68 tilida bajarilganidek. Bu ko‘rinib turganidek, ma’lumotlar tiplarini shakllantirish sohasidagi eng katta yutuqlardan biri hisoblanadi. Foydalanuvchi tomonidan aniqlanuvchi ma’lumotlar tiplarining afzalliklarini aytib o‘tamiz. Ushbu tiplar dasturlarning o‘qiluvchanligini oshiradi, chunki ularga ma’noli ismlarni qo‘llash mumkin. Foydalaluvchi tomonidan aniqlanuvchi tiplar o‘zgaruvchilar tiplarini tekshiradi. Foydalaluvchi tomonidan aniqlanuvchi tiplar mavjud bo‘lmasligi mumkin emas. Bundan tashqari, ushbu tiplar dastur modifikatsiyasini yaxshilaydi: dasturchi dasturdagi o‘zgaruvchilar kategoriyalarining tiplarini e’lon qilish operatorini o‘zgartirish orqali o‘zgartirishi mumkin.

1970-yillar oxirlarida paydo bo‘lgan ma’lumotlar tiplarini shakllantirish konsepsiyasi foydalanuvchi tomonidan aniqlanuvchi tiplar g‘oyalarini birlashtirish natijasida Ada 83 tiliga kiritilgan edi. Foydalaluvchi tomonidan aniqlanuvchi ma’lumotlar tiplari asosida yotuvchi metodologiyasi shundan iboratki, dasturchi o‘zgaruvchilarning har bir alohida sinflari uchun alohia tip yaratishi lozim va u vazifaning predmetli sohasida aniqlanadi. Bundan tashqari, vazifaning predmetli sohasidan o‘zgaruvchilar abstraksiyasi bo‘lgan tiplar unikalligini dasturlash tili ta’minlashi lozim. Bu yetarli darajada kuchli konsepsiya bo‘lib, dasturiy ta’minot ishlab chiqarilish jarayoniga katta ta’sir ko‘rsatadi. Yana bir qadam bosib Ada 83 tilida modellashtirilishi mumkin bo‘lgan ma’lumotlarning mavhum tirlariga o‘tamiz. Ma’lumotlarning mavhum tiplari asosida yotuvchi g‘oya tipni qo‘llashdan ushbu tipdagi o‘zgaruvchilarni e’lon qilish usuli hamda ular ustida bajariluvchi amallardan ajratishdan iborat. Yuqori bosqichli dasturlash tillarida ko‘rib chiqilgan ma’lumotlarning barcha tiplari mavhum hisoblanadi. Foydalanuvchi tomonidan aniqlanuvchi mavhum tipli ma’lumotlar 10 mavzuda ko‘rib chiqilgan.

Ma’lumotlarning eng ko‘p tarqalgan ikki strukturaviy (noskalyar) tiplari massivlar hamda qaydlar hisoblanadi. Ma’lumotlarning boshqa bir nechta turlari

kabi tip operatorlari yoki konstruktorlar orqali beriladi. Ulardan ushbu tipdagi o‘zgaruvchilarni yaratish uchun foydalaniladi. Tip operatorlariga misol qilib S tilida mavjud aylana hamda kvadrat qavslarni, yulduzchalarni keltirish mumkin. Yulduzchalardan massivlar, funksiyalar va ko‘rsatkichlarni berish uchun foydalaniladi.

Deskriptorlar terminlarida o‘zgaruvchilar haqida mavhum ham, aniq ham o‘ylash qulay. Deskriptor o‘zgaruvchilar atributlarining umumlashmasi bo‘lib, ular ushbu atributlarni saqlovchi xotira katakchalari to‘plami ko‘rinishida amalga oshiriladi. Agar barcha o‘zgaruvchilar statistik bo‘lsa, u holda deskriptorlar faqat kompilyatsiya vaqtidagina kerak. Statik deskriptorlar identifikatorlar jadvali bo‘lagi ko‘rinishida kompilyator orqali yaratiladi va kompilyatsiya davrida qo‘llaniladi. Dinamik atributlar yoki uning bo‘laklari, o‘z navbatida, dastur bajarilish paytida dinamik deskriptorga muhtoj. Bu holda deskriptordan dasturlar bajarilishini qo‘llab-quvvatlash tizimi foydalanadi. Statik deskriptorlar ham, dinamik deskriptorlar ham tiplarni tekshirish hamda xotirada o‘zgaruvchilarni joylash va o‘chirish amallarida foydalaniladi.

O‘zgaruvchi qiymati va u egallovchi xotira deganda, “ob’ekt” so‘zi tushuniladi. Ushbu kitobda “ob’ekt” deganda foydalanuvchi tomonidan aniqlanuvchi mavhum tipli ma’lumotlar nusxalari tushuniladi va biz undan o‘rnatilgan tiplar o‘zgaruvchilari qiymatlarini tavsiflashda foydalanmadik. Dasturlashning ob’ektga yo‘naltirilgan tillarida har qanday o‘rnatilgan yoki foydalanuvchi tomonidan aniqlangan sinf nusxasi ob’ekt deganidir. Bunday ob’ektlar 10 va 11 mavzularda to‘liq ko‘rib chiqiladi.

Keyingi bo‘limlarda keng tarqalib borayotgan ma’lumotlar tiplari ko‘rib chiqilgan. Ulardan ko‘pchiligi uchun ular bilan bog‘liq ishlab chiqarish muammolari shakllantiriladi. Ularning barchasi uchun kamida bitta misol keltirilgan. Ma’lumotlarning barcha tiplari uchun quyidagi muammo umumiydir: mavjud tipli o‘zgaruvchilar bilan qanday amallar ko‘rib chiqilgan va ular qanday beriladi?

Ma’lumotlarning elementar tiplari

Boshqa tiplar ta’riflarida aniqlanmaydigan ma’lumotlar tiplari ma’lumotlarning elementar tiplari deyiladi. Dasturlash tillarining deyarli barchasi ma’lumotlarning elementar tiplarining aniq to‘plamini ko‘rib chiqadi. Ushbu tiplaridan bir xillari

uskunaviy ta’minot xususiyatlarining akslantiruvlari hisoblanadi – masalan, butun sonlar. Boshqa tiplarni amalga oshirish ahamiyatsiz dasturiy to‘xtalishni talab qiladi.

Ma’lumotlarning strukturaviy tiplarini yaratish uchun tilda ma’lumotlarning asosiy tiplari bir yoki bir nechta konstruktorlari bilan birga qo‘llaniladi.

Sonli tiplar

Ko‘pchilik boshlang‘ich dasturlash tillarida faqat sonli elementar tiplar mavjud bo‘lgan. Zamonaviy tillarda ushbu tiplar oldingidek asosiy rolni o‘ynaydi.

Butun sonlar

Eng ko‘p tarqalgan elementar sonli tip butun son hisoblanadi. Hozirda ko‘p kompyuterlar butun sonlarning bir qancha o‘lchamlarini qo‘llaydi va ushbu imkoniyatlar dasturlashning ba’zi tillarida o‘z aksini topgan. Ada tili amalga oshirilganda, masalan, butun sonlarning uch o‘lchamliligiga yo‘l qo‘yiladi: SHORT INTEGER, INTEGER va LONG INTEGER. S kabi ba’zi dasturlash tillari butun sonlarning ishorasiz tiplariga ega va ushbu tiplar o‘zida ishorasiz butun sonlarni taqdim etadi.

Barcha butun sonlar bitlar satrlari ko‘rinishida kompyuterda beriladi, bunda bitlardan biri (qoidaga ko‘ra, chapdan oxirgisi) ishora taqdim etadi. Ma’lumotlarning butun tiplari uskunaviy ta’minot orqali ta’minlanadi.

Manfiy butun sonlar ishorali son yozuvi ko‘rinishida xotirada saqlanadi, bunda ishora biti son manfiyligini ko‘rsatib, qolgan bitlar sonlarning absolyut qiymatlarini taqdim etadi. Ishorali sonni yozish kompyuter arifmetikasida qo‘llanilmaydi. Ko‘pgina zamonaviy kompyuterlarda manfiy sonlarni saqlash uchun hisoblash va sanash uchun qulay bo‘lgan ikkilik sanoq sistemasida sonning qo‘shimcha kodi qo‘llaniladi. Manfiy butun sonning qo‘shimcha kodi ikkilik sanoq sistemasida mantiqan unga musbat son va bir qo‘shish orqali bajariladi. Bir qancha kompyuterlarda quyidagi kabi boshqacha usul qo‘llaniladi: ikkilik sanoq sistemasidagi sonning teskari kodi. Bunday yozishda butun sonning manfiy qiymati uning absolyut qiymatiga mantiqiy qo‘shilish hisoblanadi. Teskari aloqa ko‘rinishida taqdim etishning kamchiligi shundaki, u 0 sonini yozishning ikki usuli mavjudligidir. Agar sizni butun sonlarni taqdim etish muammosi qiziqtirsa, assembler tilidagi har qanday dasturlash kitobidan o‘qib olishingiz mumkin.

Haqiqiy (suzuvchi nuqtali) sonlar tiplari

Haqiqiy sonlar tiplari bunday sonlarning ko‘pchiligining taqdimoti faqat approksimatsiya bo‘lishiga qaramay haqiqiy sonlarni modellashtiradi. Masalan, π va ye (natural logarifm asosi) asosiy sonlaridan hech biri haqiqiy sonlar tiplari ko‘rinishida aniq taqdim etilishi mumkin emas. Ushbu sonlardan hech qaysisi yozuvning hech qanday aniq ko‘rinishida taqdim etila olmaydi. Ko‘pchilik kompyuterlarda haqiqiy sonlar ikkilik kodlarda saqlanadi, bu esa ularni yozish muammosini yanada murakkablashtiradi. Masalan, hattoki o‘nlik kattalik bo‘lgan 0.1 ni ham ikkilik sonlarning aniq bir to‘plami ko‘rinishida taqdim etib bo‘lmaydi. haqiqiy sonlarini qo‘llashning boshqa muammosi arifmetik amallar bajarilishida aniqlikni yo‘qotish hisoblanadi. haqiqiy sonlarni taqdim etish muammolari to‘g‘risida kitobdan (Knuth, 1981) to‘liq bilish mumkin.

Haqiqiy sonlar ilmiy yozuvdan olingan formada mantiss va bosqich ko‘rsatkiqlari kabi taqdim etiladi. Ilgari kompyuterlar haqiqiy kattaliklarni turlicha taqdim etgan, ammo hozirgi paytda ko‘pchilik mashinalar IEEE Floating-Point Standard 754 standarti bilan tavsiflangan formatni qo‘llaydi. Dasturlash tillarini amalga oshirish vositalarini ishlab chiqaruvchilar uskunaviy ta’minot orqali qo‘llaniluvchi har qanday taqdimotni qo‘llaydi. Dasturlash tillarining ko‘pchiligi haqiqiy sonlarning ikki tipini saqlaydi, float va double. float tipli o‘zgaruvchilar xotiraning to‘rt baytiga teng bo‘lgan standart o‘lchamga ega. double tipidan o‘lcham jihatdan katta mantissalar talab qilinuvchi sohalarda foydalaniladi. O‘zgaruvchilarning ushbu tiplariga tegishli yozuvlar, odatda, ikki marta katta xotirani egallaydi va mantissada minimum ikki marta ko‘p bitlarga ega.

Haqiqiy sonlar yordamida taqdim etish mumkin bo‘lgan ko‘pgina kattaliklar ularning aniqlishi hamda diapazoni orqali aniqlanadi. Son aniqligi – bu uning mantissasining aniqligi bo‘lib, u bitlar soni bilan o‘lchanadi, diapazon tushunchasiga esa mantissa o‘zgarishi diapazoni va bosqich ko‘rsatkichi o‘zgarish diapazoni kiradi.

Ba’zi kichik kompyuterlarning uskunaviy ta’minoti haqiqiy amallarni o‘zida saqlamaydi. Bunday mashinalarda ushbu amallar dasturiy ta’minot yordamida modellashtiriladi, bu esa 10-100 marta ularning bajarilishini sekinlashtiradi.

O‘nli sonlar

Ko‘pgina yirik kompyuterlar, kommersiya ilovalari uchun ishlab chiqarilgan bo‘lib, o‘nli sonlar tiplarini qo‘llovchi uskunaviy ta’minotga ega. Bunday tipli ma’lumotlarga chekli o‘nli sonlar va o‘nli nuqtadan iborat bo‘lib, ular o‘rnatilgan joyida bo‘ladi va butun qismdan kasr qismini ajratadi. Bunday tipli ma’lumotlar

kommersiya ilovalarida asosiy hisoblanadi, shuning uchun ular COBOL tilining asosini tashkil qiladi.

O‘nli sonlarning (haqiqiy tiplardan farqli ravishda) afzalliklari shundaki ular, o‘nli kattaliklarning aniq qiymatiga ega, bu albatta, chekli diapazonda bo‘ladi. Ularning kamchiliklari bosqich ko‘rsatkichi yo‘qligi sabab o‘zgaruvchilarning o‘zgarish diapazonining cheklanganligi va ularni xotirada taqdim etishda isrofgarchilikka yo‘l qo‘yilishi hisoblanadi.

O‘nli sonlar, belgilar satri kabi, xotirada o‘nli shifrlarning ikkilik kodlari orqali yoziladi. Bunday taqdimotlar ikkilik kodlangan o‘nli sonlar (BCD – binary-coded decimal). Ba’zi hollarda o‘nli kattaliklar 1 baytga bir shifr ko‘rinishida beriladi, boshqalarida esa bayt ikkita shifrga ega. Buni misolda ko‘rsatamiz. O‘nlik shifrni kodlashtirish eng kamida 4 bit talab qiladi. Natijada, kodlangan olti razryadli o‘nlik sonni saqlash uchun 24 bit xotira kerak bo‘ladi. Buni ikkilik ko‘rinishda saqlash atigi 20 bit talab qiladi. O‘nlik kattaliklar ustida amallar mashinalarning uskunaviy ta’minoti orqali amalga oshiriladi yoki dasturiy ta’minot orqali modellashtiriladi.

Mantiqiy tiplari

Mantiqiy tiplari tiplarning orasida eng oddiysi hisoblanadi. Ularning qiymati diapazoni faqat ikkita elementdan iborat bo‘lib, birinchisi rostlik ikkinchisi yolg‘onlik belgisidir. Birinchi bo‘lib ushbu tiplar ALGOL 60 tilida paydo bo‘ldi va 1960 yildan boshlab dasturlashning ko‘pgina universal tillariga qo‘shildi. Keng tarqalgan S tili bundan mustasnodir, bunda shart o‘rniga sonli ifodalar qo‘llanilishi mumkin. Bunday ifodalarda nol bo‘lmagan qiymatli barcha operandlar to‘g‘ri, nol esa yolg‘on qiymat hisoblanadi. S++ tilida mantiqiy tipi ko‘rib chiqilganiga qaramay, bu tilda mantiqiy tiplari o‘rniga sonli tiplarni qo‘llash ham mumkin.

Mantiqiy tiplari ko‘pincha o‘zgartirishlar yoki ishoralarni taqdim etish uchun qo‘llaniladi. Ushbu maqsadlar uchun boshqa tiplar ham ishlatilishi mumkinligiga qaramay, mantiqiy tiplari dasturning o‘qiluvchanligini oshiradi.

Mantiqiy qiymatlar yagona bit bo‘lib taqdim etilishi mumkin, ammo ko‘pgina mashinalarda xotiraning alohida bitiga samarali qarashning iloji bo‘lmagani uchun, ushbu qiymatlar xotiraning minimal katakchasida saqlanadi.

Belgili tiplar

Belgili ma’lumotlar kompyuterlarda raqamli kodlash orqali saqlanadi. Kodlashning eng ko‘p tarqalgan tizimi ASCII (American Standard Code for Information Interchange – Axborot almashinuvi uchun Amerika standart kodi) bo‘lib, bunda 128 turli belgilarni kodlash uchun 0-127 qiymatli diapazon qo‘llaniladi. Alohida belgilar kodlarini qayta ishlashni ta’minlovchi vositalar sifatida ko‘pchilik dasturlash tillari ular uchun alohida asosiy tip ko‘rib chiqadi.

Kommersiya dunyoviylashuvi va kompyuterlar orasidagi aloqalarning zaruriyligi natijasida, butun dunyo bo‘ylab ASCII belgilar to‘plami tezroq keraksiz bo‘lib bormoqda. Yaqinda alternativ 16 bitli belgilar to‘plami ishlab chiqildi, u Unicode nomini oldi. Ushbu to‘plamda dunyoning ko‘pgina tabiiy tillarining belgilari kiritilgan. Masalan, unda kirilcha belgilar va taycha raqamlar mavjud. Unicode belgilar to‘plamini birinchi bo‘lib qo‘llagan dasturlash tili Java tili hisoblanadi, ammo ushbu to‘plam albatta boshqa keng tarqalgan tillarda ham qo‘llanila boshlaydi.

Belgili satrlar

Belgili satrlar belgilar ketma-ketligi hisoblanadi. O‘zgarmas belgili satrlar natijalar chiqarilishini o‘zi bilan birga olib keladi, barcha ma’lumot tiplarini kiritish va chiqarish ko‘pincha satrlar orqali bajariladi. Albatta, belgili satrlar barcha dasturlarda asosiy rol o‘ynaydi.

Ishlab chiqarish muammolari

Quyida belgili satrlar bilan bog‘liq ishlab chiqarishning ikkita asosiy muammosi keltirilgan.

Satrlar oddiygina belgilardan iborat bo‘lgan massivlar xilma-xilligi yoki elementar tip (massivga xos indeksatsiyasiz) bo‘lishi mumkinmi?

Satrlar statistik yoki dinamik uzunlikka ega bo‘lishi kerakmi?

Satrlar va ular ustida amallar

Agar satrlar elementar tip kabi aniqlanmagan bo‘lsa, satrli ma’lumotlar, odatda, alohida belgilardan iborat massivlarda saqlanadi va ularga massiv elementi bo‘libgina ssьilkalanish mumkin. Bunday yondashuv Pascal, C, C++ va Ada dasturlash tillarida qabul qilingan. Pascal tilida satrlar asosiy tip bo‘lmaydi va packed atributiga ega char massivlariga qiymatlar berish mumkin va ular munosabatlar operatorlari orqali o‘zaro taqqoslanishi mumkin.

Ada tilida STRING tipi o‘rnatilgan hisoblanadi va bir o‘lchamli elementlar massivini aniqlaydi, ular CHARACTER tipiga tegishli. Quyi satrni chaqirish butun sonli diapazon orqali beriladi, ular aylana qavsga olingan hamda kerakli belgilar joylashuvini ko‘rsatadi. Masalan,

NAME1 (2:4)

buyrug‘i ikkinchi, uchinchi va to‘rtinchi satr belgiidan iborat quyi satrni beradi.

Ada tilida belgilar satri konkatenatsiyasi operatsiya bo‘lib, & belgii bilan beriladi. Keyingi operator NAME2 o‘zgaruvchisini NAME1 o‘zgaruvchisi qiymati bo‘lgan satrning o‘ng oxiriga oxiriga qo‘yishni bajaradi:

NAME1 := NAME1 & NAME2;

Masalan, agar NAME1 o‘zgaruvchisi “PEACE” satriga ega bo‘lib, NAME2 o‘zgaruvchisi “FUL” satriga ega bo‘lsa, qabul qilish operatsiyasidan so‘ng NAME1 o‘zgaruvchisi “PEACEFUL” satriga ega bo‘ladi.

S va S++ tillarida belgilar satrini saqlash uchun char massivlaridan foydalaniladi; satrlar bilan amallar to‘plami esa string.h sarlavhali faylli odatiy kutubxonada ko‘rib chiqilgan. Satrlar bilan bajariluvchi amallar va kutubxona funksiyalarining ko‘pchiligida belgilar satrlari alohida belgi bilan tugallanadi deyiladi: nolli bayt bo‘lib, 0 soni bilan taqdim etiladi.

char *str = “apples”;

Ushbu misolda str – char tipining ko‘rsatkichi bo‘lib, apples0 belgilar satrini manzillaydi, bu yerda 0 – nol belgii. str ko‘rsatkichini initsializatsiyalashga ruxsat berilgan, chunki belgilar satrlarining literallari char tipi ko‘rsatkichliari orqali taqdim etiladi.

C va C++ tillarida keng tarqalgan kutubxona funksiyalarining ba’zilarini aytib o‘tamiz, ular belgilar satrlari bilan amallar bajaradi. Ushbu strcpy funksiyasi satrlarni joyini o‘zgartiradi; strcat funksiyasi bir mavjud satrni boshqasi bilan konkatenlashtiradi; strcmp funksiyasi ikki berilgan starlarni taqqoslaydi (ASCII kodi bo‘yicha).

FORTRAN 77, FORTRAN 90 va BASIC tillarida satralr asosiy tiplar kabi interpretatsiya qilinadi va ular uchun qaabul qilish, konkatenatsiya, ssылka va munosabatlar operatorlari ko‘ri chiqilgan.

Quyidagi dastur bo‘lagini ko‘rib chiqamiz:

LETTER = ‘abcdefghijklmnopqrstuvwxyz’

WORDPAT = BREAK (LETTER) SPAN (LETTER) . WORD

LETTER – bu tezkor xarflarning baridan iborat satr qiymati hisoblangan qiymatdir. WORDPAT – bu namuna bo‘lib, u quyidagicha so‘zlarni tavsiflaydi: harf topilmaguncha boshida barcha belgilar qo‘yiladi, keyin esa to harf bo‘lmagan belgi topilmaguncha ushbu harflar birlashtiriladi. Namuna “.” operatoriga ham ega.

Ushbu namuna operatorda ham qo‘llanilishi mumkin

TEXT WORDPAT

“+” belshisi ismda kamida bitta belgi bo‘lishi lozimligini bildiradi. Shunday qilib, butun etalon satrga mos, bundan so‘ng esa kamida bitta harflar yoki sonlar keladi.

Quyidagi iboraviy namuna ko‘rib chiqamiz:

/\d+\.?\d*|\.\d+/

Ushbu namuna sonli o‘zgarmaslarga mos. \. belgilari sonning o‘nlik kattaligida nuqta beradi.

Satrlar uzunligi namunalari

Satrli kattaliklar uzunligiga tegishli bir qancha loyihaviy yechimlar mavjud. Birinchidan, uzunlik statistik bo‘lishi mumkin va e’londa berilishi mumkin. Bunday satr statik uzunlikli satr deyiladi. Bunday satrlar FORTRAN 77, FORTRAN 90, COBOL, Pascal va Ada tillarida mavjud. Masalan, FORTRAN 90 tilining quyidagi operatori NAME1 va NAME2 o‘zgaruvchilarini e’lon qiladi:

CHARACTER (LEN = 15) NAME1, NAME2

Statistik uzunlikdagi satrlar doimo to‘liq; agar satrli o‘zgaruvchiga kichik uzunlikdagi satr berilsa, bo‘sh joylar 0 belgilari bilan to‘ldiriladi.

Baholash

Satrli tiplar dasturlarning yozish osonligini oshiradi. Satrlarni chaqirish, elementar satrli tiplarni chaqirishdan osonroq.

Satrlar bilan amallar zaruriy hamda satrli tiplarga tegishli kattaliklarda mavjud bo‘lishi shart.

Belgili satrlarni amalga oshirish

Bog‘langan ro‘yxatli metodlarni qo‘llash katta hajmdagi xotirani talab qilgani bilan, bunda yuz beruvchi hodisalar oddiydir. Bunga qaramay, xotirada joylashtirish amali sekin bajariladi. Qo‘shilgan katakchalar metodi tarqatilgan va bo‘shatilgan

katakchalarni boshqarish muammosini keltirib chiqaradi. Bu muammo 5.10.10.3 bo‘limda to‘liq ko‘rib chiqilgan.

Foydalanuvchi tomonidan aniqlanuvchi ketma-ket tiplar

Ketma-ket tip deb, ehtimol qiymatlar sohasi natural raqamlar ketma-ketligi bilan bog‘liq bo‘lishi mumkin. Pascal va Ada tillarida asosiy ketma-ketlikli tiplar butun, belgili va mantiqiy tiplari hisoblanadi. Ko‘pgina tillarda foydalanuvchilarning o‘zi ikkita turli ketma-ket tiplarini aniqlashlari mumkin: sanaluvchi va cheklangan tiplar.

Sanaluvchi tiplar

Sanaluvchi tiplar deb, uni tavsiflashda barcha ehtimol qiymatlar sanab o‘tilgan tipga aytiladi. Oddiy sanaluvichi tip quyidagicha:

type DAYS is (Mon, Tue, Wed, Thu, Fri, Sat, Sun);

Sanaluvchi tiplarga xos muammo quyidagicha: literal o‘zgarmas bir nechta tip tavsifida kelishi mumkinmi va agar javob ha bo‘lsa, aniq literal tipi dasturda qanday aniqlanadi?

Strukturalar

Pascal tilida literal konstantani sanaluvchi tiplarning bir qancha tavsiflarida keltirish mumkin emas. Sanaluvchi tiplarga tegishli o‘zgaruvchilar massivlar indeksi kabi qo‘llanilishi mumkin, ammo kiritilishi yoki chiqarilishi mumkin emas.

```
type colortype = (red, blue, green, yellow);
var color : colortype;
…
color:=blue;
if color>red …
```

Bu yerda bul ifodasi if operatorida rost deb berilgan.

Massivlar

Massivlar deb bir turdagi ma’lumotlarga aytiladi va bunda har bir alohida element birinchi elementga muvofiq holda identifikatsiyalanadi. Massiv elementini

chaqirish dasturda ko‘pincha bir yoki bir nechta indeks o‘zgaruvchilarini saqlaydi. Bunday ssылkalarni bajarish vaqtida hisoblash mumkin.

Ishlab chiqarish muammolari

Quyida massivlarga xos bo‘lgan ishlab chiqarishda kelib chiquvchi muammolar keltirilgan:

Indekslar qanday tipda bo‘lishi mumkin?

Kerakli diapazondan indeksning chiqishiga yo‘l qo‘yiladimi?

Indekslar qiymatlar sohasi cheklanganmi?

Qachon massiv xotirada joylashadi?

Indekslarning qanchasiga yo‘l qo‘yilgan?

Massivlar ular xotiraga joylashtirilganidan so‘ng initsializatsiyalanishi mumkinmi?

Massivlarning qanday kesishmasiga yo‘l qo‘yiladi yoki bunga umuman ruxsat berilganmi o‘zi?

Keyingi bo‘limlarda loyihaviy yechimlarga misollar keltirilgan bo‘lib, dasturlashning eng keng tarqalgan tillarida qo‘llaniladi.

Assotsiatsiativ massivlar

Assotsiatsiativ massivlar deb, tekislanmagan ma’lumotlarning bir qancha elementlari tushuniladi, kalit deb nomlanuvchi xuddi shuncha kattaliklar bilan indekslanadi. Assotsiatsiativ bo‘lmagan massivlarda indekslarni saqlab bo‘lmaydi. Assotsiatsiativ massivlarda esa foydalanuvchi tomonidan aniqlanuvchi kalitlar massiv strukturasida saqlanishi lozim. Shunday qilib, assotsiatsiativ massivlarning har bir elementi elementlar juftligi bo‘ladi: kalit va kattalik.

Assotsiatsiativ massivlar uchun quyidagi ishlab chiqarish savollari xos hisoblanadi:

Elementlarni chaqirish formasi qanday?

Assotsiatsiativ massiv o‘lchami qanday: statistikmi yoki dinamikmi?

Strukturalar va amallar

Perl tilining assotsiatsiativ massivlari xesh deb ataladi, chunki ularning elementlari xotiraga joylanadi va xesh funksiyasi orqali ulardan chiqariladi. Perl tilida xesh ismlar fazosi aniq belgilangan: har bir xeshlangan o‘zgaruvchi protsent belgisi bilan boshlanishi kerak.

salaries = (“Cedric” => 7500, “Perry” => 57000,
“Mary” => 55750, “Gary” => 47850);

Pascal, Modula-2 va Ada tillari yozuvlarda COBOL tilida tasdiqlangan bosqichlar nomerini emas, balki o‘zinining sintaktik qoidalarini qo‘llaydi: ushbu tillarda yozuvlar strukturasini yaratish uchun ortogonal struktura qo‘llaniladi, u bir e’lonlarni boshqalariga kiritishga ruxsat beradi. Ada tilidagi quyidagi e’lonni ko‘rib chiqamiz:

EMPLOYEE_RECORD :
record
EMPLOYEE_NAME :
record
FIRST : STRING (1..20);
MIDDLE : STRING (1..10);
LAST : STRING (1..20);
end record
HOURLY_RATE : FLOAT;
end record;

S tilida yozuvlar ham ko‘rib chiqilgan va struktura deb nomlanadi. Ular ko‘p narsada Pascal tilidagi yozuvlarga o‘xshaydi. 5.8 bo‘limda ko‘rib chiqilgan variantli yozuvlar yoki birlashuvlarni strukturalar o‘zida saqlamasligi bundan mustasnodir.

FORTRAN 90 tilida yozuvlarni e’lon qilish har qanday kiritilgan yozuvlarni tip kabi tavsiflashni talab qiladi. Shunday qilib, yuqorida keltirilgan ishchi haqidagi yozuvlarja ishchi nomini yozish oldinan tavsiflashni talab qiladi, bundan keyin esa ishchi yozuvi maydonida u shunchaki tip deyiladi.

Glossariy

- **Ўзгарувчи** –программа объекти бўлиб, хотирадаги бир нечта ячейкаларни эгаллайди ва берилганларни сақлаш учун хизмат қилади;
- **Идентификатор** – катта ва кичик лотин ҳарфлари, рақамлар ва таг чизиқ („_“) белгиларидан ташкил топган ва рақамдан бошланмайдиган белгилар кетма–кетлиги тушунилади;
- **Дескриптор ўзгарувчилар** – атрибутларининг умумлашмаси бўлиб, улар ушбу атрибутларни сақловчи хотира катакчалари тўплами кўринишида амалга оширилади;
- **Объект** – ўзгарувчи қиймати ва у эгалловчи хотира ва фойдаланувчи томонидан аниқланувчи мавҳум типли маълумотлар нусхалари тушунилади;

- **Маълумотларнинг элементар типлари** – бошқа типлар таърифларида аниқланмайдиган маълумотлар типлари;
- **Константа (литерал)** - бу фиксирланган сонни, сатр ва белгини ифодаловчи лексемадир;
- **Ҳақиқий константалар** – сузувчи нуқтали сон бўлиб, ўнлик форматда (24.56; 13.0; 66., .87) èки экспоненциал шаклда (1e2; 5e+3; 3.14e–2; .25e4);
- **Мантиқий типлар** – ўгарувчилар қийматлар ўртасидаги муносабатларни ифодалайдиган мулоҳазаларни рост (true) ёки ёлғон (false) эканлигини тавсифлашда қўлланилади;
- **ASCII** (American Standard Code for Information Interchange) – Ахборот алмашинуви учун Америка стандарт коди).

Амалий машғулот 2.

Берилганлар турлари. C# тилининг таянч турлари

Ишдан мақсад: C++ дастурлаш тилида берилганларнинг турли кўринишларидан фойдаланишни ўрганиш, улардан фойдалана олиш.

Масаланинг қўйилиши: Тингловчи вариант бўйича берилган масалани C++ дастурлаш тилида ишлаши ва керакли натижа олиши лозим.

Ишни бажариш учун намуна

Мисол: Учта томони билан берилган учбурчакнинг периметри ва юзаси топилсин.

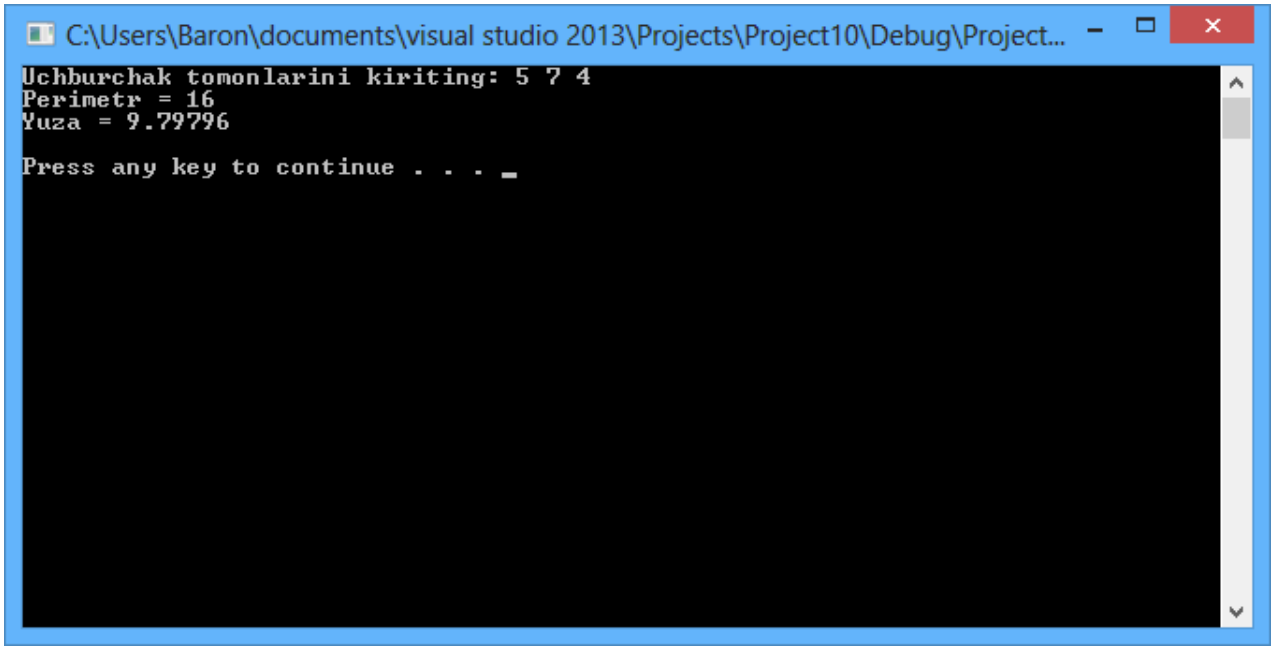
Дастур коди:

dastur.cpp файли:

```
#include <iostream>

#include <cmath>

using namespace std;
```

```
int main(){
        int a, b, c;
        double p, s;
        cout << "Uchburchak tomonlarini kiriting: ";
        cin >> a >> b >> c;
        p = (a + b + c) / 2.;
        s = sqrt(p * (p-a) * (p-b) * (p-c));
        cout << "Perimetr = " << p*2 << endl;
        cout << "Yuza = " << s << endl;
        system("pause");
        return 0;
        }
```

Дастур ишлаши натижаси:



Амалий топшириқлар

1. Турғун сувдаги қайиқ тезлиги V км/с. Дарё суви оқимининг тезлиги U км/с (U<V). Қайиқ кўлда T_1 соат, дарёда эса (оқимга қарши) T_2 соат ҳаракат қилган. Қайиқ сузган умумий S масофа топилсин.
2. Биринчи автомобил тезлиги V_1 км/с, иккинчисиники - V_2 км/с, улар орасидаги масофа - S км. Автомобиллар бир-биридан узоқлашса (бир-бирига қараб ҳаракат қилганда), T соатдан кейин улар орасидаги масофа қандай бўлади?
3. Асослари a ва b (a>b), катта асосдаги бурчаги α бўлган тенг ёнли трапетсиянинг периметри ҳамда юзаси топилсин (бурчак радианда берилади).
4. Нолдан фарқли берилган R_1, R_2, R_3 электр қаршиликлари учун R ҳисоблансин. Бунда: $\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}$.
5. Ходимнинг ойлик иш ҳақига 45% мукофот пули қўшилсин. Ҳосил бўлган миқдордан 17% даромад солиғи, 1,5% касаба уюшмаси ва 1% нафақа солиғи ушлаб қолинсин. Қўлга тегадиган пул миқдори чоп этилсин.
6. Уч хонали бутун сон (k) рақамлари йиғиндисини (s) бутун ўзгарувчига ўзлаштирилсин.
7. Тенг томонли учбурчак томони берилган, учбурчак юзаси топилсин.
8. Учта мусбат сон берилган. Сонлар ўрта геометригининг каср қисми топилсин.
9. Берилган катетлари бўйича тўғри бурчакли учбурчакнинг периметри ва юзаси ҳисоблансин.
10. Берилган икки томони ва улар орасидаги бурчак (градусда) асосида учбурчакнинг учинчи томони ва юзаси топилсин.
11. Берилган уч хонали сон рақамларини тескари тартибда ёзишдан ҳосил бўлган сон топилсин. Масалан, 345 сонининг тескари тартиби 543

бўлади.

Тестлар

1. **Маълумотлар типларининг замонавий концепциялари неча йиллардан бери ривожланиб келмоқда?**
   a) 40
   b) 45
   c) 60
   d) 70

2. Иловалар соҳасини кенгайтириш мақсадида қайси тилга турли типдаги маълумотлар киритилган?
   a) Ada
   b) FORTRAN
   c) PL/I
   d) COBOL

3. Қайси йилда Ada 83 тилига маълумотлар типларини шакллантириш концепцияси киритилган?
   a) 1975
   b) 1985
   c) 1950
   d) 1970

4. Фойдаланувчи томонидан аниқланувчи мавҳум типли маълумотлар нечта мавзудан кўриб чиқилган?
   a) 5
   b) 10
   c) 17
   d) 25

5. Ўзгарувчи қиймат ва эгалловчи хотира деганда нима тушунилади?
   a) тил
   b) тип
   c) объект
   d) қиймат

6. Ada тили амалга оширилганда қайси дастурлаш тиллари бутун сонларнинг ишорасиз типларига эга?

a) SHORT INTEGER
b) INTEGER
c) LONG INTEGER
d) Ҳамма жавоблар тўғри.

7. Knuth, 1981 китобида қандай муаммолар тўлиқ кўриб чиқилган?

a) Сузувчи нуқтали сонларни тақдим этиш муаммоси.
b) Сонли типларни тақдим этиш муаммоси.
c) Элементар типларни тақдим этиш муаммоси.
d) Структуравий типларни тақдим этиш муаммоси.

8. Нечинчи йилдан бошлаб бул типлари дастурлашнинг кўпгина универсал тилларига қўшилди?

a) 1960
b) 1970
c) 1980
d) 1990

9. ASCII стандартида қандай қийматли диапазон қўлланилади?

a) 0-150
b) 0-176
c) 0-127
d) 0-131

10. Unicode символлар тўплами неча битдан иборат?

a) 8
b) 32
c) 16
d) 64

3-Mavzu. Ifodalar va operatorlar

Reja

1. Kirish

2. Arifmetik ifodalar
3. Qo‘shimcha yuklangan operatorlar
4. Tiplarni o‘zgartirish

Sarlavha ko‘rsatib turganidek, ushbu mavzuning mavzusi ifodalar va tayinlash buyruqlari hisoblanadi. Ifodalar operatorlarining baholash tartibini aniqlash semantik qoidalari birinchi muhokama qilinadi. Bundan keyin esa operandlarning ketma-ketligini baholash jarayoni bilan mumkin bo‘lgan muammoni tushuntirish mumkin bo‘ladi (bunda funksiyalar vazifalari yon ta’siri bo‘lishi mumkin). Qo‘shimcha o‘rnatilgan operatorlari (***Overloaded operators***), ham oldindan belgilagan ham foydalanuvchi belgilagan, keyin dasturlardagi ifodalarga o‘z ta’siri bilan birga, muhokama qilinadi. Keyingi, aralash-rejimi ifodalar tasvirlangan va baholanadi. Relatsion va mantiqiy ifodalar jarayoni, shu jumladan, qisqatutashuv-baholash muhokama qilinadi. Belgili satr holat-mosligi (***pattern matching***) oldingi mavzuda ham belgi satrlari ustida materiallarning bir qismi yoritilgan, shuning uchun ular bu mavzuda zikr qilinmaydi.

Kirish

Ifodalar—dasturiy tildagi hisoblarni ko‘rsatishni asosiy vositasidir. Dasturchi uchun, ishlatilayotgan tilning ham sintaksisini va semantik ifodalarni tushunish muhim. Ifodalar sintaksisini tasvirlab berish uchun formal mexanizm **(BNF)** avvalgi mavzuda tanishtirilgan. Ushbu mavzuda ifodalarning semantikligi muhokama qilinadi.

Ifodalarni baholanishini tushunish uchun, operator va operand baholash tartibi bilan tanish bo‘lishi lozim. Ifodalarning operator baholash tartibi tilining bog‘liqlik va birinchilik qoidalari bilan boshqariladi (aniqlanadi). Bir ifoda qiymati ba’zan, bunga bog‘liq bo‘lsa-da ifodalarda operand tartibini baholash ko‘pincha til loyihalar tomonidan berilmaydi(unstated). Bu amalga oshiruvchilarga dasturlari tartibni tanlash imkoniyati olib keladi, qaysiki turli dasturlarda turli natijalar ishlab chiqarishga sabab bo‘ladi. Ifodalardagi boshqa semantik masalalar: turni mos tushmasligi.

Imperativ dasturlash tillarining mohiyati tayinlash ifodalarining dominant o‘rnidir. Bu buyruqlarning maqsadi dasturning o‘zgaruvchilar qiymatlari yoki holatini o‘zgarish yon ta’siriga sabab bo‘lishi mumkin. Shunday qilib, barcha imperativ tillarning ajralmas qismi dastur ijro paytida o‘zgaradigan o‘zgaruvchi tushunchasidir.

Funksional tillar, funksiya parametrlari kabi, har xil turdagi o‘zgaruvchilar foydalanadi. Bu tillarda ham nomlarini qiymatlarga bog‘laydigan deklaratsiya bayonotlar bor. Bu deklaratsiyalar tayinlash buyruqlariga o‘xshash, ammo yon ta’siri yo‘q.

Arifmetika ifodalar

Yuqori darajadagi dasturlash tillari birinchi asosiy maqsadlaridan biri matematika, ilm-fan va muhandislikda topilgan arifmetik ifodalarni avtomatik baholash bo‘lgan. Dasturlash tillarida arifmetik ifodalarning ko‘pchilik xususiyatlari matematikada rivojlangan konvensiyalardan meros edi. Dasturlash tillarida, arifmetik ifodalar operatorlar, operandlar, qavslar va funksiya chaqiruvlaridan iborat. Operatorlar “Unar”, ya’ni u bir operandli, “binar” ya’ni ikki operandli “ternari”(ma’nosi uchta operandi bor) bo‘lishi mumkin.

Ko‘pchilik dasturlash tillarida, binar operatorlari ***infix,*** ular operandlar o‘rtasida bo‘ladi. Bir istisno bu Perl, ba’zi operatorlari ***prefix*** ular operandlaridan oldin keladi.

Arifmetik ifodaning maqsadi - arifmetik hisoblashni belgilash hisoblanadi. Bunday hisoblash amalga oshirish ikki harakatlarga sabab bo‘lishi kerak: operandlarni olib kelish, odatda xotiradan, bu operandlar ustida arifmetik amallarni bajarish. Quyidagi qismilarda, biz arifmetik ifodalarning umumiy loyihasini tadqiq etamiz.

Arifmetik ifodalar uchun asosiy dizayn masalalari quyidagilarni, barchasi shu qismida muhokama qilinadi:

- Operator ustunlik qoidalari nima?
- Operator bog‘liqlik(birlashish) qoidalari nima?
- Operand baholash tartibi qanday?
- Operand baholash yon ta’sirida cheklashlar mavjudmi?
- Til foydalanuvchiga ortiqcha yuklangan belgilangan operatori yo‘l qo‘ya oladimi?
- Ifodalarda qanday tur aralashtirish yo‘l qo‘yiladi?

Operator hisoblash tartibi

Bir tilning operator ustunligi va birlashish qoidalari uning operatorlarini hisoblash tartibini aytib beradi.

O‘rin (Navbat , Ketma ketlik)

Bir ifoda qiymati kamida ifodadagi operatorlarini hisoblash tartibidagi qismga bog‘liq. Quyidagi ifoda ko‘rib chiqaylik:

```
a + b * c
```

Faraz qilaylik, o‘zgaruvchilar a, b va c deylik va mos ravishda 3, 4 va 5. Agar (birinchi qo‘shish, keyin ko‘paytirish) chapdan o‘ngga hisoblansa natija 35. Agar o‘ngdan chapga hisoblansa natijasi 23 bo‘ladi.

Ifodalardagi operatorlarni oddiygina chapdan o‘ngga yoki o‘ngdan chapga hisoblash o‘rniga, matematiklar uzoq yillar oldin hisoblash ustuvorligining ierarxiyasida operatorlarni joylashtirish va qisman bu ierarxiyada ifodalarning hisoblash tartibini asoslash tushunchasini rivojlantirishgan. Misol uchun, matematikada ko‘paytirish qo‘shishdan ko‘ra ortiq oliy ustuvor hisoblanadi, ehtimol uning yuqori darajasi murakkabligi tufaylidir. Agar bu usul oldingi misoldagi ifodaga qo‘llaniladigan bo‘lsa, ko‘p dasturlash tillaridagi kabi, ko‘paytirish birinchi qilinishi kerak bo‘ladi.

Ifoda baholash uchun operator ustunlik qoidalari, turli darajadagi ustunlikning operatorlari xisoblash tartibini qisman aniqlaydi. Ifodalar uchun operator qoidalari, til loyihalari tomonidan ko‘rilgan, operator ustuvorligi ierarxiyasiga asoslangan. Odatiy imperativ tillarning operator ustunlik qoidalari deyarli barcha bir xil bo‘ladi, chunki ular o‘sha matematikaga asoslanadi. Bu tillarda, eksponent eng yuqori o‘ringa (u til tomonidan taqdim qilingan paytda) ega, ortidan ko‘paytirish va bo‘lish, keyin shu darajada olish binary qo‘shish va ayirish.

Ko‘pchilik tillar qo‘shish va ayirishning unar versiyalarini ham o‘z ichiga oladi. u bir terimli Kiritilgan identifikatsiya operatori deb ataladi , chunki u odatda hech qanday bog‘liqlik amali yo‘q va shu tariqa uning operand ustida hech qanday ta’siri yo‘q. Ellis va Stroustrup (1990, bet 56), **S ++** haqida gapirib, uni tarixiy halokat deb ataydi va uni foydasizligini ko‘rsatib beradi. **Unar minus**, albatta, o‘zining operandi belgisini o‘zgartiradi. Shuningdek, **Java** va **S#** da, **unar minus, short** va **bayt** operandlarining **int** turiga mutloq konvertatsiyasiga sabab bo‘ladi.

Barcha odatiy imperativ tillarda, unar minus operatori , modomiki u qavslar bilan ajratilib boshqa operatorlar yonida bo‘lishdan oldini olinar ekan, ifodaning yo boshida yo istagan joyida ko‘rinishi (paydo bo‘lishi) mumkin. Masalan,

```
a + (- b) * c
```

bo‘lgan to‘g‘ri (legal), lekin

```
a + - b * c
```

odatda to‘g‘ri emas.

Keyingi, quyidagi ifodalarni ko‘rib:

```
- a/ b
- a * b
- a ** b
```

Birinchi ikki hollarda, unar minus operator va ikkilik operatorning nisbiy ustunligi ikki operatorlar baholash tartibining ifoda qiymatiga ta’siri yo‘q. Oxirgi holatda, biroq, u muhim emas.

Umumiy dasturlash tillarining, faqat **Fortran, Ruby, Visual Basik** va **Ada** eksponention operatori bor. Barcha to‘rtalasida,eksponention unar minusdan yuqori ustunlikka ega, shuning uchun

```
- A ** B teng bo‘ladi  - (A ** B)
```

Ruby va S asoslangan tillarda arifmetik amalar ketma-ketligi quyidagicha:

.

Ruby		S-asoslangan tillar
Eng birinchi	**	postfix ++, --
.	unar +-	prefix ++,--, unar + , -
.	*,/,%	*,/,%
Eng oxirgi	binary +,-	binary +,-

****** Operator eksponent hisoblanadi. **%** Operatori ikki butun operand olib, birinchisini ikkinchisiga bo‘lishdan qoldig‘ini beradi.

APL(Application programming language) tillar orasidan ajralib turadi, chunki ketma-ketlikning bir darajasiga ega sifatida,. Keyingi mavzuda tasvirlanganidek.

Ustunlik operatori operator baholashning tartibining faqat ayrim qoidalariga hisobga oladi, birlashish qoidalari ham ta’sir qiladi.

Birlashish qoidalari

Quyidagi ifoda ko‘rib chiqaylik:

```
a - b + c - d
```

Qo‘shish va ayirish operatorlari bir xil darajadagi o‘ringa ega, ular dasturlash tillarida bajarganidek, ustunlik qoidalari bu ifodadagi operatorlar baholash tartibi haqida hech narsa aytmaydi.

Bir ifoda operatorlarining bir xil o‘rin darajada ikki qo‘shni ko‘rinishini o‘z ichiga olsa, qaysi operator birinchi baholanadi degan savolga tilining birlashish(bog‘liqlik) qoidalari javob beradi. Operator yo chap yoki o‘ng bog‘liqlikda bo‘lishi mumkin , ya’ni bir xil o‘rin darajada ikki qo‘shni operatorlari kelganda,

birinchi chap operator baholanadi yoki o‘ng operator birinchi baholanadi, mos ravishda.

Umumiy tillarda birlashish chapdan o‘ngga qilinadi, eksponention operatori berilgan holat bundan mustasno,bunda(Taqdim etilgan) operator ba’zan o‘ngdan chapga qilinadi. **Java**da bu ifoda

```
a - b + c
```

chap operator birinchi baholanadi.

Fortran va Ruby da eksponention o‘ng biriktirilish, shuning uchun bu ifoda

```
A ** B ** C
```

o‘ng operator birinchi baholanadi.

Ada da eksponention nonassociative bo‘ladi,shuning uchun quyidagi ifoda noqonuniy hisoblanadi.

```
A ** B ** C
```

Bunday ifoda, istalgan tartibini ko‘rsatish uchun qavslar kerak

```
(A ** B) ** C
```

Kabi yoki

```
A ** (B ** C)
```

yoki Visual Basis, eksponention operator, ^, chap assotsiativ hisoblanadi.

bir necha umumiy tillar uchun assotsiativ (birlashish) qoidalari bu yerda taqdim etiladi:

Til	associativit qoidasi
Ruby	Chap : *,/,+,-
	O‘ng : **
S –ga asoslangan tillar	Chap: *,/,%binary +,binary –
	O‘ng: ++,--,unar -,unar +
Ada	Chap: barchasi ,** dan tashqari
	Associativ emas :**

APLda , barcha operatorlar bir xil darajada ustunlikka ega. Shunday qilib, APLifodalarda operatorlari baholash tartibi barcha operatorlari uchun o‘ngdan chapga bo‘lgan associativy qoidasi bilan aniqlanadi. Misol uchun, ifoda

```
A × B + C
```

Qo‘shish operatori birinchi baholanadi, keyin Ko‘paytirish amali(* **APL**da ko‘paytirish operatori hisoblanadi). A 3 bo‘lsa edi, B 4 edi, va S 5 edi , keyin bu **APL** ifoda qiymati 27 bo‘ladi.

Umumiy tillar uchun ko‘p kompilyatorlar Aslida foydalanish, ba’zi arifmetik operatorlar matematik Assotsiativ ligi faktidan,qaysiki associative qoidalari faqat o‘z ichiga olgan ifoda qiymati hech qanday ta’sir ko‘rsatmaydi, foydalanadi. Misol uchun, qo‘shish matematik assotsiativ bo‘ladi, shuning uchun, matematikada ifoda

A + B + S

operator baholash tartibi bog‘liq emas. Floating-point amallarida bo‘lsa matematik Birlashtiruvchi operatsiyalar ham associativ edi, kompilyator ba’zi oddiy cheklovlarni bajarish uchun bu faktlardan foydalanishi mumkin. Xususan, agar kompilyatorga operatorlari qayta baho tartiblashtirish uchun ruxsat etilsa, u ifodani baholash uchun bir oz tezroq kodni ishlab chiqarishi mumkin. Kompilyatorlar tez-tez bu turdagi optimallashtirishni amalga oshiradi.

Afsuski, kompyuterda, har ikkila haqiqiy son (1.356885985864448) ko‘rinishlari va arifmetik operatsiyalari, ularning matematikadagi shu tushunchalarning faqat taxminaniy yaxlitlanganligidir (chunki hajmi cheklashlar). matematik operator associative ekanliginifloating point operationlari assotsiativligini anglatmaydi. Aslida, faqat agar hamma operandlar va oraliq natijalari aniq floating-point da namoyish qilinsa jarayon aniqroq (associativ) biriktirilishi mumkin. Misol uchun, kompyuterda integer kiritilgan assotsiativ bolmaydigan patologik holatlar bor. Masalan, faraz qiling dastur ifodani baholash kerak,

A + B + S + D

va **A** va **S** juda katta musbat raqamlar, va **B** va **D** manfiy raqamlar juda katta absolut qimatdagi. Bu vaziyatda, **A B** qo‘shish oshib ketish xatosini chiqarmaydi, lekin **A S** qo‘shishda boladi xato . Xuddi shu tarzda, **S B** qo‘shish toshib ketish xatosini chiqarmaydi, lekin **B D** qo‘shishda boladi. Kompyuter arifmetik cheklanishi sabab, qo‘shish bu holatda esa fojiali nonassociative hisoblanadi. Shuning uchun agar Kompilyator bu qo‘shish operatsiyalarni qaytadan joylashtirsa ,bu ifoda qiymatiga ta’sir qiladi. Bu muammo, albatta, dasturchi tomonidan oldini olinishi mumkin ,o‘zgaruvchilar taxminiy qiymatlarini ma’lum holda. Dasturchi, (ikki belgilash jadvaldagi ham) ifodani ikki qismdan ko‘rsatish mumkin, toshib ketishni oldini olishni ta’minlab. Ammo, bu holat ancha nozik yo‘llar bilan yuzaga chiqishi mumkin, unda dasturchi tartibini farq qilish uchun kamroq imkoniyat bo‘ladi.

Qavslar

Dasturchilar ifodalarda qavslar joylashtirish bilan ustunlik va birlashish qoidalarini o‘zgartirishlari mumkin. ifodaning qavslar bilan oralgan qismida uning

qo‘shni unparenthesized qismlari dan ustunlik bor. Misol uchun, ko‘paytirish qoshish dan ustun tursada , bu ifodada

```
(A + B) * C
```

Qo‘shish birinchi baholanadi. Matematik jihatdan, bu mukammal tabiiy. Bu ifoda, ko‘paytirish operatorining birinchi operandi mavjud emas toki Qavsga olingan kichik ifodadagi qo‘shish baholanmaguncha. Bundan tashqari, Qismi 2.1.2 dan ifoda quyidagicha belgilangan bo‘lishi mumkin

```
(A + B) + (S + D)
```

toshib ketishni oldini olish uchun.

Arifmetik ifodalarda Qavslar mumkin bo‘lgan tillar Barcha ustunlik qoidalariga bo‘lib olishi va shunchaki operatorlarni chapdan o‘ng yoki o‘ng dan chapga sherik qiladi. Dasturchi istalgan tartibini Qavslar bilan belgilash mumkin . Bu yondashuv oddiy bo‘ladi, chunki, na muallif na dastur o‘quvchilari har qanday ustunlik yoki associativelik qoidalarini yodda saqlashi kerak. Bu sxema yomon tomoni :yozuv ifodalar ko‘proq zerikarli qiladi va u kodi oqilishiga(chiroyli syntax) ham jiddiy xavf. Biroq, bu Ken Iverson tomonidan amalgaoshirilgan tanlov edi,APL dizayneri.

Ruby iboralar

Ruby sof ob’ekt yo‘naltirilgan til ekanligini eslab , orasida, boshqa narsalar orasida, har bir ma’lumotlar qiymati, ob’ekt hisoblanadi. Ruby Sga asoslangan tillarda kiritilgan arifmetik va mantiqiy operatsiyalar to‘plamni qo‘llab-quvvatlaydi. Ifodalar sohasida S asoslangan tillarida farqli tomoni Ruby ning arifmetik, bogliqlik va tenglashtirish operatorlari, shuningdek masssiv indeks ,siljishlar va bit bilan ishlovchi mantiq operatorlari funksiya sifatida amalga oshirildi. Misol uchun, ifoda a + b a Ob’ektinining + funksiyasiga chaqiruv b obektni parameter qilibberilgan.

Operatorlarni funksiya qilib amalga oshirishning biri qiziqarli jihati shuki ular amaliy dasturlar tomonidan qayta yozilishi mumkin. Shuning uchun, bu operatorlari qayta aniqlanishi mumkin. oldindan belgilangan operatorlarni tez-tez qayta aniqlash foydali bo‘lmasa-da biz hali 3 da ko‘ramiz, ba’zi tillarda foydalanuvchi belgilangan turlari ortiqcha yuklash operatorlari bilan amalga oshirilishi mumkin.

LISP dagi ifodalar

Ruby bilan bo‘lgani kabi, LISPdagi barcha arifmetik va mantiqiy operatsiyalar kichik dasturlarni tomonidan amalga oshiriladi. Lekin LISP yilda, kichik dastur ochiq

chaqirilishi kerak. Misol uchun, LISP yilda S ifoda a + b * c belgilash uchun, quyidagi ifodani yozish lozim:

```
(+ a (* b c))
```

Bu ifoda esa, + va * vazifalari nomlari bor.

Shart operatorlari

If-then-else so‘zlar shartli ifoda amalga oshirish uchun foydalanish mumkin. Misol uchun,

```
if(count== 0 )
 average = 0;
     else
average = sum / count;
```

S asoslangan tilda, bu kod, ko‘proq qulay belgilangan shakliga ega bo‘lgan shartli bayonot ifoda yordamida belgilash mumkin,

```
expression_1? expression_2: expression_3
```

Bunda expression1 Boolean ifodasi sifatida talqin etiladi. Agar expression1 to‘gri bo‘lsa butun ifoda qiymati expression2 qiymati hisoblanadi ;aks holda, u expression3 qiymati hisoblanadi. Misol uchun, ***If-then-else*** misoli ta’siri quyidagi belgilash ifodalaridan bilan erishish mumkin, shartli ifodadan foydalanib:

```
average =(count== 0)? 0: sum / count;
```

Aslida, savol belgisi ***then*** bandida boshlanishini anglatadi va ‘:’ ***else*** bandida boshlanganini anglatardi. Har ikki bandlar bo‘lishi shart. Eslab qoling ? uchlik operatori sifatida shartli ifodalarda ishlatiladi.

Shartli ifodalar ,boshqa har qanday ifodadan foydalanish mumkin bolgan dasturdagi(S ga asoslangan tilda) har bir joyda foydalanish mumkin. S asoslangan tillardan tashqari, shartli ifodalar **Perl, JavaScript,** va **Ruby** da berilgan.

Operand baholash tartibi

Ifodalarning kam tarqalgan muhokama dizayn xususiyati operanddan baholash tartibi hisoblanadi. ifodalardagi o‘zgaruvchilar ularning qiymatlarini xotiradan olib kelish bilan baholanadi. O‘zgarmaslar ham ba’zan shu tarzda baholanadi. Boshqa holatlarda, o‘zgarmaslar mashina buyruq tili qismi bo‘ladi va xotiradan olib kelish talab qilinmaydi. Agar operand qavslar bilan o‘ralgan ifoda bo‘lsa, ozining qiymati ishlatilmasdan oldin, uning barcha operatorlari baholanishi kerak.

Operatorning operandlarida yon ta’siri bo‘lmasa, u holda operand baholash tartibi ahamiyatsiz bo‘ladi. Shuning uchun, qiziqarli holat faqat operand baholash yon ta’siribor bo‘lgenda paydo bo‘ladi.

Qo‘shimcha yuklangan operatorlar

Arifmetik operatorlar ko‘pincha bir necha maqsad uchun ishlatiladi. Masalan, + odatda integer va floating-point qo‘shishni ibelgilash uchun ishlatiladi. Bir qancha tillar-Java, masalan- buni string zanjir hosil qilish uchun foydalanadi. bu bir necha operator foydalanish qo‘shimchayuklangan operator deyiladi va odatda, deb hisoblanadi va modomiki na o‘qiluvchanligi na ishinchlilik jihatdan, umuman qabul qilinadi.

Qo‘shimcha yuklash xavfining misoli qilib **S++** dagi (&) belgisi ishlatilishini ko‘ring. Binary operatori sifatida, **AND** mantiqiy amalini bildiradi. Unar operatori sifatida, shu bilan birga, uning ma’nosi butunlay farq qiladi. uning operand sifatida o‘zgaruvchilar bilan unar operatori sifatida, ifoda qiymati bu o‘zgaruvchilar manzili. Bu holda, & belgisi manzili operatori deb ataladi. Misol uchun,

```
x = &;
```

ijrosi y manzili x joylashtirilgan bo‘ladi. Shu & turli foydalanishi bilan ikki muammo bor. Birinchidan, butunlay ikki bog‘liq bo‘lmagan operatsiyalar uchun bir xil ramzi foydalanish o‘qiluvchanlik uchun zararli hisoblanadi. Ikkinchidan, oddiy bitta operandning qoldirib ketilishi **AND** operatorida kompiler tomonidan aniqlanmay qolishi mumkin, chunki,u address operatori sifatida qaraladi bu paytda. Bunday xatoni tashxis qilish qiyin bo‘lishi mumkin.

Deyarli barcha dasturiy tillari, jiddiy bo‘lmagan lekin shunga o‘xshash muammo bor, minus operator haddan tashqari tez-tez qo‘shimcha yuklanishi bilan paydo bo‘ladi. Muammo shuki kompilyator operator unar yoki binar ekanligini ayta olmaydi. Shunday qilib, yana bir bor, Birinchi operandni qoldirib ketish,operator binary deb hisoblanganda, kompilyator tomonidan xato sifatida ko‘rilmaydi. Shu bilan birga , binary va unar operatorlarining ma’nolari, kamida yaqindan bog‘liqligi bor, shuning uchun o‘quvchanlikka salbiy ta’sir qilmaydi.

Abstrakt ma’lumotlar turlarini qo‘llab-quvvatlovchi ba’zi tillar uchun, masalan, **S ++, S #,** va **F#,** dasturchiga qo‘shimcha simbollar operatori yuklashni imkonini beradi. Misol uchun, bir foydalanuvchi skalar integer va butun son array qator o‘rtasida * operatorini massivning har bir elementi skalarga kopaytirishni bajaradigan qilishini aniqlashni istaydi deylik. Bunday operator *nomli kichkina

subprogram funksiya (* ning yangi holatini aniqlovchi) yozib amalga mumkin. Kompilyatorning o‘zi tilning-ozida bor qo‘shimcha yuklash operatori kabi operandlarning turiga qarab kerakli ma’nosini tanlab oladi. Misol uchun, *, bu yangi ta’rifi S # dasturi uchun belgilangan bo‘lsa, S # kompilyatori qachon chap operandda ineteger son va o‘ng operandda integer massiv ni ko‘rsa * ning yangi ma’nosini ishlatadi. OQILONA ishlatiladigan bo‘lsa, foydalanuvchi tomonidan yuklangan operator o‘rnatish o‘quvchanlikka yordam mumkin. Misol uchun,agar + va * Matritsa abstrakt ma’lumotlar tur uchun yuklangan va A, B ,S va D bu turdagi o‘zgaruvchilar bo‘lsa

```
A * B+ S * D
```

Quyidagining o‘rniga foydalanish mumkin

```
MatrixAdd (MatrixMult (A, B), MatrixMult (S, D))
```

Boshqa tomondan, foydalanuvchi tomonidan yuklangan operator o‘quvchanlikka uchun zararli bo‘lishi mumkin. Bir narsa uchun, hech narsa foydalanuvchini ko‘paytirirsh ma’nosida + belgisini dan foydalanishni oldini ololmaydi.Bundan tashqari, bir dasturdagi * operatorini ko‘rib, o‘quvchi ham operand turlari va operator ma’nosini aniqlashi kerak. Bu ta’riflar istalgan biri yoki barchasi boshqa fayllarda bo‘lishi mumkin.

Yana bir hisobga olinadigan narsa qurish jarayoni turli guruhlar tomonidan yaratilgan dasturiy ta’minot tizimi yaratish jarayoni. Turli guruhlar bir xil operatorlarni qo‘shimcha yuklagan bo‘lsa tizimni birgalikda qo‘yishdan oldin, bu farqlar bartaraf qilish kerak.

S ++ qo‘shimcha yuklab bo‘lmaydigan bir necha operator mavjud. Bular orasida sinf yoki tuzilishi a’zo operator (.) va ko‘lami qaror operator (: :). Qizig‘i shundaki, operator qo‘shimcha yuklash S ++ ning Java ko‘chirilmagan xususiyatlaridan biri edi. Biroq, u **S #** da qayta ko‘rindi.

Tiplarni o‘zgartirish

Amaliyotda ko‘p hollarda tiplarni o‘zgartirishga to‘g‘ri keladi. S# da tiplarni o‘zgartirishning 2 xil ko‘rinishi bo‘lib oshkormas va oshkor.

Oshkormas tip almashtirish qiymat o‘zlashtirilayotganda avtomatik amalga oshiriladi. Masalan, **int** yoki **short** tipiga mansub qiymat **long** tipiga mansub o‘zgaruvchiga berilganda. Quyidagi misolda **int** qiymatlar yig‘indici **long** tipiga o‘zlashtirilyapti:

```
int a=34;
int b=45;
long c =a + b;
Console.WriteLine("c=" + c);
```

long tipi **int** tipiga nisbatan baytlar soni ko‘p bo‘lganligi uchun bu yerda xatolik bo‘lmaydi. Quyidagi jadvalda **C#** tilidagi oshkormas almashtirishlar keltirilgan:

sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long, ulong	float, double, decimal
float	Double
char	ushort, int, uint, long, ulong, float, double, decimal

Oshkor tip almashtirish

Ayrim hollarda tiplarni almashtirish oshkor tarzda amalga oshiriladi. Quyidagi oshkormas tip almashtirish hollarda xatolik yuzaga keladi:

int ni **short** ga – ma’lumot yo‘qotilishi mumkin

int ni **uint** ga – ma’lumot yo‘qotilishi mumkin

uint ni **int** ga – ma’lumot yo‘qotilishi mumkin

float ni **int** ga – ma’lumot yo‘qotilishi mumkin

ixtiyoriy sonli tipni **char** tipiga - ma’lumot yo‘qotilishi mumkin

decimal ni boshqa ixtiyoriy sonli tipga – o‘nli tip boshqacha tashkil qilinganligi bois, ma’lumot yo‘qotilishi mumkin.

Bunday hollar uchun oshkor tip almashtirish qo‘llaniladi. Tipik sintaksisi quyidagicha:

qabul_qiluvchi =(qabul_qiluvchi_tipi) ifoda;

```
long a;
int b;
```

```
a = 2300;
b = (int)a;
```

Biror tip boshqasiga o‘tkazilganda, kompilyator ushbu amalni bajarishi lozim.

```
long a;
int b;
a = 2300000000;
b = (int)a;
Console.WriteLine(b);
```

Mana shu dastur qismi bajarilganda ekranda -1994967296 soni hosil bo‘ladi. Demak bu yerdan ko‘rinib turibdiki tiplarni almashtirganda shu tip diapozoniga tushadigan qiymatlarni o‘tkazish zarur aks holda xatolik yuz beradi. C# tilida **checked** operatori mavjud bo‘lib, stekning holatini aniqlash mumkin ya’ni xatolik yuz bergani to‘g‘risida xabar beradi.

```
long a;
int b;
a = 2300000;
b = checked((int)a);
```

Tiplarni almashtirish xatoliklarni hosil qilishi mumkinligini inobatga olib, xatoliklarni qayta ishlovchi **try...catch** konstruksiyasidan foydalanish ham mumkin. Masalan:

```
long a;
int b;
a = 2300000000;
try {
b = checked((int)a);
Console.WriteLine(b);
}
catch {
Console.WriteLine("Tiplarni almashtirganda xatolik yuz berdi");
}
```

Quyidagi misolda suzuvchi nuqtali tipni butun son tipiga almashtirilyapti:

```
double d=10.23;
int i;
i = (int)(d + 2.4);
```

Keyingi misolda ishorasiz butun son **char** tipiga o‘tkaziladi:

```
ushort c = 38;
char sym = (char)c;
Console.WriteLine(sym);
```

Ekranga ASCII kodining 38-kodli & simvoli chiqariladi. Quyidagi misolda decimal qiymat char ga o‘tkazilgan:

```
decimal x = 105m;
char sim = (char) x;
      Console.WriteLine(sim);// i harfi hosil bo‘ladi
  char s = 'i';
decimal d = (decimal) s;
Console.WriteLine(d);// 105 qiymati qaytariladi
```

Agar hosil qilingan natija, yangi tipga o‘tkazilmasa, tip almashtirish baribir amalga oshiriladi. Ammo natija kutilayotgan qiymatni qaytarmaydi. Masalan,

```
int i = -1;
char symbol = (char) i;
```

Ushbu tip almashtirish amalga oshirilmasligi lozim. Ammo natija sifatida ? simvoli qaytariladi.

Agar sonli tipli qiymatni harfiy tipga o‘tkazish lozim bo‘lsa, **.NET** bibliotekasi klasslari metodlaridan foydalanish lozim. **object** klassi **ToString()** metodiga ega bo‘lib, ushbu amalni bajarish mumkin:

```
int i = 10;
string s = i.ToString();
```

Shuningdek, harfiy tipli qiymatni sonli tipga o‘tkazish uchun **Parse()** metodidan foydalaniladi:

```
string s = 100;
int i = int.Parse(s);
Console.WriteLine(i+50);
```

Converter klassi bir tipdan ikkinchi tipga o‘tkazuvchi metodlardan tashkil topgan bo‘lib ulardan quyidagilarni keltiramiz:

Metod nomi	Qaysi tipga o‘tkazishi
`Convert.ToByte()`	`byte`

`Convert.ToChar()`	Belgili Char
`Convert.ToDateTime()`	Sana va vaqt
`Convert.ToDecimal()`	decimal
`Convert.ToDouble()`	double
`Convert.ToInt16()`	short
`Convert.ToInt32()`	int
`Convert.ToInt64()`	long
`Convert.ToSByte()`	sbyte
`Convert.ToSingle()`	float
`Convert.ToString()`	string
`Convert.ToUInt16()`	ushort
`Convert.ToUInt32()`	uint
`Convert.ToUInt64()`	ulong

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

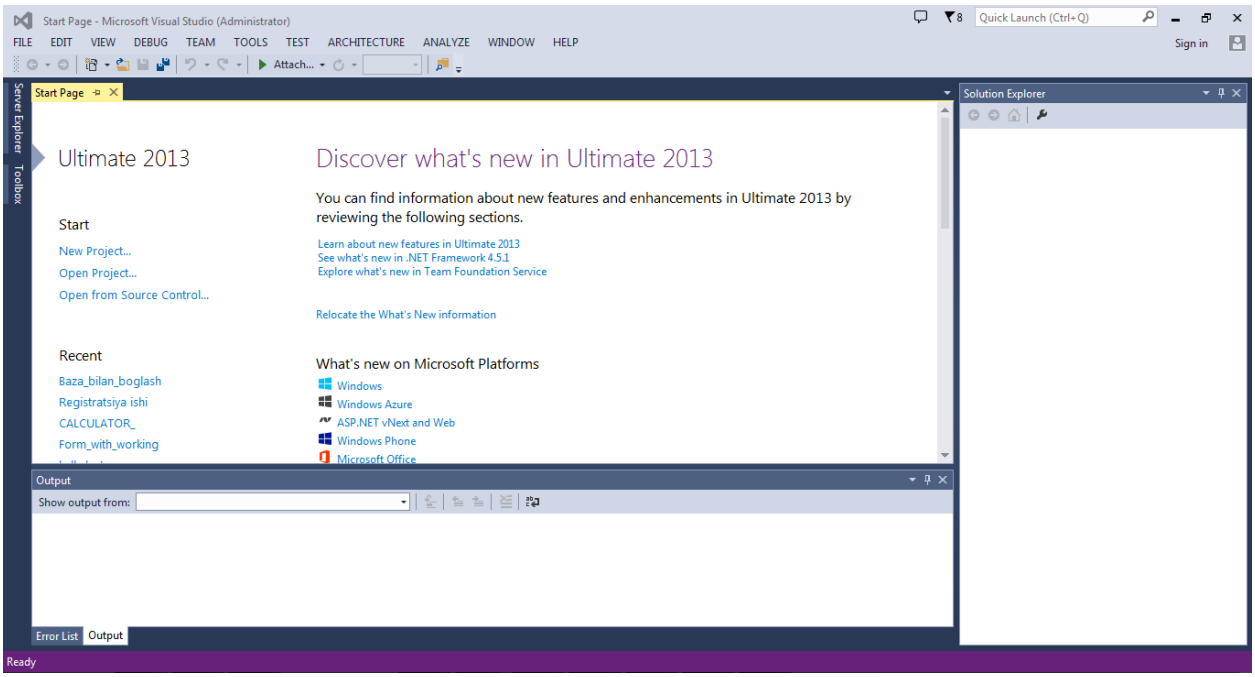
- **Operator** – jarayonki ma'lumotlar ustida bir nechta ishlarni amalga oshiradi
- **Unar (bittalik)** – faqatgina bitta operantni oladi
- **Binar (ikkilik)** – ikkita operantlarni olish

- **Ternar(uchlik) (?:)** – uchta operantlarni oladi
- **Arifmetik operatorlar** – matematikadagi amallar kabi bir xil amal bajariladi
- **Bo‘lish operatori** – (/) butun sonlar ustida ishlatiladigan bo‘lsa, butun qiymat va haqiqiy sonlar ustida ishlatiladigan bo‘lsa, haqiqiy son qiymatini qaytaradi
- **Qoldiq operatori** – (%) butunni bo‘ligandagi qoldiqni qaytaradi
- **Maxsus qo‘shish operatorida** – (++) o‘zgaruvchini qiymatini oshirishda ishlatiladi
- **Razryadli operator** – (~) barcha 0 da 1 ga va barcha 1 da 0 ga aylanadigan operatorlar
- **Kirish operatorlari** - «.» odatda ob’ektga bog‘liq holda qo‘llaniladi
- **To‘rtburchakli qavslar** – ([]) massivlarni indeksi va atributlari uchun qo‘llaniladi
- **Bir xil tipga o‘tkazish** – bir ma’lumot tipni boshqa ma’lumot tipi qiymatiga o‘tkazish
- **Oshkor ravishda tipga o‘tkazish** – o‘zgaruvchi oldiga qavs ichida boshqa tip nomi yoziladi

Amaliy mashg‘ulot

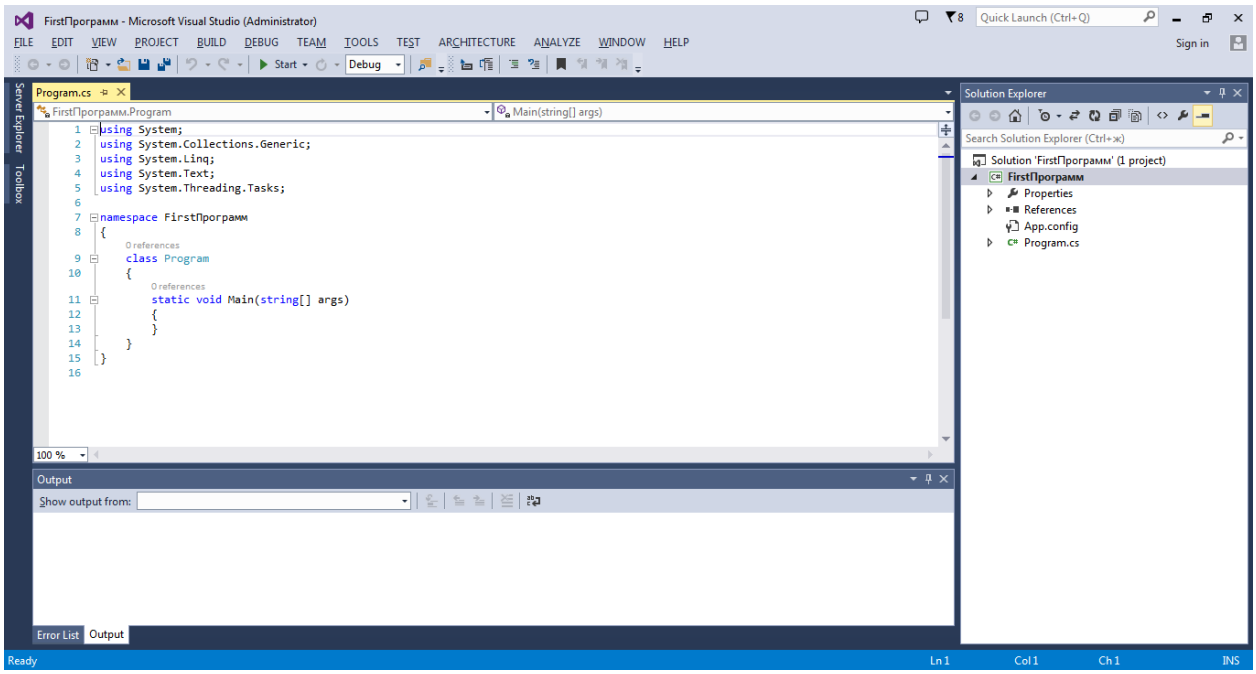
1-misol. 4 xonali butun ***n*** sonni konsoldan o‘qib oling va har bir xona birligi asosida ekranga chop eting.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **1.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



1.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **FirstProgramm** kabi kiritib, **OK** tugmasini bosamiz. Natijada **1.2 rasm**da keltirilgan quyidagi oyna shakllantiriladi.



1.2-rasm. Dasturiy kod oynasi

Endi berilgan masala kodini

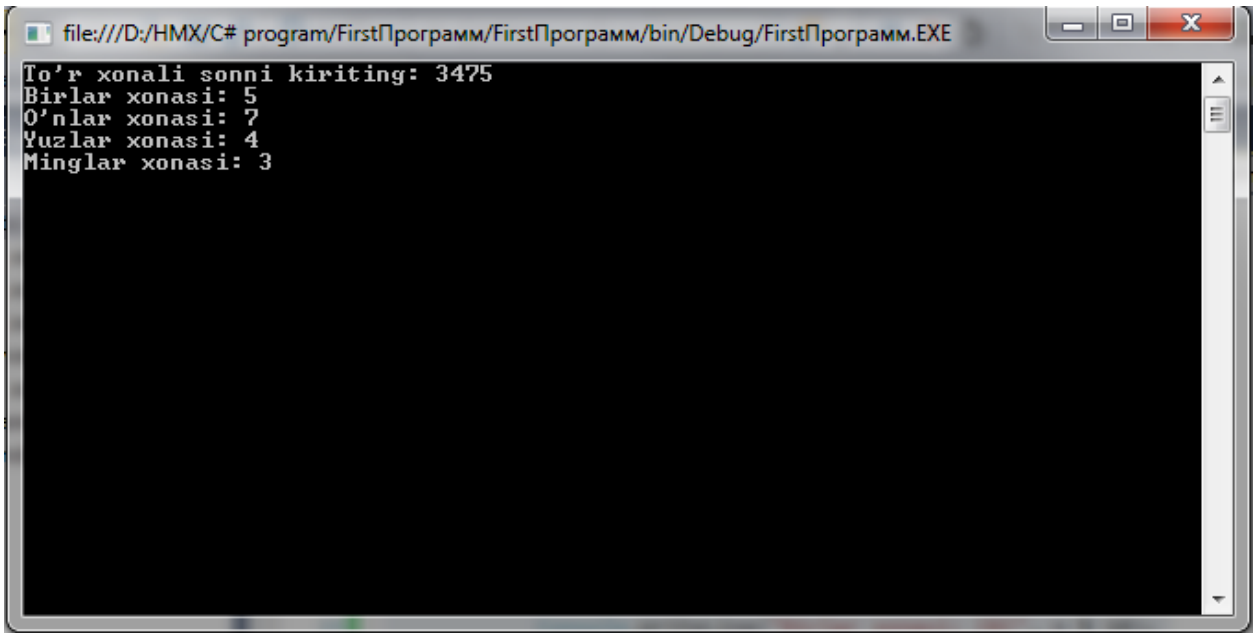
```
static void Main(string[] args)
      {
        }
```

S# dagi asosiy metod blokiga yoziladi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace FirstProgramm
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("To'r xonali sonni kiriting: ");
            int n = int.Parse(Console.ReadLine());
            Console.WriteLine("Birlar xonasi: {0}", n % 10);
            Console.WriteLine("O'nlar xonasi: {0}", (n %
100)/10);
            Console.WriteLine("Yuzlar xonasi: {0}", (n /
100)%10);
            Console.WriteLine("Minglar xonasi: {0}", n /
1000);
            Console.ReadKey();
        }
    }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. ***n*** sonini kiritamiz va 1.3-rasmda keltirilgan natijaga erishamiz.



1.3-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning

har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliyot topshiriqlari

1. Butun sonni juft yoki toqligini tekshiruvchi dastur tuzing.
2. Berilgan son ham 5 ga ham 7 ga bo‘linishini aniqlaydigan mantiqiy ifoda tuzing.
3. Kiritilgan sonning 3-raqami 7ga tengligini tekshiruvchi dastur tuzing.
4. Kiritilgan sonning necha bit ekanligini aniqlovchi dastur tuzing.
5. Berilganlarga ko‘ra uchburchak yuzini toping.
6. Foydalanuvchi tomonidan kiritilgan sonlarga ko‘ra konsolda uchburchak perimetri va yuzini hisoblovchi dastur tuzing.
7. Oy gravitatsiya doimiysi yer gravitatsiya doimiysining 17%ini tashkil etadi. Odamning yerdagi massasi oyda qancha bo‘lishini aniqlovchi dastur tuzing.(yerda g=9.81m/s2)
8. Kiritilgan ushbu juftlik {x, y} R=5 ga teng bo‘lgan aylananing markazlari bo‘lishini tekshiruvchi dastur tuzing.
9. Kiritilgan ushbu juftlik {x, y} R=5 ga teng bo‘lgan aylananing markazlari bo‘lishini tekshiruvchi va ushbu to‘g‘ri burchakli uchburchakdan [{-1, 1}, {5, 5}]. tashqaridaligini tekshiruvchi dastur tuzing.
10. Kiritigan 4 xonali abcd sonni quyidagi shartlar bo‘yicha tekshiruvchi dastur tuzing:

a. Raqamlar yig‘indisini hisoblovchi
b. Sonni ushbu tartibga o‘tkazuvchi dcba
c. Oxirgi raqamini boshiga o‘tkazuvchi dabc
d. 2- va 3- raqamlarini almashtiruvchi acbd

Testlar

1.Ifodalar operatorlarining baholash tartibini aniqlash semantik qoidalari nechinchi bo‘lib muhokama qilinadi?

a) birinchi

b) ikkinchi
c) uchinchi
d) muhokama qilinmaydi

2. Ifoda—……..

a) dasturiy tildagi hisoblarni ko‘rsatishni asosiy vositasi
b) dasturlash tillarining mohiyatini tayinlash
c) matematika, ilm-fan va muhandislikda topilgan arifmetik ifodalarni avtomatik baholash
d) to‘g‘ri javob berilmagan

3. Ko‘pchilik dasturlash tillarida, binar operatorlari infix ,ular operandni qaerida bo‘ladi?

a) o‘rtasida
b) boshida
c) bo‘lmaydi
d) ifodalar bilan birga

4. Yuqori darajadagi dasturlash tillari birinchi asosiy maqsadlari ?

a) matematika, ilm-fan va muhandislikda topilgan arifmetik ifodalarni avtomatik baholash
b) matematikada rivojlangan konvensiyalardan meros qoldirish
c) tayinlash ifodalarining dominant o‘rnini qoldirish
d) hech qanday maqsad ko‘rilmagan

5. Ko‘pchilik tillar ham yig‘ish va olib tashlash bo‘yicha bir terimli versiyalarini o‘z ichiga oladimi?

a) oladi
b) olmaydi
c) olishi ham olmasligi ham mumkin
d) ham oladi ham olmaydi

6. Funksional tillar, funksiya parametrlari kabi, qanday turdagi o‘zgaruvchilar foydalanadi?

a) har xil
b) bir xil
c) foydalanmaydi

d) funksiyalar parametrsiz bo‘ladi

7. Barcha imperativ tillarning ajralmas qismi nima?

a) dastur ijro paytida o‘zgaradigan o‘zgaruvchi tushunchasi.
b) operator va operand baholash tartibi
c) tilining bog‘liqlik va birinchilik qoidalari
d) semantik va sintaktik qoidalar

8. Qaysi ifodalar jarayoni, shu jumladan, qisqa tutashuv-baholash keyin muhokama qilinadi?

a) Relatsional va mantiqiy
b) Qayta yuklash operatorlari
c) pattern matching
d) Mantiqiy operatorlari

9. Ob’ektni baholash tartibi qanday?

a) bir tilning operator ustunlik, birlashish qoidalari tartibini aytib uning operatorlari baholash.
b) bir ifoda qiymati baholash tartibi
c) til foydalanuvchi belgilangan operatori ortiqcha yuk yo‘l oladi
d) to‘g‘ri javob berilmagan

10. Operandlar qanday ma’noni anglatadi?

a) oldin
b) ob’ekt
c) tartib
d) metod

4-Mavzu. Konsoldan kiritish va chiqarish operatorlari

Reja

Ushbu mavzuda ma’lumotlarni kiritish va chiqarish uchun konsol oynasidan foydalanamiz. Konsol oynasi va undan qanday foydaganish haqida ko‘rib chiqiladi. Matn va sonlarni chop etish va o‘qish hususiyatlarini **S#** tilida кщrib chiqiladi. Turli hil formatdagi ma’lumotlarni chop etish uchun satr formatidan foydalanamiz va **`Console.In`**, **`Console.Out`** va **`Console.Error`** foydalanib asosiy oqimni aniqlaymiz.

Kirish

Operatsion tizim dasturi yoki boshqa konsol ilovalarini foydalanuvchi bilan o‘zaro aloqa o‘rnatadigan operatsion tizim oynasi. O‘zaro aloqa standard input dan kiritish yoki standard output dan chiqarish orqali amalga oshiriladi. Bu jarayonlar kiritish chiqarish amallar sifatida nomoyon bo‘ladi. Konsolda yozilgan matn, birqancha axborot va bir qancha dasturlar tomonidan yuborilgan belgilar ketma-ketligi chiqaradi. Har bir konsol ilovalar uchun kiritish va chiqarish qurilmalarini bog‘laydi. Boshlang‘ich holda bular klavish va ekrandir ammo ular fayllar yoki boshqa qurilmalarga yo‘naltirilgan bo‘lishi mumkin.

1.1 Foydalanuvchi va dastur o‘rtasidagi aloqa

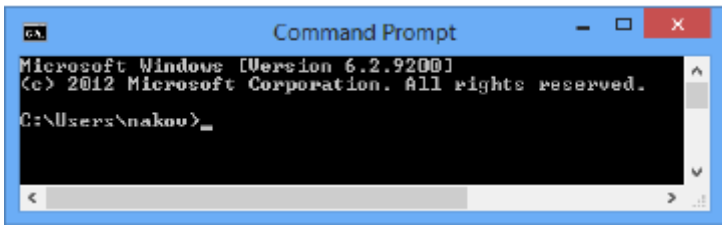
Ko‘pgina dasturlar foydalanuvchilar bilan bir qancha yo‘llar orqali aloqa qiladi. Bu ularga foydalanuvchiga qo‘llanma berish kerakligini anglatadi. Zamonaviy aloqa turlari ko‘p va turlicha. Ular grafik, vebga asoslangan interfeys, konsol yoki boshqalar bo‘lishi mumkin. Foydalanuvchi va dastur o‘rtasidagi aloqani o‘rnatadigan jihoz bu konsol rejimi hisoblanadi.

1.1.1 Konsol qanday chiqariladi

Har bir operatsion tizim konsolni chiqarishda o‘z yo‘liga ega. Misol uchun Windows da quyidagicha bajariladi:

Start -> (All) Programs -> Accessories -> Command Prompt

Yuqoridagi amallar bajarilgandan keyin, quyidagi konsol oynasi hosil bo‘ladi.



Konsol ishga tushirilganda boshlang‘ich xolatda foydalanuvchining asosiy papkasini yo‘li ko‘rsatiladi.

Konsol oynasini chiqarishda pusk tugmasini bosib va qidiruv qutisiga "cmd" yozib [Enter] tugmasini bosish orqali hosil qilishimiz mumkin(Windows Vista, Windows 7 va undan keyingi versiyalarida). Windows XP uchun quyidagi ketma-ketliklar orqali **Start -> Run… ->**, ni tanlab "**cmd**" ni yozib [Enter] tugmasini bosish orqali chiqariladi.

1.2 Windowsda konsol buyruqlari

Konsol ishlayotganda interprator buyrug‘i **"Command Prompt"** yoki **"MS-DOS Prompt"** (Windows ning eski versiyalarida) deb nomlanadi. Bu interprator uchun asosiy buyruqlarni ko‘rib chiqiladi.

Buyruqlar	**izoh**
dir	Joriy papka tarkibini kursatadi
cd <papka nomi>	Joriy papka nomini o‘zgartiradi
mkdir <papka nomi>	Joriy papkada bitta yangi papka yaratadi
rmdir <papka nomi>	Mavjud papkani o‘chirish
i <fayl nomi>	Fayl tarkibini chop etadi
copy <joriy fayl> <fayl adresi>	Bir faylni boshqa faylga nus’halaydi

Windowsda bajariladigan ko‘plab buyruqlar bir qancha misollarda keltirilgan.

```
C:\Documents and Settings\User1>cd "D:\Project2009\C# Book"

C:\Documents and Settings\User1>D:
```

```
D:\Project2008\C# Book>dir
 Volume in drive D has no label.
 Volume Serial Number is B43A-B0D6

 Directory of D:\Project2009\C# Book

26.12.2009   12:24    <DIR>          .
26.12.2009   12:24    <DIR>          ..
26.12.2009   12:23           537 600 Chapter-4-Console-Input-
Output.doc
26.12.2009   12:23    <DIR>          Test Folder
26.12.2009   12:24                 0 Test.txt
             2 File(s)        537 600 bytes
             3 Dir(s)  24 154 062 848 bytes free

D:\Project2009\C# Book>
```

1.3 Konsoldan kiritish va chiqarish qurilmalari

Ilovaga kiruvchi ma’lumot klavyaturadan tashqari ko‘pgina boshqa joylar fayl, mikrafondan ma’lumotlar kelishi mumkin. Dasturdan chiquvchi ma’lumot konsolda ekran bo‘lishi mumkin yoki boshqa chiqaruvchi qurilmalar shuningdek printer bo‘lishi mumkin:

C# dan foydalanib ma’lumotlarni standart kiritish va chiqarish yo‘llarini konsolga chop etish ko‘rsatiladi:

```
Console.Out.WriteLine("Hello World");
```

Yuqoridagi kodni natijasi quyidagicha bo‘ladi:

```
Hello World
```

1.4 Console.Out oqimi

System.Console sinfi o‘qishda foydalaniladigan va konsolda textni namoyish etadagan va uni format asosida chiqaruvchi turli hil xususiyat va metodlarga ega. Ular o‘rtasida 3 ta xususiyat mavjud va ular ma’lumotlarni kiritish va chop etish bilan bog‘lanadi. Misol uchun **Console.Out, Console.In** va **Console.Error**. Ular standart oqimdan konsolda chop etish, o‘qish, va xato xabar xaqida mos ravishda xabar beradi. Shunga qaramay biz konsole sinfining boshqa metodlaridan ham foydalanishimiz mumkin. **System.Console** bizga kiritish va chiqarish operatsiyalari bilan ishlashda bir qancha qulayliklar beradi . Aslida bu oqimlar ustida ishlash konsole funksiyasining qismidir. Biz input / output / error larni mos ravishda **Console.SetOut(...), Console.SetIn(...), Console.SetError(.)** lar bilan almashtirishimiz mumkin.

Konsolga chiqarish

Bu metodlar bilan ishlash ancha oson chunki ular barcha tipdagi ma’lumotlarni chop eta oladi. Misol uchun:

```
// chop etish string tip

Console.Writeline("Hello World");

// chop etish int tip

Console.Writeline("5");

// chop etish  double tip

Console.Writeline("3.14159265358979");
```

```
Hello World

5

3.14159265358979
```

Write(...) va **WriteLine(...)** larning farqi shundaki **WriteLine** ning ma'nosi "write line" ya'ni har birini yangi qatorga yozadi.

```
Console.Writeline(" I love ");

Console.Write(" this ");

Console.Write(" book ");
```

```
I love

This  book
```

Write metodi esa bir qatorga yozadi.

2.1 Satrlarni ulash

Umuman olganda **C#** da operatorlarni satr ustida amal bajarishga ruhsat berilmagan faqatgina "+" amalidan tashqari. Bunda u **n** satr ma'lumotlarni tartib bo'yicha qo'shadi va yangi satr sifatida qabul qiladi. Misol uchun:

```
String  yosh="yigirma bir";

String matn = "U " + yosh + " yoshda";

Console.WriteLine(matn);
```

Quyidagicha natijaga erishamiz.

```
U yigirma bir yoshda
```

2.2 Turli tiplarni qo'shish

Biz **S#** da turli tipdagi ma'lumotlarni "+" operatoridan foydalangan holda qo'shishimiz mumkin. Lekin agar satr tipi qatnashayotgan bo'lsa, yangi ma'lumot satri hosil bo'ladi.

Misol:

```
int yosh=21;

String matn ="U "+yosh +" yoshda";

 Console.WriteLine(matn);
```

Natija quyidagicha bo‘ladi.

```
U 21 yoshda
```

2.3 Satrda qo‘shishning bir qancha xususiyatlari

Bu usullarni bilish ahamiyatli va qiziqarli bo‘lishi bilan birgalikda ehtiyotkorlikni talab etadi.Misol:

```
string s="to’rt:" + 2 + 2;

Console.WriteLine(s);

//to’rt:22

string s1="to’rt:" + ( 2 + 2 );

Console.WriteLine(s1);

//to’rt:4
```

Bular juda muhim hisoblanadi. Dasturda qo‘shish chapdan o‘ngga tomon boradi. Shuning uchun biz 2- natijani olishda qavslardan foydalanamiz. Qavs qo‘yilsa dastlab qavs ichidagi amal bajariladi.

2.4 Write va WriteLine bilan formatlash (output)

Bu quyidagicha bo‘ladi.

```
String str="Hello World";

Console.Write(str);

//umumiy yo’l

//formatlangan yo’l

Console.Write("{0}",str);
```

Natija quyidagicha:

```
Hello World Hello World
```

Har ikkisi ham bir xil. Bu yerda {0} ning o‘rniga 1-argumentni qo‘yadi. Qo‘shimcha misol.

```
string  ism="Madina";
```

```
int yosh=20;
string shahar="Toshkent";
Console.Write("{0} {1} yoshda {2}dan!/n", ism, yosh, shahar);
```

Natija quyidagicha

```
Madina 20 yoshda Toshkentdan!
```

Bu yerda /n yangi qatorga ko‘chishni bildiradi.

2.5 Aralash formatlash

Ular asosan konsole ga chop etishda foydalaniladi. U 2 narsadan foydalanadi.1-si satr aralash formatlash 2-si argument seriyalari. Raqamlar uchun standart formalar:

Belgilar	**Ma’nisi**
C yoki c	Berilgan davlat uchun pul belgisini chiqarib beradi.
D yoki d	Butun son. 0 dan boshlanadigan butun sonlarni chiqarib beradi.
E yoki e	Matematikadagi e=2,7… soni chiqarib beradi.
F yoki f	Haqiqiy yoki butun sonni chiqarib beradi
N yoki n	F bilan 1 xil bir xil vazifa bajaradi faqat 1 bilan 0 lar orasiga vergul qo‘yadi
P yoki p	Foiz 100 ga bo‘lib, sonni foizini chiqarib beradi
X yoki x	Sonni 16 lik sanoq sistemasiga o‘tkazadi

Misollar ko‘rib chiqamiz:

```
class StandardNumericFormats
{
   static void Main()
   {
      Console.WriteLine("{0:C2}", 123.456);
      //Output: 123,46 лв.
      Console.WriteLine("{0:D6}", -1234);
      //Output: -001234
      Console.WriteLine("{0:E2}", 123);
      //Output: 1,23E+002
      Console.WriteLine("{0:F2}", -123.456);
      //Output: -123,46
```

Yuqoridagi kod natijasi quyidagicha bo‘ladi:

```
      Console.WriteLine("{0:N2}", 1234567.8);
      //Output: 1 234 567,80
      Console.WriteLine("{0:P}", 0.456);
      //Output: 45,60 %
      Console.WriteLine("{0:X}", 254);
      //Output: FE
   }
}
```

2.6 Sonlar uchun odatiy(formal bщlmagan) formalar

Quyidagi formatlar foydalanuvchilar tomonidan odatiy tusga aylangan lekin bu

```
      Console.WriteLine("{0:N2}", 1234567.8);
      //Output: 1 234 567,80
      Console.WriteLine("{0:P}", 0.456);
      //Output: 45,60 %
      Console.WriteLine("{0:X}", 254);
      //Output: FE
   }
}
```

formatlar S# ning standart formatlari emas. Standart formatlardan farqi shundaki bu formatlar belgilardan tashkil topgan:

Belgilar	Ma’nosi
0	Raqamni ifodalaydi. Agar raqam tushib

	qolsa 0 yoziladi
#	Raqamni ifodalaydi. Agar bu belgi yozilsa hech nima chop etmaydi.
.	O‘nlar xonasigacha yaxlitlab beradi
,	Minglar xonasigacha yaxlitlab beradi
%	Foizni chiqaradi. Natijani 100 ga ko‘paytirib, foizda chiqarib beradi
E0, E+0 va E-0	E0 va Ye+0 exponentani beradi Ye-0 ye sonini qaramma – qarshi qiymati

Odatiy formatlarni hammasi keltirilmagan. Ular haqidagi barcha ma’lumotlarni **MSDN**. Quyidagi misollar formatlardan foydalanish yo‘llarini ko‘rsatadi:

```
class CustomNumericFormats
{
  static void Main()
  {
     Console.WriteLine("{0:0.00}", 1);
     //Output: 1.00
     Console.WriteLine("{0:#.##}", 0.234);
     //Output: .23
     Console.WriteLine("{0:#####}", 12345.67);
     //Output: 12346
     Console.WriteLine("{0:(0#) ### ## ##}", 29342525);
     //Output: (02) 934 25 25
     Console.WriteLine("{0:%##}", 0.234);
     //Output: %23
  }
}
```

2.7 Sana uchun standart satr formalar

Sana formatlaridan faydalanganimizda standart va odatiy formatlarni ajratamiz.

2.7.1 Sana formatlarining standart shakli

Bu formatlar juda ko‘p biri qanchasi keltirildi va qolganlarini **MSDN** da tekshirib olinidi.

Belgilar	**Ma’nosi**

D	20/05/2016
D	May 20, 2016
T	13:16 (soat)
T	13:16:15 (soat)
Y yoki y	May 2016

2.7.2 Odatiy sana formatlari

Sananing odatiy formatlari sonlarning odatiy formatlariga o‘xshash va ularda ikkilangan format belgilar ham bor. Quyidagi jadvalda sana formatlari qanday ishlatilishi ko‘rsatib o‘tilgan:

Belgilar	Format ma’nosi
d	1 dan 31 gacha kunlar
dd	01 dan 31 gacha kunlar
M	1 dan 12 gacha oylar
MM	01 dan 31 gacha oylar
yy	Yilning oxirgi ikki raqami (00 dan 99 gacha)
yyyy	4 ta raqam yozilgan yil
hh	00 dan 11 gacha bo‘lgan soat
HH	00 dan 23 gacha bo‘lgan soat
M	0 dan 59 gacha bo‘lgan minut
Mm	00 dan 59 gacha bo‘lgan minut
S	0 dan 59 gacha bo‘lgan soniya
SS	00 dan 59 gacha sekund

Belgilarni ishlatganda ularni ajratish uchun 2 ta belgidan ya'ni "." yoki "/" lardan foydalaniladi.

```
DateTime d = new DateTime(2012, 02, 27, 17, 30, 22);
Console.WriteLine("{0:dd/MM/yyyy HH:mm:ss}", d);
Console.WriteLine("{0:d.MM.yy}", d);
```

Yuqoridagi kod natijasi:

```
27/02/2012 17:30:22
27.02.12
```

```
27.02.2012 17:30:22
27.02.12
```

Konsoldan kiritish

Kiritish standartlari dastur qaerdan kiritilgan ma'luotni qabul qilishini tekshirib turuvchi operatsion tizimning qismi hisoblanadi. Klavish orqali kiritilgan ma'lumotni avtomatik ravishda o'qiydi. Agar kiritgan ma'lumotimiz o'zgarsa, standart kiritish yana qayta ma'lumotlarni o'zgartiradi. Har bir dasturlash tilida konsolda o'qish va yozish uchun shunday mehanizm bor. Console.In. orqali ob'ekt C# dagi standart kiritish oqimini tekshiradi.

Konsoldan turli hil ma'lumotlarni o'qib olishimiz mumkin:

Text

Textni o'tkazganimizdan keyingi boshqa tiplar odatda **`Console.In.`** orqali kiritish kam ishlatiladi. Kolsol klassi kiritilgan ma'lumotni o'qishda 2 ta metodddan foydalaniladi: `Console.Read()`va `Console.ReadLine()`

3.1 Console.ReadLine() orqali o'qish

`Console.ReadLine()` metodi konsoldan o'qishda juda katta qulaylik yaratadi. U qanday ishlaydi? Qachon bu metod chaqirilganda u konsoldan ma'lumot kiritilishini kutib turadi. Foydalanuvchi bir qancha satrlarni kiritadi va **[Enter]** tugmasini bosadi. Shu vaqtda konsol foydalanuvchi kiritishni tugatganini tushinadi va satrni o'qishni boshlaydi. Foydalanuvchi tomonidan kiritilgan satrni `Console.ReadLine()` metodi natija sifatida qaytaradi. `Console.ReadLine()` deb nom qo'yilish sababi ham shundadir.

Console.ReadLine() ga misol ko‘rib chiqiladi:

```
class UsingReadLine
{
   static void Main()
   {
      Console.Write("Please enter your first name: ");
      string firstName = Console.ReadLine();

      Console.Write("Please enter your last name: ");
      string lastName = Console.ReadLine();

      Console.WriteLine("Hello, {0} {1}!", firstName, lastName);
   }
}

// Output:  Please enter your first name: John
//          Please enter your last name: Smith
//          Hello, John Smith!
```

Console.ReadLine() metodida bajarilgan natijani tahlil qilinadigan bo‘lsa, foydalanuvchi tomonidan so‘ralgan ma’lumot konsol orqali kiritiladi. **[Enter]** bosilguncha bo‘lgan ma’lumotni **ReadLine()** metodi o‘qiydi. Bu jarayon 2-so‘rovda ham takrorlanadi. Kerakli ma’lumot yig‘ilganidan so‘ng u konsolda chop etiladi.

3.2 Console.Read() metodi orqali o‘qish

Read() metodi va **ReadLine()** metodidan ozgina farq qiladi. Aytish joizki u bir belgini o‘qiydi, bo‘sh joygacha emas. Boshqa bir muhim farq shuki kod bo‘lsa ham javob qaytarmaydi. Agar biz natijadan o‘zgaruvchi sifatida foydalanmoqchi bo‘lsak biz uni belgiga almashtirishimiz kerak yoki **Convert.ToChar()** metodidan foydalanamiz. Bu yerda juda muhim xususiyat bor: belgi **[Enter]** tugmasi bosilguncha o‘qiladi. Konsoldan kiritilgan satr, satr kiritish standarti bufferiga ko‘chiriladi va **Read()** metodi undagi birinchi belgini o‘qiydi. Agar buffer bo‘sh bo‘lmasa metodning keyingi so‘rovlari uchun dastur natijasi to‘xtamasdan keyingi belgini buffer bo‘sh bo‘lguncha o‘qishni davom etadi. Bundan keyin foydalanuvchi **Read()** metodini kiritmaguncha dastur kutib turadi. Misol.

```
class UsingRead

{

  static void Main()
```

```
    {
      int codeRead = 0;
      do
      {
        codeRead = Console.Read();
        if (codeRead != 0)
        {
          Console.Write((char)codeRead);
        }
      }
      while (codeRead != 10);
    }
  }
```

Bu dastur foydalanuvchi tomonidan kiritilgan 1-qatorni o‘qiydi va uni belgi sifatida chop etadi. Biz shunga ahamiyat berishimiz kerakki **Console.Read()** metodi **Console.ReadLine()** ga qaraganda amaliyotda kam qo‘llaniladi.bunga sabab **Console.Read()** metodi bilan xatolar qilish ehtimoli mavjudligidadir.

3.3 Sonlarni o‘qish

S# da sonlarni konsoldan to‘gridan to‘gri o‘qish hali mavjud emas.sonni o‘qishimiz uchun biz kiritilgan sonni Konsole.ReadLine metodi orqali satr tipida o‘qib olamiz va keyin uni son tipiga o‘tkazamiz.bu jarayon parsing deb ataladi.quyidagi misol sonni o‘qib olishga va uni parse (o‘tkazish)ga berilgan.

```
class ReadingNumbers
{
```

```
        static void Main()
        {
            Console.Write("a = ");
            int a = int.Parse(Console.ReadLine());

            Console.Write("b = ");
            int b = int.Parse(Console.ReadLine());

            Console.WriteLine("{0} + {1} = {2}", a, b, a + b);
            Console.WriteLine("{0} * {1} = {2}", a, b, a * b);

            Console.Write("f = ");
            double f = double.Parse(Console.ReadLine());
            Console.WriteLine("{0} * {1} / {2} = {3}",
                a, b, f, a * b / f);
        }
    }
```

Natijasi:

```
a = 5
b = 6
5 + 6 = 11
5 * 6 = 30
f = 7.5
5 * 6 / 7.5 = 4
```

Bu misolda muhim narsa shuki biz **Parse** metodini sonlar uchun qo‘lladik. Qachonki biz xato natija olsak natija **System.FormatException** ga yuboriladi. Bunday

jarayon haqiqiy sonlarni o‘qishda bo‘ladi. Chunki sonlar orasida chegara mavjudligi va har bir kichik to‘plam uchun operatsion tizimda alohida muhit mavjudligidir.

3.4 Console.ReadKey() orqali o‘qish

Console.ReadKey() metodi konsoldan belgilarni kiritilishini kutib turadi va **[Enter]** tugmasini bosilishini kutmasdan o‘qiy boshlaydi. **ReadKey()** dan so‘ralgan natija bosilgan tugma ma’lumoti bo‘lib u **ConsoleKeyInfo.** tipiga tegishlidir.

Quyida misol

```
ConsoleKeyInfo key = Console.ReadKey();

Console.WriteLine();

Console.WriteLine("Character entered: " + key.KeyChar);

Console.WriteLine("Special keys: " + key.Modifiers);
```

Natija:

```
A
Character entered: A
Special keys: Shift
```

2.5 Sonlarni Nakov.IO.Cin orqali oson o‘qib olish

Sonlarni osongina o‘qib olishning alohida standarti mavjud emas. **S#** va **.Net Framewor**kda biz satrlarni o‘qishga ehtiyoj sezamiz, uni belgilarga bo‘lishda bo‘sh joylardan foydalaniladi boshqa tillar va platformalarda **S++** da sonlar belgilar va matnlarni parse qilmasdan o‘qiy olamiz. Lekin **S#** da qo‘shimcha kutubxona va klass lardan foydalanamiz.

```
using Nakov.IO;

…

int x = Cin.NextInt();

double y = Cin.NextDouble();

decimal d = Cin.NextDecimal();

Console.WriteLine("Result: {0} {1} {2}", x, y,
d);
```

Quyidagicha natijaga erishamiz:

```
3   2.5
 3.58
Result: 3 2.5 3.58
```

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

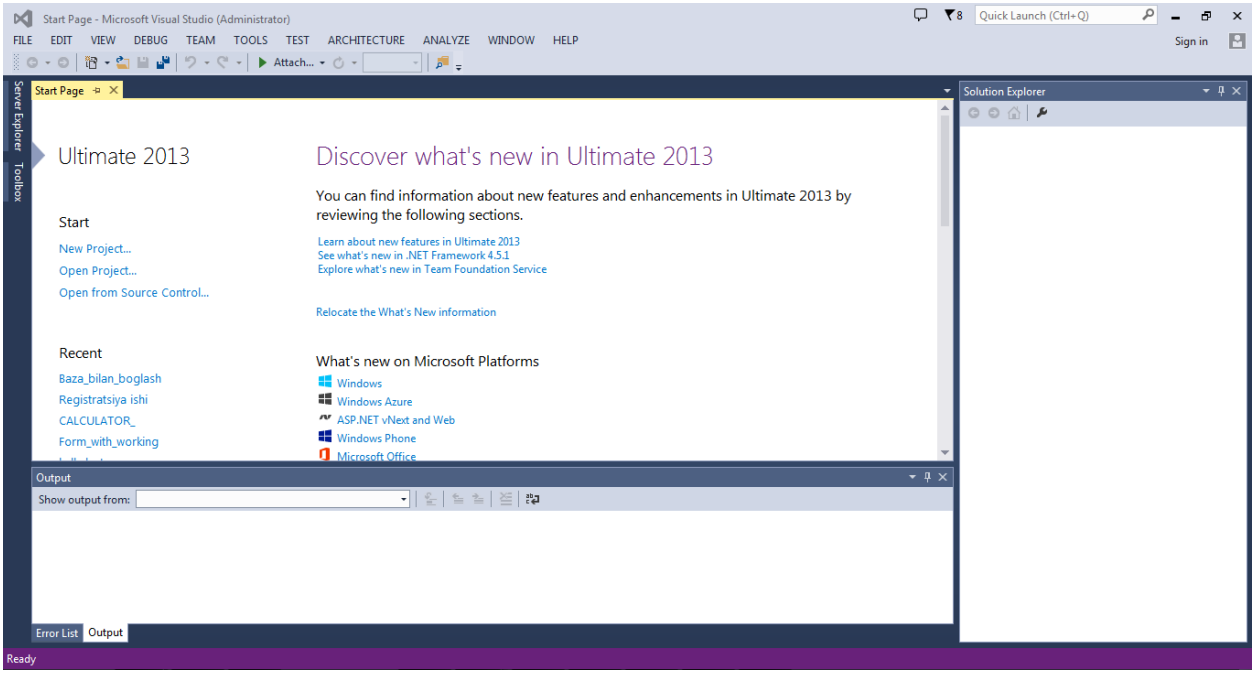
Glossariy

- **Konsol** – oynada ma'lumotlar va dastur natijasini ko'rish uchun ishlatiladi
- **System.Console** - **Console** sinfi yordamida konsolga chiqarish
- **Read(…)** – bitta belgini o'qish
- **ReadKey(…)** – klavish kombinatsiyasini o'qish
- **ReadLine(…)** – belgilarni bitta qatorini o'qish
- **Write(…)** – konsolga argumentlarni chiqarish
- **WriteLine(…)** – konsolga ma'lumotlarni chiqarish va keyingi qatorga o'tish
- **Console.ReadKey()** – klavish bosilguncha ekranni ushlab turadi
- **KeyChar** – kiritilgan belgini ushlab turadi
- **Modifiers** – **[Ctrl], [Alt],** … xoltini ushlab turadi
- **int.Parse(string)** – satrni butun tipga o'tkazadi
- **Convert klassi** – metodidan foydalangan holda boshqa tipga o'tkazish uchun ishlatiladi
- **try-catch** – hatoliklarni ushlab qolish uchun ishlatiladi
- **TryParse()** – hatoliklarni ushlab qolish uchun ishlatiladi

Amaliy mashg'ulot

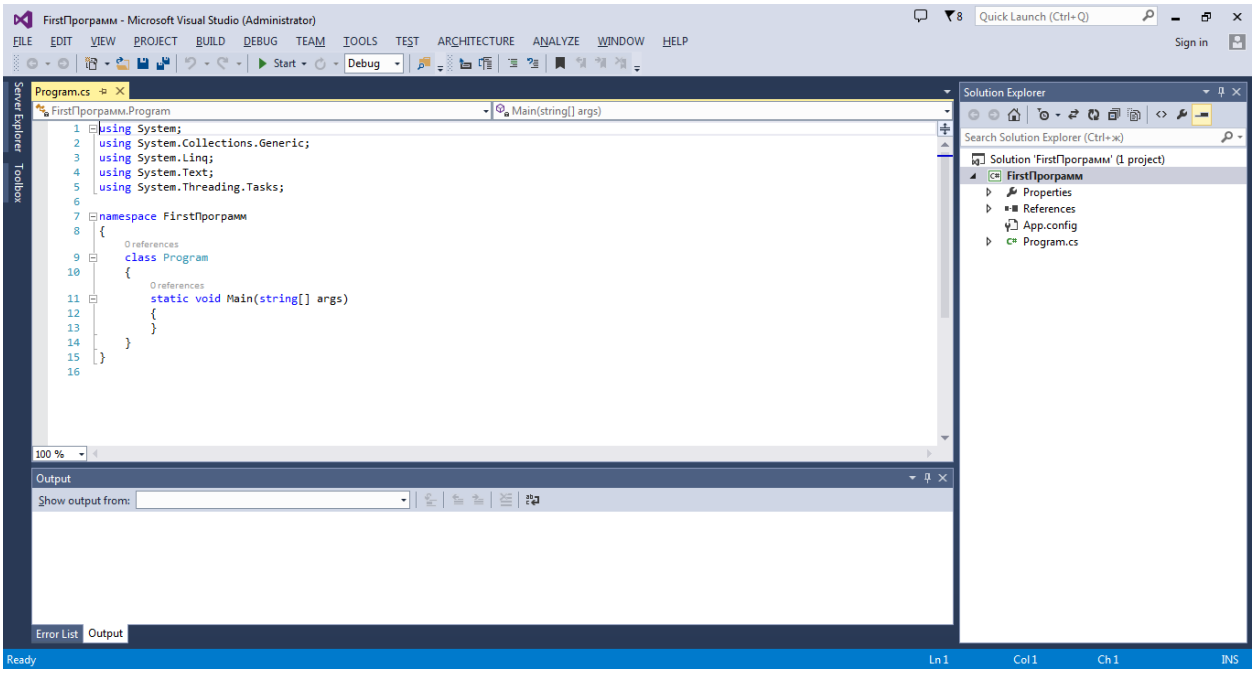
- **1-misol.** 4 xonali butun ***n*** sonni konsoldan o'qib oling va har bir xona birligi asosida ekranga chop eting.

- **Visual Studio 2013 (VS 2013)** muhiti o‘rnatilgach, tizim ishga tushiriladi, **1.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



- **1.1-rasm. Visual Studio 2013** tizimining boshlang‘ich sahifasi

- **VS 2012** muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **FirstProgramm** kabi kiritib, **OK** tugmasini bosamiz. Natijada **1.2 rasm**da keltirilgan quyidagi oyna shakllantiriladi.



- **1.2-rasm.** Dasturiy kod oynasi

- Endi berilgan masala kodini

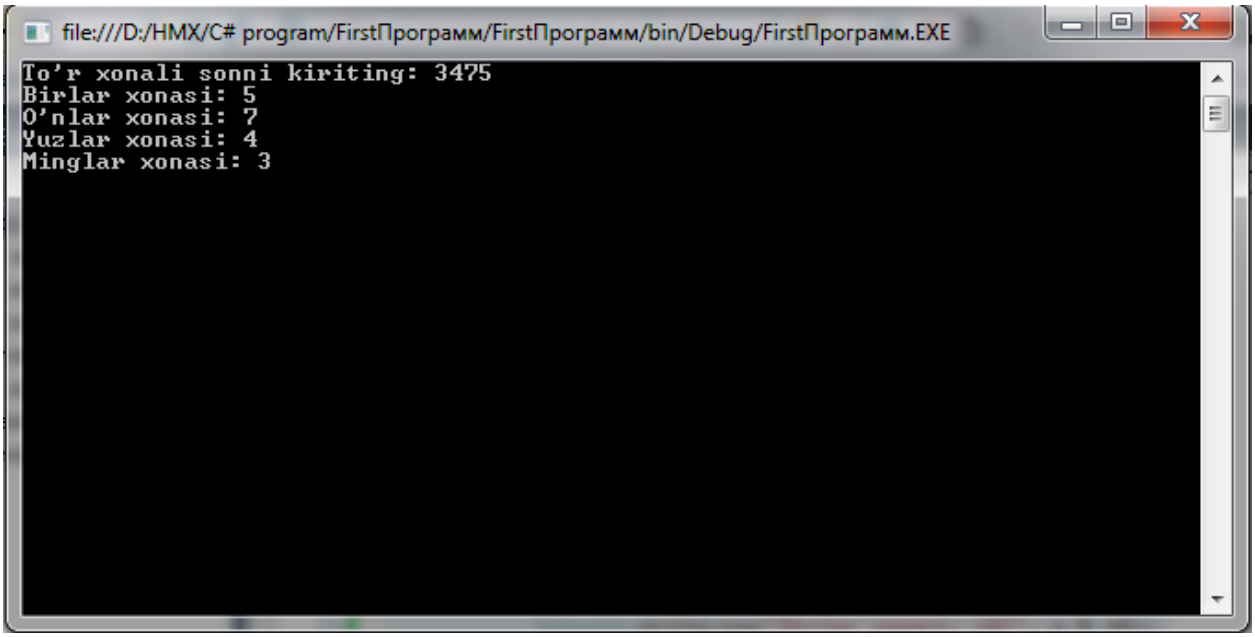
```
static void Main(string[] args)
        {
        }
```

- S# dagi asosiy metod blokiga yoziladi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("To'r xonali sonni kiriting:
");
            int n = int.Parse(Console.ReadLine());
            Console.WriteLine("Birlar xonasi: {0}", n %
10);
```

```
            Console.WriteLine("O'nlar xonasi: {0}", (n % 100)/10);
            Console.WriteLine("Yuzlar xonasi: {0}", (n / 100)%10);
            Console.WriteLine("Minglar xonasi: {0}", n / 1000);
            Console.ReadKey();
        }
    }
}
```

- Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. ***n*** sonini kiritamiz va 1.3-rasmda keltirilgan natijaga erishamiz.



1.3-rasm. Konsol oynasi.

- Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliyot topshiriqlari

2. Konsolda ismingiz va familiyangizni chop qiluvchi dastur tuzing.

3. Konsolda 1, 101, 1001 larni har birini yangi qatordan chop etuvchi dastur tuzing.
4. Konsolda bugungi kun va vaqtni chiqaruvchi dastur tuzing.
5. Konsoldan yoshingizni o‘qib olib 10 yildan keying yoshingizni chop etuvchi dastur tuzing.
6. int tipida 2 ta o‘zgaruvchi e’lon qiling. Ularning qiymatlarini mos ravishda 5 va 10 ga teng. Ularni almashtirib chop qiluvchi dastur tuzing.
7. Quyidagi qiymatlarning mos tiplarini aniqlang:
   sbyte, byte, short, ushort, int, uint, long va **ulong** 52,130; -115;4825932; 97; -10000; 20000; 224; 970,700,000; 112; -44; -1,000,000; 1990; 123456789123456789.
8. Quyidagi qiymatlar qaysi tipga tegishli:
   float, double va **decimal**
   5, -5.01, 34.567839023; 12.345; 8923.1234857; 3456.09112487595654215125668346 7?
9. 2 ta haqiqiy sonni 0.000001 aniqlikda taqqoslaydigan dastur tuzing.
10. **int** tipidagi 256 sonini 16 lik sanoq sistemasiga o‘tkazuvchi dastur tuzing.
11. Marketing bilan shug‘ullanuvchi kompaniya o‘z xodimlaridan ma’lumotlarni yozishini xoxladi. Ushbu ma’lumotnomada ismi, familiyasi, yoshi, jinsi, alohida kartasi (27560000 dan 27569999shungacha) bo‘lishi kerak. O‘zgaruvchilar e’lon qilgan holda har bir xodim uchun yarating.
12. 3 ta sonli tiplarni konsoldan o‘quvchi va ularni qiymatini chiqaruvchi dastur tuzing.
13. Aylanani **r** radusini konsoldan o‘qib primetri va yuzasini chop etuvchi dastur tuzing.
14. Kompaniyani nomi, adresi, telefon nomeri, faks nomeri, veb sayti va boshqaruvchisi bor. Boshqaruvchisining ismi, familyasi, yoshi, va telefon nomeri bor. ma’lumotlarni konsoldan o‘qib oluvchi va chiqaruvchi dastur tuzing.
15. 2 ta musbat butun sonni ekrandan o‘qib oling va shu sonlar orasidagi sonlar p soniga teng ular orasida 5 ga bo‘lganda qoldiq 0 qoladigan sonlarni nechta ekanini aniqlovchi dastur tuzing.
    Masalan: p(17,25) = 2.
16. Konsoldan 2 ta sonni o‘qib oling va ularni kattasini chop eting. **if** operatoridan foydalanmang.

17. $ax^2+bx+c=0$ kvadrat tenglamaning koeffitsentlari a, b va c konsoldan o‘qib olinsin va kvadrat tenglamaning yechimlari (haqiqiy son) chop etilsin.
18. **n** sonini ekran orqali kiriting va yana **n** ta son kiriting va ularni yig‘indisini hisoblang.
19. **n** butun sonni konsoldan o‘qing va **[1..n]** oralig‘dagi barcha sonlarni har birini alohida qatorga chop etish dasturini yozing.
20. Fibbonachi ketma-ketligini birinchi 100 ta elementini chop etish dasturini tuzing: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, …
21. Quyidagi summani hisoblash dasturini yozing (0.001 aniqlikda):
    1 + 1/2 - 1/3 + 1/4 - 1/5 + ...

Testlar

1. Konsol oynasi nima vazifalarni bajaradi?

a) oynada ma’lumotlarni ko‘rish uchun ishlatiladi
b) oynaga ma’lumotlarni chiqarish uchun ishlatiladi
c) oynadan ma’lumotlarni o‘qib olish uchun ishlatiladi
d) berilgan javoblarni barchasi to‘g‘ri

2. Qaysi sinfi yordamida konsolga chiqariladi?

a) **System.Console**
b) **ConsoleKeyInfo**
c) **int.Parse(string)**
d) **float.Parse(string)**

3. Konsoldan belgilar ketma – ketligini o‘qish uchun qaysi metoddan foydalaniladi?

a) **WriteLine(…)**
b) **ReadKey(…)**
c) **ReadLine(…)**
d) to‘g‘ri javob berilmagan

4. Konsolga yozish qaysi metoddan foydalaniladi?

a) **WriteLine(…)**
b) **ReadKey(…)**
c) **ReadLine(…)**
d) **Read()**

5. Console.WriteLine(…) metodi nima vazifni bajaradi?

a) konsolga argumentlarni chiqarish

b) konsolga ma’lumotlarni chiqarish va keyingi qatorga o‘tish
c) klavish kombinatsiyasini o‘qish
d) bitta belgini o‘qish

6. Console.Write(…) va Console.WriteLine(…) metodlari orasida qanday farq bor.

a) ikkisi ham bir hil vazifani ba
b) tasdiklar
c) matinlar
d) satrlar

7. dir buyrug‘i nima vazifasini bajaradi?

a) Joriy papka nomini o‘zgartiradi
b) Joriy papkada bitta yangi papka yaratadi
c) Joriy papka tarkibini kursatadi
d) Fayl tarkibini chop etadi

8. Konsolda standart oqimlarni ko‘rsating?

a) Console.Out
b) Console.In
c) Console.Error
d) Berilgan javoblarni barchasi to‘g‘ri

9. Satrlar bilan ishlashda qaysi amalni ishlatishga ruhsat berilgan?

a) +
b) -
c) /
d) *

10./n nima vazifani bajaradi?

a) yangi qatorga ko‘chishni
b) satrlarni joyini almashtirishni
c) 2 ta satr orasida joy ajratishni
d) to‘g‘ri javob berilmagan

5-maruza. Shart operatorlari

Reja

1. Taqqoslash operatorlari va mantiqiy ifodalar
2. Mantiqiy operatorlar

3. “if” va “if – else” shart operatorlari
4. “if” strukturasini ichma – ich joylashishi
5. “if – else –if – else …”operatorining ketma – ketligi
6. “switch-case” tanlash operatori

Bu mavzuda **C#**da shartli operatorlar haqida ma’lumot beriladi. Biz shartli **if** va **else** operatorlarining sintaksisini va tanlash **switch-case** operatori tushuntiramiz. Biz quyida shart operatorlarini ishlatganda amaliy misollar bilan ko‘rsatib beramiz.

1. Taqqoslash operatorlari va mantiqiy ifodalar

Navbatdagi bo‘limda biz **C#** tilidagi asosiy taqqoslash operatorlarini esga olamiz. Ular muhim chunki biz ulardan shartli operatorlarda ularning shartini tasvirlashda foydalanamiz.

1.1 Taqqoslash operatorlari

C#da bir qancha taqqoslash operatorlari bor, ular asosan butun sonlar juftini, haqiqiy sonlarni, belgilarni, satrlarni va boshqa tiplarni taqqoslashda ishlatiladi:

Operator	**Ma’nosi**
==	Tenglik
!=	teng emas
>	qat’iy katta
>=	katta yoki teng
<	qat’iy kichik
<=	kichik yoki teng

Taqqoslash operatorlaridan ikkita sonlarni, ikkita raqamli ifodalarni yoki raqamli o‘zgaruvchilarni taqqoslashda foydalanamiz. Taqqoslash natijasi mantiqiy qiymat (**true** yoki **false**) bo‘ladi. C#da taqqoslash operatorga oid misol ko‘rib chiqamiz:

```
int weight = 700;
Console.WriteLine(weight >= 500); // True

char gender = 'm';
Console.WriteLine(gender <= 'f'); // False

double colorWaveLength = 1.630;
Console.WriteLine(colorWaveLength > 1.621); // True

int a = 5;
int b = 7;
bool condition = (b > a) && (a + b < a * b);
Console.WriteLine(condition); // True

Console.WriteLine('B' == 'A' + 1); // True
```

Bu namunada ko‘rsatilgan taqqoslashdagi dastur sonlar va belgilar o‘rtasida. Bu misolda sonlar o‘zlarining o‘lchami bo‘yicha taqqoslanyapti. Belgilar taqqoslanganda esa ularning **Unicode** da turgan tartibiga qaralyapti. Misolda ko‘rinib turibdiki **char** tipi o‘zini xuddi sonlardek tutadi. Biz bu tipni sonlar bilan istalgan amalimizni (**‘+’** va **“-”**) bajarishimiz mumkin va bir-biri bilan taqqoslay olamiz. Vaholanki bunday dastur kodini qiyinligi o‘qish va tushunish murakkab bo‘lsa ham foydalaniladi.

Misolning natijasi quyidagicha:

```
True
False
True
True
True
```

C#da quyidagi ma’lumot tiplari taqqoslanishi mumkin:

- ***sonlar*** *(int, long, float, double, ushort, decimal…)*
- ***belgilar*** *(char)*
- ***mantiqiy*** *(bool)*

- ***qiymatlarga havolalar, qiymatlar ko‘rsatkichlari*** *(string, object, massiv va boshqalar)*

Har doim ikkita sonlarni o‘zaro ta’sirini, ikkita mantiqiy qiymatlarni yoki ikkita qiymat havolalarini taqqoslaymiz. Turli xil tipdaga ifodalarni taqqoslay olamiz va misol uchun xaqiqiy sonlar bilan butun sonlarni. Ammo har birini ma’lumotlar tiplarini to‘g‘ridan – to‘g‘ri taqqoslay olmaymiz. Misol uchun sonlar bilan satrlarni taqqoslay olmaymiz.

1.2 Belgilar va sonlarni taqqoslash

Qachonki sonlar va belgilarni taqqoslayotganda to‘g‘ridan-to‘g‘ri biz ularning xotiradagi ikkilik sanoq sistemasidagi ko‘rinishlarini taqqoslaymiz. Misol uchun agar biz butun tipdagi ikki sonni taqqoslaganda biz har birining 4 baytdagi qiymatini taqqoslaymiz. Quyidagi misolda butun sonlar va belgilar taqqoslangan:

```
Console.WriteLine("char 'a' == 'a'? " + ('a' == 'a'));  // True
Console.WriteLine("char 'a' == 'b'? " + ('a' == 'b'));  // False
Console.WriteLine("5 != 6? " + (5 != 6));               // True
Console.WriteLine("5.0 == 5L? " + (5.0 == 5L));         // True
Console.WriteLine("true == false? " + (true == false)); // False
```

Misolning natijasi:

```
char 'a' == 'a'? True
char 'a' == 'b'? False
5 != 6? True
5.0 == 5L? True
true == false? False
```

1.3 O‘zgaruvchilarning xotiradagi joyi bo‘yicha taqqoslash

.NET FRAMEWORKda o‘zgaruvchi ma’lumot tiplarining qiymatlari saqlanmaydi. Lekin xotiradagi manzili va qancha joy egallaganligi haqida ma’lumot turadi. Satr, massiv va klass tiplari shular jumlasidandir. Ular o‘zlarini qiymat bor yoki yo‘q **(null)**dek tutadi. Agar biz o‘zgaruvchilart tipi haqidagi ma’lumotlarni taqqoslaganimizda, ular egallagan joyni tekshiramiz. Tekshirish natijasi bir xil bo‘lsa, demak ular bir narsadir. Bunda ularda 3 xil ko‘rinishda ya’ni, ular bir qiymatli, turli qiymatli yoki ulardan biri bo‘sh **(null)** bo‘lishi mumkin.

Navbatdagi misolda ikkita bir xil qiymat ega bo‘lgan o‘zgaruvchi e’lon qildik.

```
string str = "beer";
string anotherStr = str;
```

Yuqoridagi bajarilgan kod natijasi shuni ko‘rsatadiki ikki **str** va **anotherStr** o‘zgaruvchilar bir xil narsa (satr tipli qiymati **“beer”**) va bosh xotirada bir xil joy egallaydi. Biz o‘zgaruvchilar qiymatlarining bir xil yoki bir xil emasligini **(==)** operatori orqali tekshiramiz. Ko‘plab ma’lumotlar tiplarida bu operator ularni qiymatlarini emas balki xotirada bir xil joy egallaganmi yo‘qmi ekanini tekshiradi. Ularni taqqoslaganda quyidagi amallardan foydalanamiz (**< , > , < =** va **>=**).

Quyidagi misolda o‘zgaruvchi qiymatlarining xotiradagi joyini taqqoslash ko‘rsatilgan:

```
string str = "beer";
string anotherStr = str;
string thirdStr = "be" + 'e' + 'r';
Console.WriteLine("str = {0}", str);
Console.WriteLine("anotherStr = {0}", anotherStr);
Console.WriteLine("thirdStr = {0}", thirdStr);
Console.WriteLine(str == anotherStr); // True - same object
Console.WriteLine(str == thirdStr); // True - equal objects
Console.WriteLine((object)str == (object)anotherStr); // True
Console.WriteLine((object)str == (object)thirdStr); // False
```

Bu misolni bajarganimizdan keyin, quyidagi natijani olamiz:

```
str = beer
anotherStr = beer
thirdStr = beer
True
True
True
False
```

Chunki bu misolda satr tipidan foydalanilgan (**C#** dasturlash tilida satr **string** kalit so‘zi orqali belgilanadi u **System.String** klassining ichida joylashgan), ularning qiymatlari xotiradan ob’ekt sifatida joy olgan. Bu ikki o‘zgaruvchi **str** va **thirdStr** ning qiymatlari teng, ammo turli o‘zgaruvchilar va xotirada joylashgan adresi turli xil. **anotherStr** o‘zgaruvchisining adresi **str** ning adresiniga murojaat qiladi va **str** o‘zgaruvchisiga teng. **str** va **anotherStr** o‘zgaruvchilarini taqqoslash orqali ular teng va bir xilligini ko‘ramiz. Ikkita **str** va **anotherStr** ni taqqoslash natijasi ularning tengligini ko‘rsatadi, chunki **==** operatori satr tipdagi o‘zgaruvchilarni qiymatlari orqali taqqoslaydi ularni adreslar orqali emas (adreslar orqali taqqoslash qoidasiga

istisno holat). Agar biz uchta o‘zgaruvchining qiymatlarini o‘zgartirsak va keyin ularni taqqoslasak, ularning qiymatlari joylashgan addreslarni taqqoslaymiz va ularning natijasi turlicha bo‘ladi.

Yuqoridagi misol shuni ko‘rsatadiki **==** operatorida satrlarni taqqoslash uchun maxsus xususiyatga ega, ammo qolgan tiplar xotiradagi joylashgan adresi bo‘yicha taqqoslanadi.

2. Mantiqiy operatorlar

S# dagi mantiqiy operatorlarni esga oladigan bo‘lsak, ular ko‘pincha mantiqiy ifodalarni qurishda ishlatiladi. Ular quyidagilar: **&&, ||, !, ^** lar kiradi.

2.1. && va || mantiqiy operatorlari

&& (va) va **|| (yoki)** mantiqiy operatorlar ko‘pincha mantiqiy tipdagi ifodalar uchun foydalaniladi (qiymatlar **bool** tipida). **&&** operator orqali ikkita ifodani taqqoslashda olingan natija – **true (**to‘g‘ri**)** bo‘ladi qachonki, ikkala o‘zgaruvchi qiymat ham bir vaqtda **true (**to‘g‘ri**)** bo‘lsa. Misol uchun:

Bu ifodada “**true**” chunki 2 ta operandlar: (2<3) va (3<4) bir vaqtda “**true**”.

```
bool result = (2 < 3) && (3 < 4);
```

&& mantiqiy operatori **qisqa hisoblash** deb ataladi, chunki u qo‘shimcha kerak bo‘lmagan hisoblashlar uchun vaqt sarflamaydi. U dastlab birinchi operandga qaraydi agar qiymat **false (**yolg‘on**)** bo‘lsa, u ikkinchi qismini hisoblash uchun vaqt yo‘qotmaydi. Birinchi qismi “**true**” bo‘lmasa oxirgi qiymati ham “**true**” bo‘lmasligi aniq. Shu sababli bu mantiqiy ifoda qisqa xisoblash **"and"** deb ataladi.

Shuningdek, **||** operatori **true** qiymat qaytaradi agar ikkalasining bittasi **true** qiymat bo‘lsa. Misol:

```
bool result = (2 < 3) || (1 == 2);
```

Bu misolda **“true”,** chunki birinchi operandi **“true”.** Shunga o‘xshab **&&** operator hisoblashni tez amalga oshiradi – agar birinchi operator “**true**” bo‘lsa, ikkinchi qismi hisoblanmaydi, ammo natija aniq bo‘ladi. Shu sababli bu mantiqiy ifoda qisqa xisoblash **"or"** deb ataladi.

2.2 & va | mantiqiy operatorlar

& va **|** operatorlar **&&** va **||** operatorlar bilan bir xil. Farqi shundaki **&&** va **||** operatorlar ikkila operandlarni biridan keyin boshqasini hisoblaydi, shunga qaramasdan oxirgi natija aniq bo‘ladi. **&** va **|** operatorlar esa ikkala operandlarni to‘liq tekshiradi shuning uchun bu operatorlar **to‘liq mantiqiy operatorlar** deb ataladi va ular kam foydalaniladi.

Misol uchun, ikki operand **&** operator orqali taqqoslanganda birinchisi **“false”(**yolg‘on) bo‘lsa ham, ikkinchisini ham hisoblaydi. Natija aniq **“false**” bo‘lsa ham. Shuningdek, ikki operand **|** operator orqali taqqoslanganda birinchisi **“true”** (rost) bo‘lsa ham, ikkinchisini ham tekshiradi, natijasi “**true**” bo‘lsa ham(ikkinchisini tekshirmasdan ham).

Biz mantiqiy **&** va **|** operatorlarini butun sonlardagi **&** va **|** amallar bilan garchi ular bir xil yozilsa ham adashtirmasligimiz kerak. Ular turli qiymatlar qabul qiladi (mantiqiy ifoda yoki butun son) va turli natijalar qaytaradi (mantiqiy yoki butun) va ularning bajarish vazifasi bir xil emas.

2.3 ^ va ! mantiqiy operatorlar

^ operatori maxsus **OR** (**XOR**), u to‘liq operatorga bog‘liq, chunki ikki qismi ham ketma – ket hisoblanadi. Agar birinchisini qiymati “**true**” bo‘lsa natija “**true**” bo‘ladi, ammo ikkalasi bir vaqtda bir xil bo‘lmaydi. Agar shunday bo‘lsa, natija “**false**” bo‘ladi. Masalan:

```
Console.WriteLine("Exclusive OR: "+ ((2 < 3) ^ (4 > 3)));
```

Natijasi:

```
Exclusive OR: False
```

Dastlab ikkila ifoda (2<3) va (4>3) bajariladi va ular natijasi “**true**” **OR** operatoridan so‘ng natija “**false**” ga teng bo‘ladi.

! operatori mantiqiy tipdagi ifodani aksini qaytaradi. Masalan:

```
bool value = !(7 == 5); // True
Console.WriteLine(value);
```

Yuqoridagi kod natijasi **true** qiymat qaytaradi chunki “7==5” ifoda **False** qiymat qaytaradi va **False** ni inkori esa **true (**rost)ni beradi. Eslatib o‘tamiz konsolda **true** qora ekranga chiqanda katta harf bilan yoziladi **“True”.** Bu kamchilik VB.NET tilidan **.NET FRAMEWORK** ga kirib kelgan.

“**if**” va “**if – else**” shart operatorlari

Ifodalarni qanday taqqoslashni ko‘rganimizdan so‘ng, biz shart operatorlari bilan ishlashni davom ettiramiz, u bizga dasturlashning mantiqiy qismini bajarishga ruxsat beradi.

Shart operatorlar **if va if – else** operatorlar bo‘lib, ular orqali shart tekshiradi va dasturda turli xil amallar bajariladi.

3. “ if ” shart operotori

if ning asosiy sintaksisi quyidagicha:

```
if (Boolean expression)
{
   Body of the conditional statement;
}
```

Yuqoridagi sintaksisda **if** dan so‘ng Boolean express-mantiqiy ifoda bo‘lib, agar u rost qiymat qaytarsa tana qismga. Mantiqiy ifoda butun tipda bo‘la olmaydi.

Tana qismi ya’ni ikkitalik qavs ichida berilgan: **{ }.** U bir va ko‘p operatorlardan iborat bo‘lishi mumkin. Bizda qo‘shimcha blok operatorlar yordamida boshqa jarayonlarni ham amalga oshirishimiz mumkin bo‘ladi va {} qavslar ketma – ketligida bajariladi.

{} qavs ichidagi operatorlarga o‘tishi uchun **if** operatoridagi mantiqiy qiymati **true** qiymatlarini qaytarishi zarur. Agar ifoda hisoblanganda qiymati **true** bo‘lsa, asosiy qismi bajariladi. Agar natija **false** bo‘lsa asosiy qismidan sakrab o‘tib ketadi.

“if” shart operatoriga – misol:

Misoldagi **if** shart operatori ishlatilishiga etibor qaratamiz:

```
static void Main()
{
   Console.WriteLine("Enter two numbers.");
   Console.Write("Enter first number: ");
   int firstNumber = int.Parse(Console.ReadLine());
   Console.Write("Enter second number: ");
   int secondNumber = int.Parse(Console.ReadLine());
   int biggerNumber = firstNumber;
   if (secondNumber > firstNumber)
   {
      biggerNumber = secondNumber;
   }
   Console.WriteLine("The bigger number is: {0}", biggerNumber);
}
```

Agar biz 4 va 5 butun sonlarini kiritsak natija quyidagicha bo‘ladi:

```
Enter two numbers.
Enter first number: 4
Enter second number: 5
The bigger number is: 5
```

3.1. If shart operatori va { } qavslar

Agar **if** shartining ichida bitta operator bo‘lsa, {} qavslar ish bajarmaydi. Ammo {} qavslardan foydalanish xatoliklarni kamayishiga zamin yaratadi.

Bu yerda {} qavslar bajarilmasligiga misol:

```
int a = 6;
if (a > 5)
   Console.WriteLine("The variable is greater than 5.");
   Console.WriteLine("This code will always execute!");
// Bad practice: misleading code
```

Bu misolda **if** operatorini asosiy qismining ikkita chiqarish amali a>5 ifoda rost bo‘lgan xolda birinchi chiqarish operatori bajariladi.

Doimo if bloklari uchun {} qavslarni qo‘ying hatto u bitta operator bo‘lsa ham.

3.2. “if – else” shart operatori

Ko‘p dasturlash tillarida shart operatorlari **else** bilan ishlatiladi shu jumladan C# da ham. **if – else** strukturasini tuzilishi quyidagicha:

```
if (Boolean expression)
{
  Body of the conditional statement;
}
else
{
  Body of the else statement;
}
```

if – else strukturasining sintaksisi **if** mantiqiy ifoda va uning tana qismi, **else** (u holda) tana qismidan iborat . **Else –** strukturasining asosiy qismi bir va ko‘p operatorlardan va {} qavslardan iborat bo‘lishi mumkin.

Bu jarayon quyidagicha ishlaydi: () qavslar ichidagi ifoda (mantiqiy ifoda) – **true** yoki **false** ga teng bo‘lishi mumkin. Uning natijasi ikkita ehtimolga bog‘liq. Agar mantiqiy ifodani hisoblaganda **true** ga teng bo‘lsa, **if** operatorining tana qismi bajariladi va **else –** operatori ishi bajarilmaydi. Agar mantiqiy ifodani hisoblaganda **false** ga teng bo‘lsa, **else –** qismi bajariladi, **if** operatorining tana qismi bajarilmaydi **else** operatorining tana qismi bajariladi.

“if – else” shart operatoriga misol:

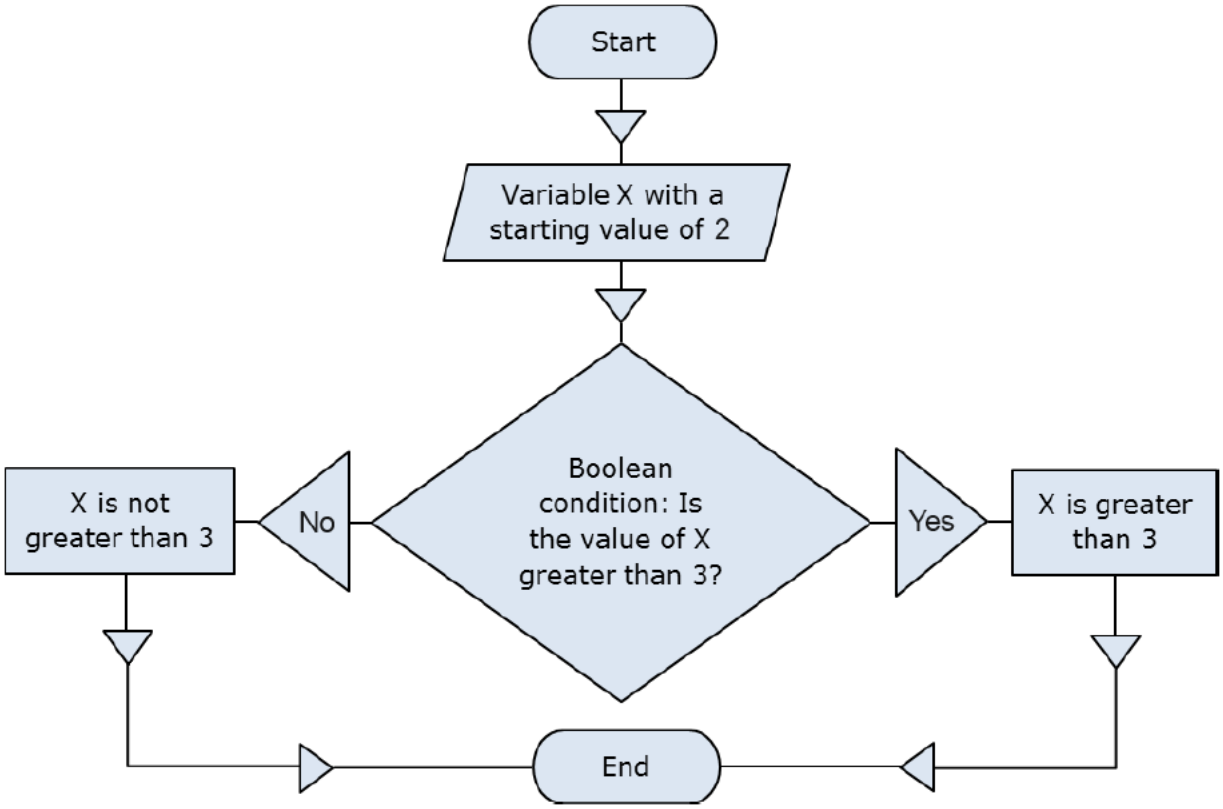
Bu misolda **if – else** operatori ishlatilishini ko‘rib chiqamiz:

```
static void Main()
{
   int x = 2;
   if (x > 3)
   {
      Console.WriteLine("x is greater than 3");
   }
   else
   {
      Console.WriteLine("x is not greater than 3");
   }
}
```

Bu misoldagi kodda x butun son berilgan, agar x>3 bo‘lsa natija **“x ni qiymati 3 dan katta”** aks holda (**else**) uni qiymati **”x ni qiymati 3 dan kichik”** bo‘ladi. x butun sonni qiymati 2 ga teng, mantiqiy ifodani hisoblaganda mantiqiy ifoda natijasi “**false**” qiymat qaytaradi va biz yuqorida aytib o‘tganimizdek (agar mantiqiy ifoda “**false**” qiymat qaytaradigan bo‘lsa, shart operatorning tana qismi bajarilmaydi) keyingi **else** strukturasi bajariladi. Misol natijasi:

```
x is not greater than 3
```

Quyidagi sxema jarayonni qanday ishlashini ko‘rsatadi:



4. if strukturasini ichma – ich joylashishi

Ba’zan dastur ichida mantiqiy dastur yoki **if –** strukturalari ichma – ich joylashishi orqali ifodalaniladi. Biz bularni **ichma – ich if** yoki **ichma – ich if – else** strukturalari deb ataymiz.

if yoki if – else strukturasi boshqa bir **if** yoki **else** strukturasi ichida joylashishi mumkin. Ko‘p jarayonlarda **else** bo‘limi dastlabki **if** bo‘limiga o‘xshash bo‘ladi. Buni biz **else** bo‘limi **if** bo‘limiga bog‘liq deb tushumishimiz mumkin.

Ichma – ich joylashgan jarayonlarni uchtadan oshirib yuborish yaxshi emas, biz bitta shart ichida uchta va undan ko‘p shartlarni bajarmasligimiz kerak. Agar ba’zi sabablarga ko‘ra uch va undan ko‘proq strukturani ichma – ich joylashtiradigan bo‘lsak, biz bu qismini alohida metodaga o‘tkazishimiz kerak (metod bo‘limida ko‘ramiz).

“if” strukturasini ichma – ich joylashishi – misol:

```
int first = 5;
int second = 3;

if (first == second)
{
   Console.WriteLine("These two numbers are equal.");
}
else
{
   if (first > second)
   {
      Console.WriteLine("The first number is greater.");
   }
   else
   {
      Console.WriteLine("The second number is greater.");
   }
}
```

Yuqoridagi misolda biz ikki sonni taqqoslaymiz: birinchi biz ularni teng ekanini tekshiramiz agar tekshirish natijasi yolg‘on qiymat qaytarsa ularni yana taqqoslaymiz va qaysi biri kattaligini aniqlaymiz. Yuqoridagi kod natijasi:

```
The first number is greater.
```

5. “if – else –if – else …”operatorining ketma – ketligi

Ba’zan biz **if strukturasini ketma – ket** ishlatishimiz zarur bo‘lib, qoladi bunda **else** qismi yangi **if** strukturasiga teng bo‘ladi. Agar biz ichma – ich **if** strukturasidan foydalansak kod to‘g‘ri javobdan ancha uzoqlashadi. **Else** dan keyin yangi **if** dan foydalansak to‘g‘ri javobga yaqinlashamiz. Masalan:

```
char ch = 'X';
if (ch == 'A' || ch == 'a')
{
   Console.WriteLine("Vowel [ei]");
}
else if (ch == 'E' || ch == 'e')
{
   Console.WriteLine("Vowel [i:]");
}
else if (ch == 'I' || ch == 'i')
{
   Console.WriteLine("Vowel [ai]");
}
else if (ch == 'O' || ch == 'o')
{
   Console.WriteLine("Vowel [ou]");
}
else if (ch == 'U' || ch == 'u')
{
   Console.WriteLine("Vowel [ju:]");
}
else
{
   Console.WriteLine("Consonant");
}
```

Bu misoldagi dastur biz kiritgan o‘zgaruvchi ingliz tili alifbosidagi unli harflar bilan taqqoslaydi. Har bir taqqoslash ketma – ket bajariladi qachonki undan oldingi qismi to‘g‘ri bo‘lmasa. Agar hech bir **if** bo‘limi bajarilmasa oxirgi **else** bo‘limi bajariladi. Quyida dastur natijasi:

```
Consonant
```

5.1. “if” shart operatorini amaliyotdagi tavsifi

Quyida **if** da yozish uchun ba’zi tavsiyalar, strukturalar berilgan :

- chalg‘imaslik uchun **if** va **else** operatorlaridan so‘ng {} qavslardan foydalaning ;

- chalg‘imaslik va o‘qish oson bo‘lishi uchun har birini o‘z o‘rnida ya’ni o‘zining formatida (**else if** dan keyin) yozing;
- agar mumkin bo‘lsa **if**-**else**-**if**-**else** strukturalari yoki ichma-ich **if** strukturalari o‘rniga **switch**-case strukturasidan foydalaning

6. “switch-case” tanlash operatori

Bu bo‘limda **switch** operatoridan foydalanishni ko‘ramiz. U berilganlardan biror yechimni tanlash uchun foydalaniladi.

6.1. “switch – case” operatori qanday ishlaydi?

Switch – case strukturasi dastur kodining qaysi qismi berilgan ifoda bo‘yicha hisoblanishini tanlab beradi. Formati quyidagicha:

```
switch (integer_selector)
{
   case integer_value_1:
      statements;
      break;
   case integer_value_2:
      statements;
      break;
   // …
   default:
      statements;
      break;
}
```

Tanlovchi bu - natijaning qiymatini taqqoslab bo‘ladigan qilib qaytaradigan ifodadir. U son yoki satr bo‘lishi mumkin. **switch** operatori tanlovchining har bir natijasini **switch** strukturasidagi har bir **case label** lar bilan solishtiradi. Agar **case** lardan unga mosi topilmasa, **default** qismi mavjud bo‘lsa, unga o‘tiladi va bajariladi. Tanlovchining qiymati **switch** strukturasidagi qiymatlar solishtirilganidan keyin hisoblanadi.

Yuqoridagi tavsifdan ko‘rinadiki har bir **case break** operatori bilan tugaydi. C# dasturlash tili kodning **case** qismi oxirida **break** ni ishlatishni so‘raydi. Agar **case** da **break** operatori ishlatilmagan yoki tushirib qoldirilgan bo‘lsa, qachonki **break** operatorini topmaguncha keyingi **case** larga o‘tib ketadi. Hattoki **default** strukturasidan keyin ham **break** ni qo‘yish shart. **default** qism **switch** operatori

so‘nggida bo‘lishi shart emas, ammo uni oxirida qo‘yishni tavsiya etiladi va **switch** strukturasi o‘rtasida ishlatilmaydi.

6.2. Switch ifodalari uchun qoidalar

Switch operatori ko‘p operandlardan birini tanlashdagi to‘g‘ri yo‘l. U aniq qiymat bilan hisoblanadigan o‘zgaruvchi so‘raydi. O‘zgaruvchi tipi **int, char, string** yoki **enum** bo‘lishi mumkin. Agar biz massiv yoki haqiqiy son tipidagi (float) o‘zgaruvchi kiritsak, u ishlamaydi. Butun bo‘lmagan ma’lumot tiplari uchun **if** strukturasidan foydalanamiz.

6.3. Ikkilangan case lardan foydalanish

Qachonki bittadan ko‘p **case** larda bir hil strukturani ishlatmoqchi bo‘lsak, u holda biz ikkilangan **case** lardan foydalanamiz. Navbatdagi misolda ko‘rib chiqamiz:

```
int number = 6;
switch (number)
{
   case 1:
   case 4:
   case 6:
   case 8:
   case 10:
     Console.WriteLine("The number is not prime!"); break;
   case 2:
   case 3:
   case 5:
   case 7:
     Console.WriteLine("The number is prime!"); break;
   default:
     Console.WriteLine("Unknown number!"); break;
}
```

Yuqoridagi misolda biz **break** operatorlarisiz ikkilangan **case** lardan foydalandik. Bu holda kiritilgan o‘zgaruvchining qiymati 6 ga teng va bu qiymat **case** operatorlaridagi butun qiymatilari bilan solishtiriladi. Qachonki moslik topilganda undan keyingi **case** lar bloklanadi. Agar moslik topilmasa **default** qismi ishga tushadi. Yuqoridagi misolning natijasi quyidagicha:

```
The number is not prime!
```

6.4. Switch-case foydalanish uchun qo‘llanmalar

kod oson o‘qilishi uchun **default** operatorini **switch** operatoridan keyin qo‘yish yaxshi natija olib keladi

case larni birinchi qo‘yish samarali. **Case** operatorlari struktura oxirida joylashishi mumkin qachonki biz murakkab dastur tuzayotganimizda.

agar **case** da butun sonlardan foydalanilsa, ularni o‘sib borish tartibida joylashtirish maqsadga muvoffiq.

agar **case** qiymatlari belgili tipda bo’lsa ,belgilar al**if**bo bo’yicha tartiblash tavsiya qilinadi.

default qismi foydalanilgan dastur to‘liq dastur bo‘ladi. Bu to‘liq dasturda **default** qismidan foydalanilmagan bo‘lsa, dastur xatolikka yo‘l qo‘yilgan deb aytish mumkin.

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

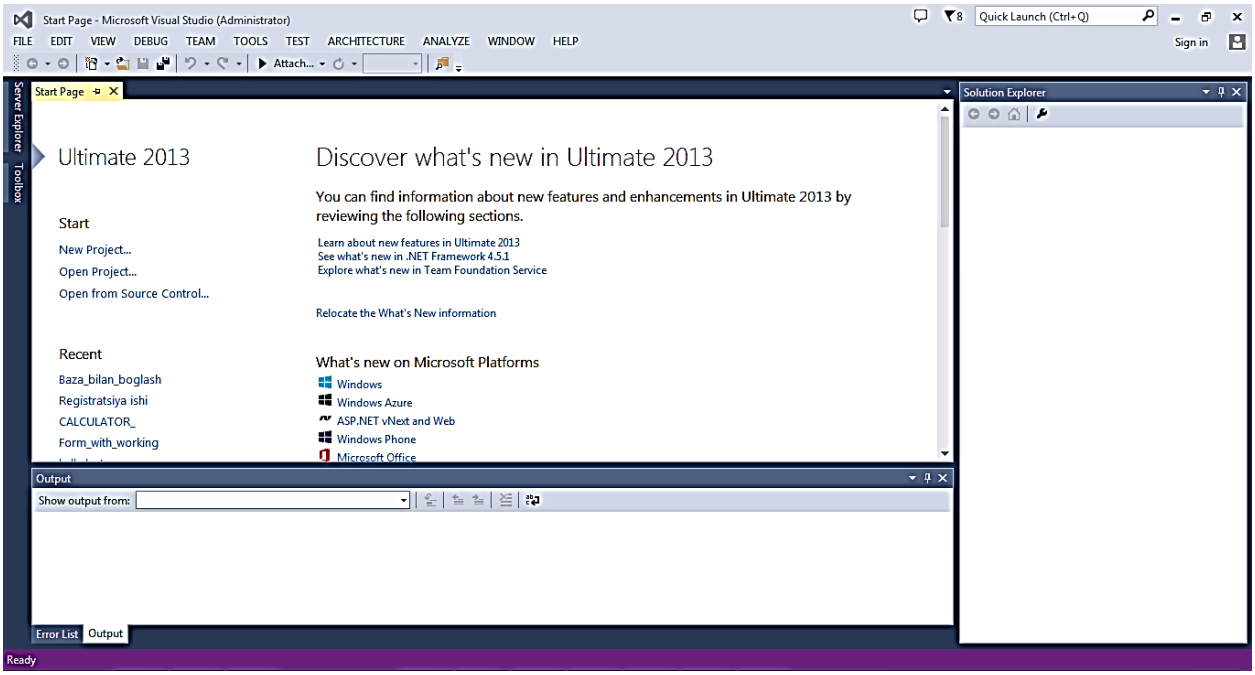
- **if –** shart operatori bo‘lib, qandaydir shartni rostlikka tekshirish natijasiga ko‘ra programmada tarmoqlanishni amalga oshiradi
- **“if – else”** – shart operatori bo‘lib, qandaydir shartni rostlikka tekshirish natijasiga ko‘ra programmada tarmoqlanishni amalga oshiradi
- **System.String –** satr tiplarni o‘z ichiga oluvchi klass hisoblanadi
- **ReadKey(…) –** klavish kombinatsiyasini o‘qish
- **ReadLine(…)** – belgilarni bitta qatorini o‘qish
- **Write(…) –** konsolga argumentlarni chiqarish
- **WriteLine(…) –** konsolga ma’lumotlarni chiqarish va keyingi qatorga o‘tish

- **Console.ReadKey() –** klavish bosilguncha ekranni ushlab turadi
- **KeyChar –** kiritilgan belgini ushlab turadi
- **Modifiers –** [Ctrl], [Alt], … xoltini ushlab turadi
- **int.Parse(string) –** satrni butun tipga o‘tkazadi
- **Convert klassi –** metodidan foydalangan holda boshqa tipga o‘tkazish uchun ishlatiladi
- **try-catch –** hatoliklarni ushlab qolish uchun ishlatiladi
- **TryParse() –** hatoliklarni ushlab qolish uchun ishlatiladi

Amaliy mashg‘ulot.

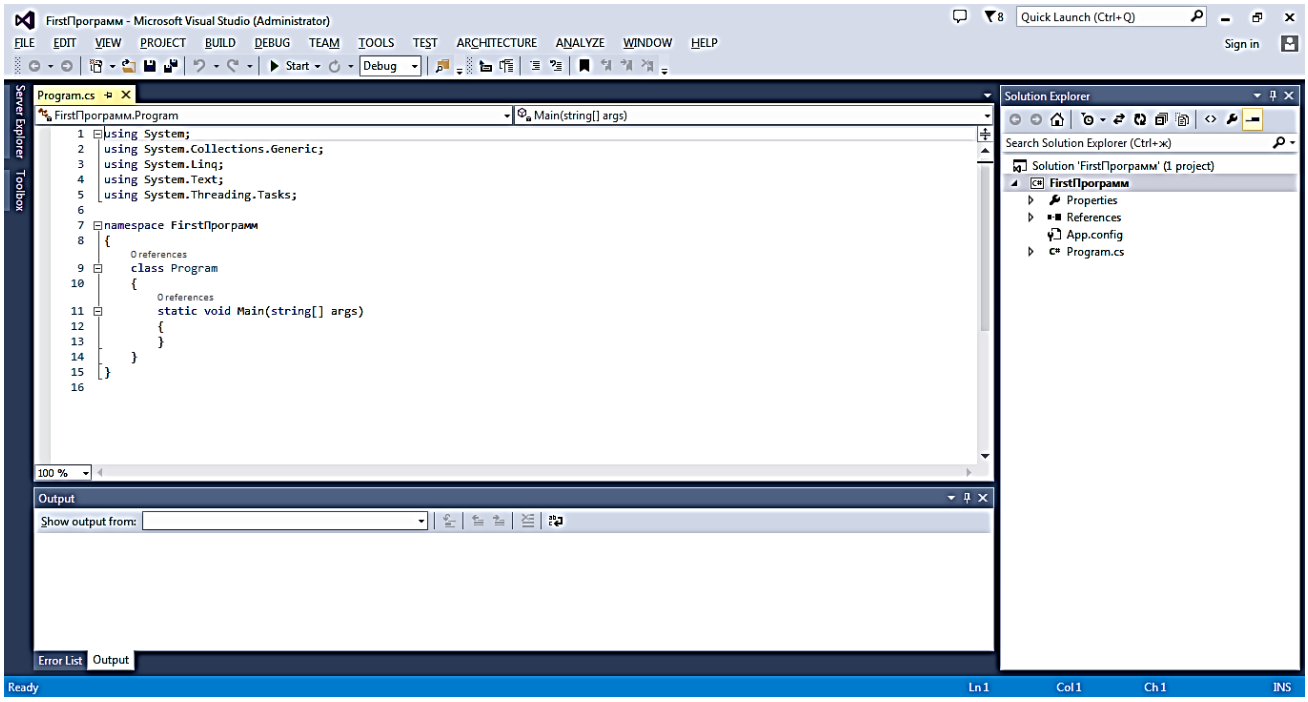
1-misol. 2 ta sonni konsoldan o‘qib oling. Ulardan kattasini topuvchi dastur tuzing.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **1.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



5.1-rasm. Visual Studio 2013 muhitining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **FirstProgramm** kabi kiritib, **OK** tugmasini bosamiz. Natijada **1.2 rasm**da keltirilgan quyidagi oyna shakllantiriladi.



1.2-rasm. Dasturiy kod oynasi

Endi berilgan masala kodini

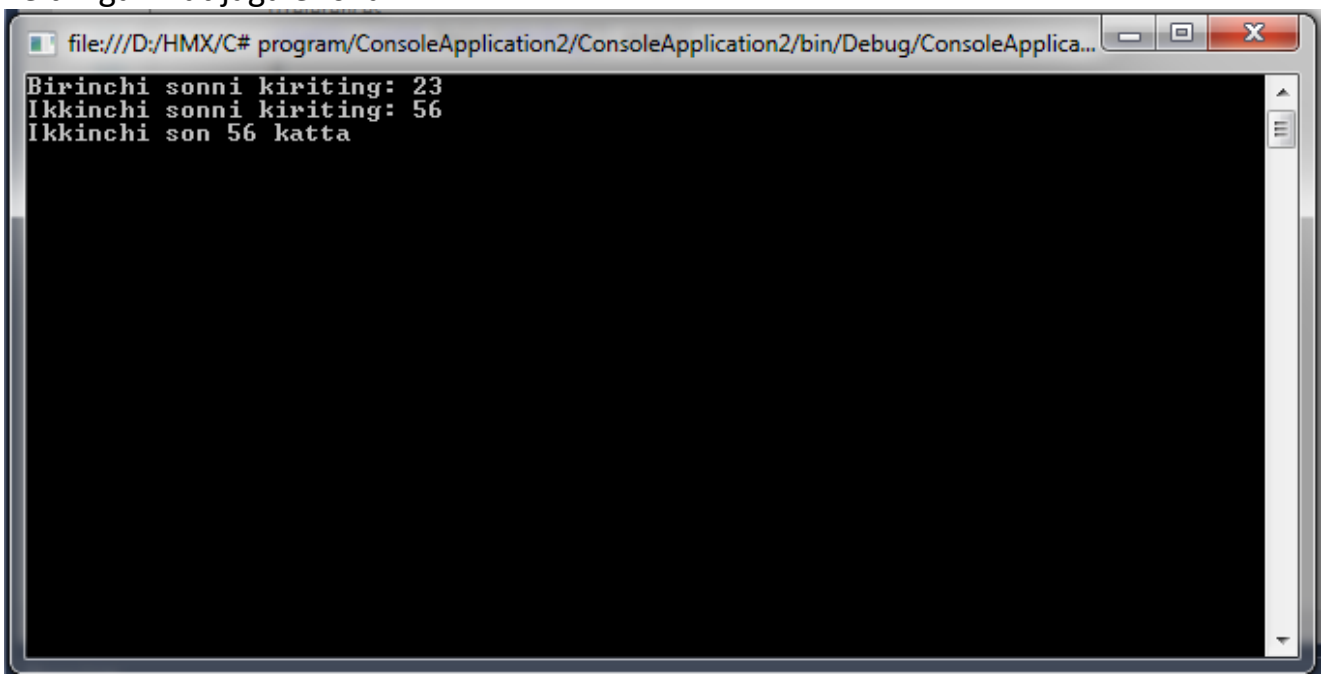
```
static void Main(string[] args)
        {
    }
```

S# dagi asosiy metod blokiga yoziladi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace FirstProgramm
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Birinchi sonni kiriting: ");
            int a = int.Parse(Console.ReadLine());

            Console.Write("Ikkinchi sonni kiriting: ");
            int b = int.Parse(Console.ReadLine());

            if (a > b)
```

```
            {
                Console.WriteLine("Birinchi son {0}  katta",
a);
            }
            else
            {
                Console.WriteLine("Ikkinchi son {0} katta",
b);
            }

            Console.ReadKey();
        }
    }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. ***a*** *va* ***b*** sonlarni kiritamiz va **1.3-rasmda** keltirilgan natijaga erishamiz.



1.3-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

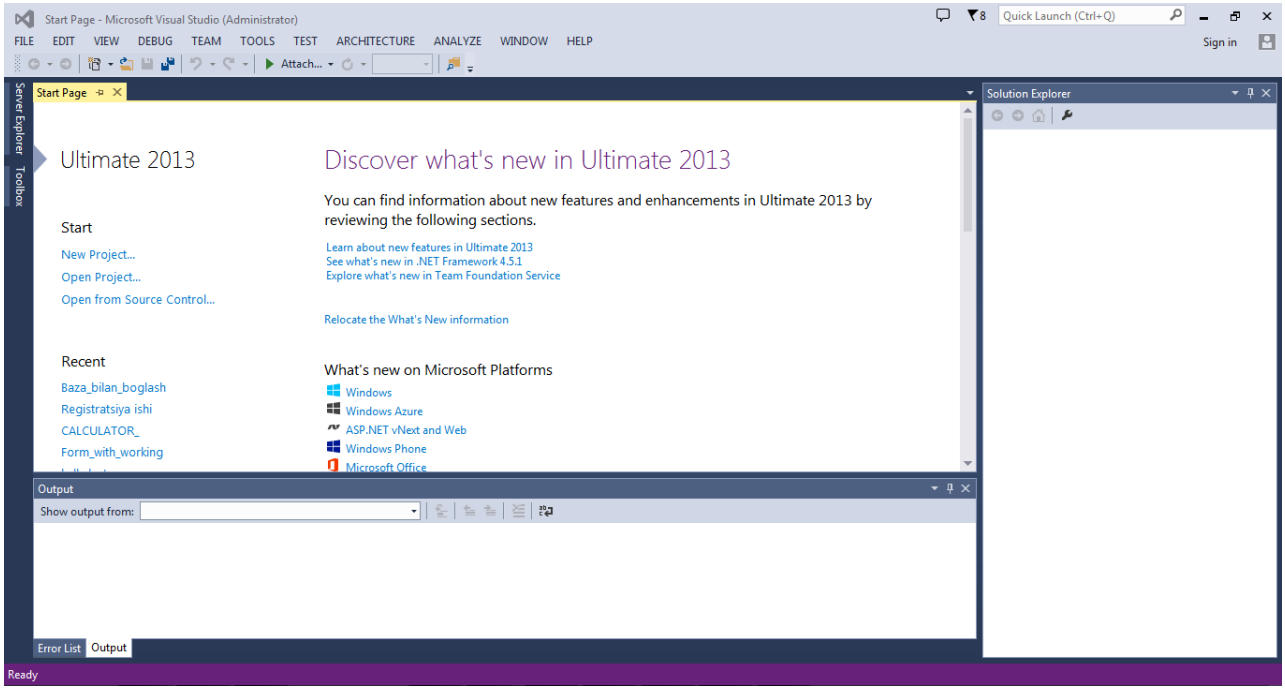
2-misol. Quyidagi shartlar bajarilganda true, aks holda false qiymat qabul qiladigan ifoda C# tilida yozilsin:

a) butun n va m sonlari bir paytda toq (true) yoki juft (false) sonlar;

b) a, b mantiqiy o‘zgaruvchilardan faqat bittasi true qiymatiga ega;

d) a, b, s mantiqiy o‘zgaruvchilardan faqat bittasi true qiymatini qabul qiladi.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **2.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



2.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

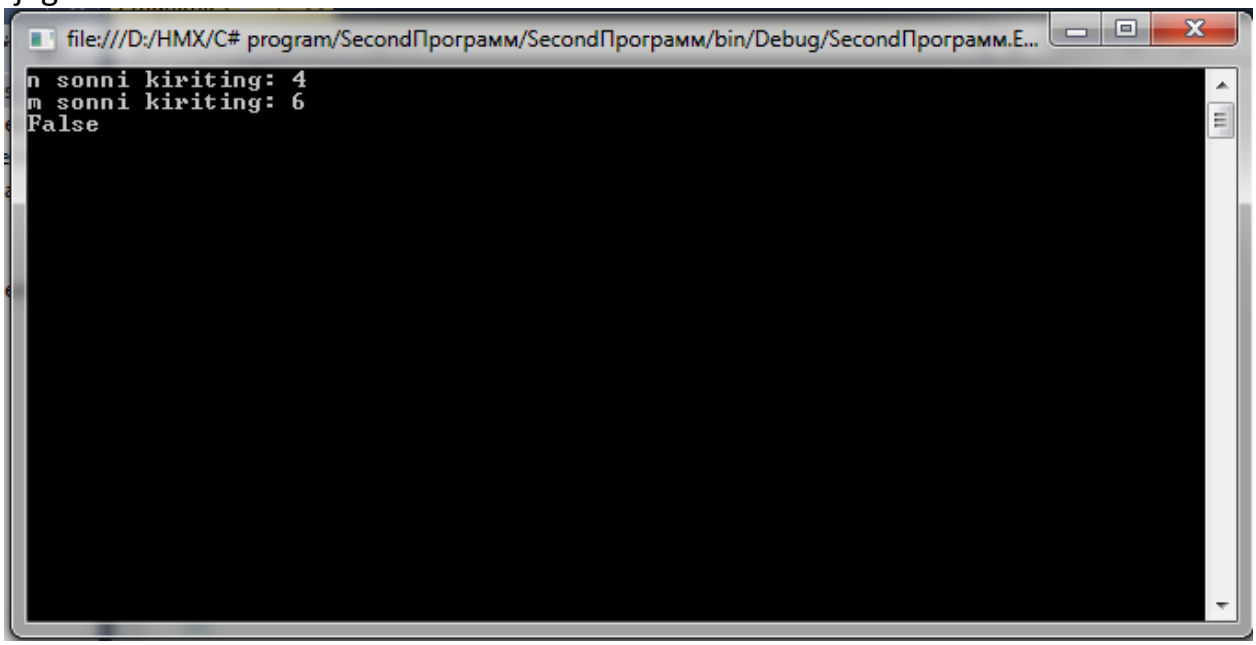
VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **SecondProgramm** kabi kiritib, **OK** tugmasini bosamiz.

Endi berilgan masala kodini kiritamiz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SecondProgramm
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("n sonni kiriting: ");
            int n = int.Parse(Console.ReadLine());

            Console.Write("m sonni kiriting: ");
            int m = int.Parse(Console.ReadLine());

            Console.WriteLine((n % 2 != 0) && (m % 2 != 0));
            Console.ReadKey();

        }
    }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. ***a*** *va* ***b*** sonlarni kiritamiz va **1.3-rasmda** keltirilgan natijaga erishamiz.



2.2-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

3 – misol. Ixtiyoriy butun ikkita son berilgan bu sonlar ustida tanlangan arifmetik amalga (“+”, “-”, “*”, “/”) mos hisoblash amalga oshiruvchi dastur tuzing. Arifmetik amallardan farqli belgi kiritilsa, bu haqda xabar berilsin.

Yangi proekt yaratamiz va dastur kodini yozamiz va ыuyidagicha:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication2
{
   class Program
   {
      static void Main(string[] args)
      {
         int a,b;
         float natija=0;
         char amal;
         Console.Write("a=");
         a=int.Parse(Console.ReadLine());
         Console.Write("b=");
         b=int.Parse(Console.ReadLine());

         Console.Write("Amalni tanlang(+,-,*,/): ");
         amal=char.Parse(Console.ReadLine());
         switch(amal)
         {
          case '+':natija=a+b;break;
          case '-':natija=a-b;break;
          case '*':natija=a*b;break;
          case '/':natija=a/b;break;
          default: Console.WriteLine("bunday amal yo'q"); break;
         }
        Console.WriteLine("Hisoblash natijasi={0}",natija);

        Console.ReadKey();


        Console.ReadKey();
```

```
      }
   }
}
```

Amaliyot topshiriqlari

22. Ifodaning qiymati topilsin:

a. x * x + 2 * y <= 4, agar x = 0.3, y = -1.6;
b. c % 7 = = c / 5 - 1, agar c = 15;
c. (10 – p) % 2 = = 0, agar p = 0.182.

23. Quyidagi shartlar bajarilganda rost, aks holda yolg‘on qiymat qabul qiluvchi mantiqiy munosabatlar C# tilida yozilsin:

a. c butun soni 7 ga bo‘linadi;
b. ax^2+bx+c=0 tenglama haqiqiy ildizlarga ega emas;
c. (x,y) nuqta, markazi (1,0) nuqtada bo‘lgan r radiusli doiraning tashqarisida yotadi;
d. n natural soni - to‘liq kvadrat.

24.Ifodalarning qiymatlari hisoblansin:

a. !(n%2), agar n=15;
b. t && (p%3==0), agar t=true , p=101010;
c. (x+y!=0) && (y>x), agar x=2, y=1;
d. (x+y=0) || (y>x), agar x=2, y=1;
e. a || (!b), agar a=false, b=true.

25.Quyidagi shartlar bajarilganda rost, aks holda yolg‘on qiymat qabul qiluvchi mantiqiy munosabatlar C# tilida yozilsin:

a. 0<x<1;
b. x=max(x,y,z);
c. x ≠ max(x,y,z) (inkor amalidan foydalanilmasin);
d. a,b mantiqiy o‘zgaruvchilardan kamida bittasi true;
e. har ikkala a,b mantiqiy o‘zgaruvchilar qiymatlari true.

26.Ayniyatlar isbotlansin:

a. a && (!a) false;
b. a || (!a) true;
c. ! (!a) a;
d. true || a true;
e. false && a false;

f. a || a a.

27.Hisoblansin:

a. false || (1/1>0);
b. (1/2>0) && true.

28.Ifodalardagi amallar bajarilish tartibi ko‘rsatilsin:

a. a && b || ! c && d ;
b. (x>=0)|| t && toq(x) || (y*y!=4).

29.O‘zgaruvchilar qiymatlari a=true va b=false bo‘lganda quyidagi ifodalar hisoblansin:

a. a || b && !a ;
b. (a || b) && !a;
c. !a && b ;
d. !(a &&b).

30.Quyidagi shartlar bajarilganda rost, aks holda yolg‘on qiymat qabul qiluvchi ifodalar C# tilida yozilsin:

a. x [0,1] kesmaga tegishli;
b. x [0,1] kesmaga tegishli emas;
c. x [2,5] yoki [-1,1] kesmalarga tegishli;
d. x [2,5] yoki [-1,1] kesmalarga tegishli emas;
e. x, y, z sonlaridan har biri musbat;
f. x, y, z sonlaridan hech bo‘lmaganda biri musbat;
g. x, y, z sonlaridan hech biri musbat emas;
h. x, y, z sonlaridan faqat biri musbat;
i. mantiqiy o‘zgaruvchi a true, b esa false qiymatini qabul qilgan holda;
j. y - yil Kabisa yilidir (Kabisa yili 4 ga karrali yillar hisoblanadi. Biroq, 100 ga karrali yillar orasida faqat 400 ga karrali yillar kabisa yili deyiladi. Masalan, 1700, 1800, 1900-oddiy yillar, 2000-kabisa yili).

31. Quyidagi shartlarga mos keluvchi soha tekislikda chizilsin:

a. (y>=x) && (y+x>=0) && (y<=1);
b. (x*x+y*y<1) || (y>0) && (y<=1);
c. (floor(y)==0) && (ceil(x+1)==0).

32. Agar (x,y) nuqta bo‘yalgan sohaga tegishli bo‘lsa, t mantiqiy o‘zgaruvchi true qiymatini qabul qiladigan ifoda yozilsin.

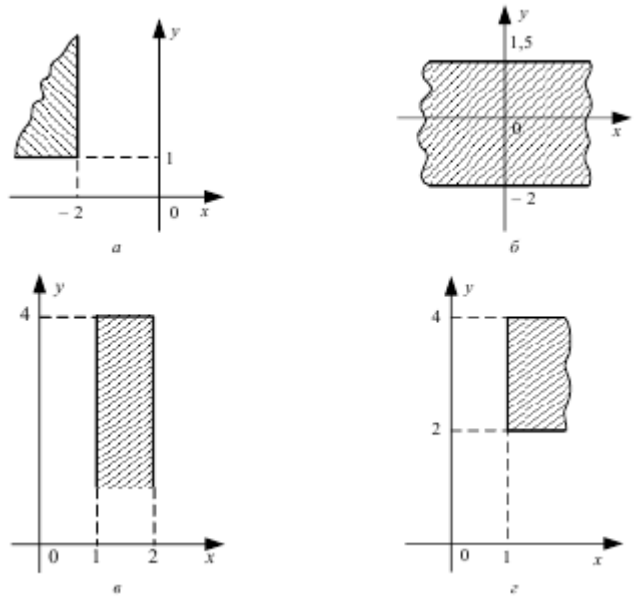


Рис. 3.1, *а—г*

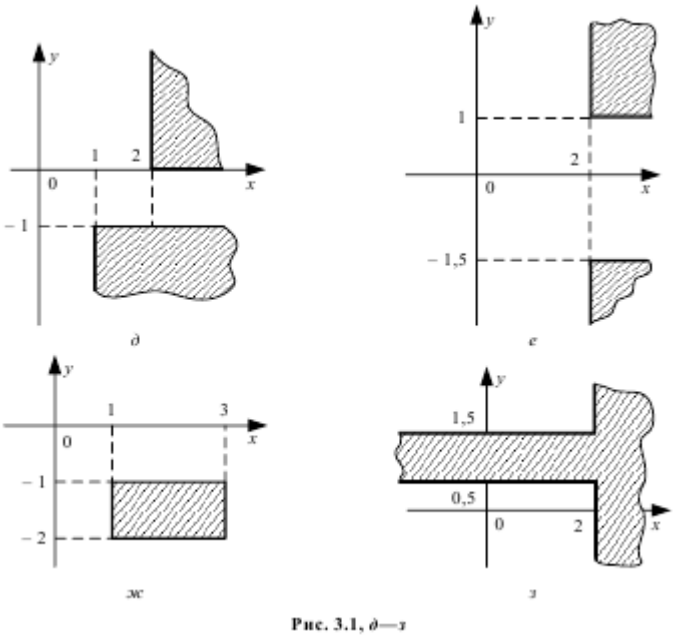


Рис. 3.1, *д—з*

33. Ifodaning qiymati hisoblansin:

a. false < true;
b. (32 || false)==1;
c. 9+3*true;
d. 16+true/2.

34. Ifodaning qiymati hisoblansin:

a. !(--s) &&((int)(s)==1), agar s=true;
b. (p<true)==(q==false), agar p=q=true;
c. a && b > a || b, agar a=true, b=false.

35. Quyidagi shartlar bajarilganda rost, aks holda yolg‘on qiymat qabul qiladigan ifoda S# tilida yozilsin:

a. butun n va m sonlari bir paytda toq yoki juft sonlar;
b. a, b mantiqiy o‘zgaruvchilardan faqat bittasi true qiymatiga ega;
c. a, b, c mantiqiy o‘zgaruvchilardan faqat bittasi true qiymatini qabul qiladi.

36. Quyidagi ifodalar o‘rinli bo‘ladigan soha XOY tekisligida chizilsin:

a) (fabs(x)<=1)==(fabs(y)>=1); b) (x*x+y*y<=4)== (y<=x).

37. Ayniyatlar isbotlansin:

a. !(a || b)≡ (!a) &&(!b) ;
b. a && (b || c) ≡ (a && b) || (a && c) ;
c. a<=b ≡ !a || b ;
d. a && b ≡ (a<true)<b ;
e. !a ≡ a<true.

38. Mantiqiy a,b,c o‘zgaruvchilar uchun quyidagi ifodalarni taqqoslash amallari qatnashmagan ko‘rinishga keltirilsin:

a. a<b;
b. a==b;
c. (a<b)==a.

39. Agar a=true va x=1 bo‘lsa, quyidagi mantiqiy d o‘zgaruvchi qanday qiymat qabul qiladi?

a. d=x<2;
b. d=!a || x%2;
c. d=a%2!=x.

40. Quyidagi shart bajarilganda t mantiqiy o‘zgaruvchisi true, aks holda false qiymatini o‘zlashtirsin:

a. x, y, z sonlar o‘zaro teng;
b. x, y, z sonlardan faqat ikkitasi o‘zaro teng;
c. x- musbat son;
d. p- soni q ga qoldiqsiz bo‘linadi (p va q-natural sonlar);
e. ax^2+bx+c=0 tenglama bitta yechimga ega, bu yerda a, b va c lar 0 ga teng bo‘lishi mumkin;
f. uch xonali butun c sonining o‘nli yozuviga ´5´ raqami kiradi;
g. shaxmat taxtasining (h1,v1) va (h2, v2) kataklari bir xil rangga ega (h1,v1, h2 va v2 o‘zgaruvchilar 1-8 oralig‘idagi qiymatlarni qabul qiluvchi o‘zgaruvchilar);
h. shaxmat taxtasining (h1,v1) katakida joylashgan «farzin» (h2,v2) katakka xavf soladi.

41. Agar tomonlarining uzunliklari ixtiyoriy a, b, c sonlarga teng bo‘lgan uchburchakni qurish mumkin bo‘lmasa 0, aks holda – uchburchak teng tomonli bo‘lsa 3, teng yonli bo‘lsa 2 va boshqa hollar uchun 1 qiymatini chop qiluvchi dastur tuzilsin.

42. Agar uchta haqiqiy, o‘zaro teng bo‘lmagan x,y,z sonlar yigindisi 1 kichik bo‘lsa, uchta sonning eng kichigi qolganlari yigindisining yarmisi bilan almashtirilsin, aks holda x va y kichigi qolganlarining yigindisining yarmi bilan almashtirilsin.

43. Berilgan 50 ta haqiqiy sonlarning eng kattasini topadigan programma tuzilsin.

Haqiqiy x,y,z sonlar berilgan bo‘lsa, quidagilar aniqlansin:

a) max(x,y,z);
b) max(x,y)+min(y,z);
k) max(x+y+z,x*y*z);
d) min((x+y+z)/2,x*z)+1;

44. Uchta X,Y,Z haqiqiy sonlar berilgan, agar ular monoton bo‘lsa ularning qiymatlari ikkilantirilsin, aks holda har bir uzgaruvchi qiymati qarama qarshisiga almashtirilsin.
45. Butun n>0 va n ta haqiqiy sonlar berilgan. Ular orasidan manfiylari nechtaligini aniqlaydigan dastur tuzilsin.

    OX va OY o‘qlarida yotmaydigan haqiqiy son ko‘rinishidagi koordinatalari berilgan. Bu nuqta joylashgan koordinata choragining nomeri chop etilsin.
46. Bo‘sh bo‘lmagan va oxiri 0 soni bilan tugaydigan musbat butun sonlar ketma-ketligi berilgan (0 ketma-ketlikka kirmaydi va uning tugaganligini bildiradi). Ketma-ketlikning o‘rta geometrik qiymatini hisoblaydigan dastur tuzilsin.
47. Haqiqiy x,y,z sonlar berilgan bo‘lsa, munosabat x<y<z to‘gri yoki yo‘qligi aniqlansin.
48. Haqiqiy x,y,z sonlar berilgan bo‘lsa, , munosabat x<y<z to‘gri bo‘lsa bu sonlar ikkilantirilsin aks holda absalyot qiymati bilan almashtirilsin.
49. Uchta ixtiyoriy son berilgan. Tomonlarining uzunliklari shu sonlarga teng bo‘lgan uchburchak yasash mumkinmi?
50. Son o‘qida uchta A, B, C nuqtalar joylashgan, B va C nuqtalardan qaysi biri A nuqtaga yaqin masofada joylashgan va bu masofani chop eting.
51. Berilgan uch xonali son raqamlari orasida bir xillari bormi?
52. Berilganlar turi va o‘zgaruvchilar quyidagicha aniqlangan:

    enum Oy {yan, fev, mar, apr, may, iyn, iyl, avg, sen, okt, noy, dek};

    int d1, d2; Oy m1, m2; bool t;

    Agar d1, m1 sana (yil hisobida) d2, m2 sanadan oldin kelsa, t o‘zgaruvchiga true qiymat, aks holda false qiymat berilsin.
53. Oy m, m1; {Oy turining aniqlanishi 1 masalada berilgan};

    int k, n;

m1 o‘zgaruvchiga qiymat berilsin:

a) m oydan keyingi oyning nomi (dekabrdan keyin yanvar kelishini hisobga olgan holda);

b) m oydan keyingi k-chi oyning nomi;

d) yilning n-chi oyi nomi berilsin.

54. enum Nota {do, re, mi, fa, sol, lya, si};
enum Oraliq {sekund, tersia, qvart, kvint, sekst, septima};
Nota n1, n2; Oraliq i;
Berilgan n1 va n2 (n1≠n2) notalardan tashkil topgan i-oraliq aniqlansin; sekund–bu ikkita qo‘shni (aylana bo‘ylab) notalardan tashkil topgan oraliq (masalan, re va mi, si va do), tersia – bu bitta notadan keyingi oraliq (masalan, fa va lya, si va re) va hokazo.

55. enum Mavsum {qish, bahor,yoz, kuz};
enum Oy {yan,fev,mar,apr,may, iyn,iyl,avg,sen,okt,noy,dek};
Oy m; Mavsum m;
Berilgan m oyga mos keluvchi s-mavsum aniqlansin.

56. enum Davlat {Germaniya,Quba,Laos,Monaqo,Nepal,Polsha};
enum Qita {Osiyo, Amerika, Evropa};
Davlat davlat; Qita: qita;
Davlatning davlat nomi bo‘yicha u joylashgan qit’a nomi qita aniqlansin.

57. enum Birlik {desimetr, kilometr, metr, millimetr, santimetr};
float x; Birlik r;
Berilgan r birlikdagi x o‘zgaruvchining qiymati metrlarda aniqlansin.

58. Berilgan k o‘zgaruvchi qiymati ( ) rim raqamlari ko‘rinishida chop qilinsin.

59. enum Kelishik {bosh, qar, tush, jun, ur_payt, chiq};
enum So‘z {ruchka, qalam, daftar, eshik};
So‘z s; Kelishik k;
Berilgan s so‘zni k kelishik, birlikda chop qilinsin. Masalan, s=daftar va k=jun bo‘lganda “davtarga” so‘zi chop qilinsin.

60. enum Yunalish {shimol, sharq, janub, garb};
enum Buyruq {oldinga, unga, orqaga, chapga};
Yunalish k1, k2;
Buyruq br;
Kema avvaliga k1 yo‘nalish bo‘yicha ketayotgan edi, keyin uning yo‘nalishi br buyruqqa asosan o‘zgartirildi. Kemaning yangi k2 yo‘nalishi aniqlansin.

61. Oy oy; {1 masalaga qaralsin }
    int kun;
    Berilgan oy oyning kunlari soni kun o‘zgaruvchiga o‘zlashtirilsin (yil kabisa yili emas deb hisoblansin).
    int yil; Oy oy; int kun; {Oy turi 1-masalada aniqlangan}
    bool t;
    Agar yil, oy, kun uchlik to‘g‘ri sanani aniqlasa, t o‘zgaruvchiga true qiymat berilsin, aks holda false qiymat berilsin (31 iyun va hakozalarda).
62. int yil,yil1; Oy oy, oy1; int kun, kun1; {Oy turi 1-masalada aniqlangan}
    Berilgan yil, oy, kun sanasi bo‘yicha keyingi kun sanasi – yil1,oy1,kun1 aniqlansin.
63. int yil_kuni, oy_kuni; Oy oy; {Oy turi 1-masalada aniqlangan}
    a) Kabisa yilning oy,oy_kuni sanasiga mos keluvchi kunning yildagi yil_kuni tartib nomeri aniqlansin.
    b) Kabisa yilining hisob bo‘yicha yil_kuni kuniga mos keluvchi oy, oy_kuni - sana aniqlansin.
64. enum Hafta_kuni {yaksh, dush, sesh, chor, paysh, juma, shanba};
    int kun, k13; Oy oy; Hafta_kuni h_kun1, h_kun2; {Oy turi 1-masalada aniqlangan}
    Agar yil kabisa yili bo‘lmasa, va uning 1 yanvari haftaning h_kun1 kuniga to‘g‘ri kelsa, quyidagilar aniqlansin:
    a) kun, oy sanaga mos keluvchi haftaning h_kun2- kuni;
    b) yildagi oyning 13 kuniga mos keluvchi dushanba kunlarining k-soni.
65. Eski yapon kalendarida 60 yillik takrorlanish qabul qilin-gan va bu takrorlanish o‘z navbatida beshta 12 yillik takrorlanish ostilaridan (qismlardan) iborat. Qism takrorlanishlar quyidagi ranglarning nomi bilan belgilangan: yashil, qizil, sariq, oq va qora. Har bir takrorlanish ostining ichidagi yillar hayvonlarning nomi bilan belgilangan: sichqon, sigir, yo‘lbars, quyon, ajdarho, ot, qo‘y, maymun, tovuq, it va to‘ng‘iz (1984 yil – yashil sichqon yili – keyingi takrorlanishning boshi bo‘lgan).
    Eramizning biror yili kiritilib, uning eski yapon kalendaridagi nomini chiqaruvchi programma tuzilsin.

Testlar

33.Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System.IO;
using System;
 class Program
{
    static void Main()
    {
        object a = 1;
        object b = 1;
        if (b == a)
        {
            Console.WriteLine("equal");
        }
        else
        {
            Console.WriteLine("not equal");
        }

    }
}
```

a) Equal

b) Notequal

s) Kompilyatsiyaning bajarilishida xatolik sodir bo‘ladi

d) Xech qanday natija chiqmaydi

34.Taqqoslanayotgan ifodani natijasi qanday qiymat qaytaradi

a) faqat mantiqiy qiymat

b) butun qiymat

s) satr qiymat

d) faqat butun va mantiqiy qiymat

35.C#da qanday ma’lumot tiplari taqqoslashimiz mumkin?

a) int, long, float, double, ushort, decimal

b) char

s) bool

d) yuqoridagi barcha tiplar

36. Qaysi tiplarni taqqoslayotganda to‘g‘ridan-to‘g‘ri biz ularning xotiradagi ikkilik sanoq sistemasidagi ko‘rinishlarini taqqoslaymiz?

a) int, long, float, double, ushort, decimal
b) char
c) bool
d) int, long, float, double, ushort, decimal, char

37. S# dagi mantiqiy operatorlarni belgilang?

a) &&
b) ||, ^
s) !, ^
d) berilgan barcha javoblar to‘g‘ri

38.(2 < 3) && (3 < 4) mantiqiy ifodani natijasi qanday qiymatga ega bщladi?

a) true

b) false

s) 12

d) xech qanday qiymatga ega emas

39.7==5 ifodani natijasi qanday qiymatga ega bщladi?

a) true
b) false
s) 12
d) xech qanday qiymatga ega emas

40.shart operatorlari berilgan qatorni belgilang?

a) “**if**” va “**if – else**”

b) **switch case**

s) **?**

d) barcha javoblar to‘g‘ri

41.“if – else” operatorining ishlash jarayoni qanday?

e) Agar mantiqiy ifodani hisoblaganda **true** ga teng bo‘lsa, **if** operatorining tana qismi bajariladi va **else –** operatori ish bajarmaydi.

f) Agar mantiqiy ifodani hisoblaganda **false** ga teng bo‘lsa, **else –** qismi bajariladi, **if** operatorining tana qismi bajarilmaydi.

s) Agar mantiqiy ifodani hisoblaganda natija rost qiymat qaytarsa, **if** operatorining tana qismi bajariladi va **else –** operatori ish bajarmaydi.

d) Berilgan javoblarni barchasi to‘g‘ri

42.Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System.IO;
using System;
 class Program
{
    static void Main()
    {
        int a = 8;
        int b = 19;
       bool t=true;
        if ((b == a) && (b!=a) || t)
        {
            Console.WriteLine("+");
        }
        else
        {
            Console.WriteLine("-");
        }
      }
```

a) +

b) -

s) /

d) *

43. Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System;
public class Test
{
    public static void Main()
    {
        int i = 3;
        switch (i)
        {
            case 3:
            case 2:
                Console.WriteLine(2);
                break;
            case 1:
            default:
                Console.WriteLine(0);
        }
    }
}
```

a) 2

b) Kompilyatsiya xatoligi vujudga keladi

s) 0

d) 0, 2

44. Console.WriteLine("Exclusive OR: "+ ((2 < 3) ^ (4 > 3))); mantiqiy ifoda natijasi berilgan javobni belgilang?

a) Exclusive OR: False
b) Exclusive : True
c) True
d) Exclusive OR: True

45. **Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
public class Test
{
    public static void Main()
    {
        bool value = !(7 == 5);
        Console.WriteLine(value);
        }
    }
}
```

a) true

b) Kompilyatsiya xatoligi vujudga keladi

s) false

d) 0

46. **Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
int number = 6;
   switch (number)
   {
       case 1:
       case 4:
```

```
                case 6:
                case 8:
                case 10:
                    Console.WriteLine("The number is not
prime!");
                case 2:
                case 3:
                case 5:
                case 7:
                    Console.WriteLine("The number is prime!");
                default:
                    Console.WriteLine("Unknown number!");
break;
            }
```

a) The number is not prime!

b) Kompilyatsiya xatoligi vujudga keladi

s) The number is prime!

d) Unknown number!

6-mavzu. Sikl operatorlari

Reja

1. Kirish

Bu bo‘limda biz **sikl dasturning konstruksiyasini** kodni bo‘laklab bajarish orqali tahlil qilamiz. Shart bo‘yicha takrorlanishlar bo‘yicha qanday algoritm tuzish (**while** va **do - while** sikllari) va qanday qilib **for – sikl**li bilan ishlashni ko‘rib chiqamiz. Ko‘plab misollar bilan ma’lum masalaga qaysi sikl tanlashni va uni qurishning 1 necha usullarini ko‘rib o‘tamiz. Bo‘lim so‘ngida esa **foreach – sikli** konstruksiyasi va biz ichma – ich joylashgan siklardan foydalanishni o‘rganamiz.

1.1. “Sikl” nima?

Dasturda tez - tez jarayonlar takrorlanishiga ehtiyoj seziladi. **Sikl** asosiy dasturning kodini takrorlanishini bajaradi. Kod siklni ichida bir necha marta takrorlanadi yoki berilgan shartli ifoda yolg‘on bo‘lguncha takrorlanadi.

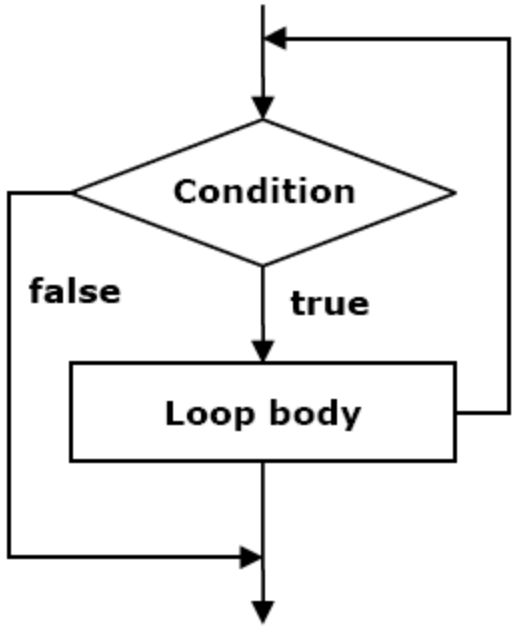
Hech qachon tugamaydigan sikllar **cheksiz sikl**lar deb ataladi. Bu sikldan juda kam holatda foydalaniladi. Agar dastur ishini o‘z vaqtidan oldin tugatmoqchi bo‘lsak, siklning tana qismida **break** operatoridan foydalanamiz.

2. While sikli

Sikllar ichida eng oddiy va eng umumiy **while** siklidir va quyida uni dasturda yozilish shakli ko‘rsatilgan.

```
while (condition)
{
   loop body;
}
```

Yuqoridagi dastur shartida biz mantiqiy qiymat qaytaradigan (**true** yoki **false**) istalgan tipdagi ifodadan foydalanishimiz mumkin. Uning siklning tana qismining necha marta takrorlanishini xisoblashdan iborat va u **shart sikli** deb ataladi. Siklning asosiy tana qismi dasturning asosiy kodini har bir siklda bajarilishi uchun kiritilgan shart har doim **true** bo‘lishi shart. **while sikli** quyidagi blok sxema orqali tavsiflangan.



While sikli dastlab mantiqiy ifodani hisoblaydi va agar u **true (rost)** bo‘lsa sikl jarayoni ketma – ketligi davom etadi. Keyin yana kiritilgan ma’lumotlarni tekshiradi va agar **true (rost)** bo‘lsa sikl yana bajariladi. Shart ifodasi **false** qiymat qaytarmaguncha sikl takrorlanadi. Agar shart ifoda **false** qiymat qaytarsa va o‘sha paytda sikl to‘xtaydi va siklning tana qismidan chiqib, dastur keyingi ishini davom ettiradi.

Agar siklning shart qiymati **false** bo‘lsa, **while siklining** asosiy qismi bajarilmaydi. Agar siklning jarayoni hech qachon to‘xtamasa bu sikl ishini cheksiz davom ettiradi. Bu sikl cheksiz sikl jarayoni deb ataladi.

2.1. while siklidan foydalanish

while siklidan foydalanishga oddiy misolni ko‘rib chiqamiz. Siklning maqsadi ekran (konsol) ga 0 dan 9 gacha sonlarni tartib bo‘yicha chiqarishdan iborat.

```
// Initialize the counter
int counter = 0;

// Execute the loop body while the loop condition holds
while (counter <= 9)
{
   // Print the counter value
   Console.WriteLine("Number : " + counter);
   // Increment the counter
   counter++;
}
```

Bajarilgan kodning natijasi :

```
Number : 0
```

```
Number : 1
Number : 2
Number : 3
Number : 4
Number : 5
Number : 6
Number : 7
Number : 8
Number : 9
```

Ko‘plab misollar berish orqali sikllarning ishlash jarayonini va sikllar orqali yechiladigan ba’zi muamolarni ko‘rsatib o‘tamiz.

2.2. 1 dan N gacha bo‘lgan sonlar yig‘indisini hisoblash

Bu misolda **while sikli** orqali **1** dan **n** gacha bo‘lgan sonlarning yig‘indisini hisoblaymiz. **n** raqami konsoldan o‘qib olinadi:

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
int num = 1;
int sum = 1;
Console.Write("The sum 1");
while (num < n)
{
   num++;
   sum += num;
   Console.Write(" + " + num);
}
Console.WriteLine(" = " + sum);
```

Dastlab biz **num** va **sum** o‘zgaruvchi qiymatlariga 1 ni o‘zlashtiramiz. Agar biz kiritgan **n** son qiymat **num** o‘zgaruvchisini qiymatidan katta bo‘lsa, **num** qiymati 1 ta ga oshiriladi va oldindan beriladigan **sum** o‘zgaruvchisiga qo‘shamiz. Har bir siklda **num** ning qiymatini 1 ga oshadi va bu jarayon **num** ning qiymati **n** bo‘lguncha davom etadi. **sum** ning qiymati 1dan **num** ning qiymatigacha bo‘lgan sonlar yeg‘indisiga teng bo‘ladi. Siklning ichida **num** ning har bir qiymati **sum** ni qiymatiga qo‘shiladi. Konsolga **num** ning har bir qiymati chiqariladi. **sum** ning qiymati sifatida 1 dan **n** gacha sonlar yig‘indisi chiqaradi. Dasturning natijasi quyidagicha bo‘ladi (**n**=17):

```
N = 17
The sum 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 +
14 + 15 + 16 + 17 = 153
```

2.3. Sonni tub yoki tub emasligini aniqlash

Tub sonmi yoki yo‘qligini tekshiradigan dastur tuzamiz. U sonni konsoldan o‘qib olamiz. Biz matematikadan bilamiz, tub sonlar butun musbat sonlar bo‘lib, bu sonlar 1 ga va o‘ziga bo‘linadi. Agar bu son tub sonligini sikl ichida tekshirmoqchi bo‘lsak, uni 2 dan boshlab kiritgan sonimizdan ildiz chiqarilgan songacha berilgan sonimizni bo‘lishimiz kerak bo‘ladi.

```
Console.Write("Enter a positive number: ");
int num = int.Parse(Console.ReadLine());
int divider = 2;
int maxDivider = (int)Math.Sqrt(num);
bool prime = true;
while (prime && (divider <= maxDivider))
{
   if (num % divider == 0)
   {
      prime = false;
   }
   divider++;
}
Console.WriteLine("Prime? " + prime);
```

Biz **divider** o‘zgaruvchisidan mavjud bo‘luvchining qiymatini saqlash uchun foydalandik. Uning dastlabki qiymati 2 ga teng (eng kichik bo‘linuvchi). **maxDivider** maksimum bo‘luvchi bo‘lib, uning qiymati kiritgan sonimizning kvadrat ildiziga teng. Agar bizda **divider** ning qiymati**√num (**kiritgan sonimizning kvadrat ildiz**)** dan katta bo‘lsa, bu holda uni tekshirish foydasiz bo‘ladi, chunki **√num** dan katta sonlarni **√num** dan kichik sonlar orqali xosil qilishimiz mumkin. Biz bu yo‘l bilan siklni ishini kamaytiramiz.

Natija uchun biz mantiqiy tipidagi o‘zgaruvchini **prime** deb belgiladik. Dastlab uning qiymati **true (rost).** Agar kiritgan sonimizni bo‘luvchisi bo‘lsa, **prime** ning qiymatini **false** bo‘ladi.

while siklning shart qismi ikkita mantiqiy operatorlardan tashkil topgan (mantiqiy operator **and**). Siklni bajarilishi uchun ikkila maniqiy ifoda bir vaqatda **true** bo‘lishi shart. Dastur davomida shunday bo‘luvchi topiladiki, **prime** ning qiymati **false** bo‘ladi va shu paytda o‘sha sikl o‘z ishini yakunlaydi. Bu shuni anglatadiki, sikl o‘z ishini kiritgan sonimizga bo‘luvchi topgunicha bajaradi yoki biz kiritgan son 2 dan **√num** gacha bo‘lgan sonlarning hech biriga bo‘linmasligini isbotlaydi. Bu esa sonni tub ekanligini bildiradi.

Agar biz yuqoridagi dasturga 37 va 34 sonlarini kiritsak natijasi quyidagicha bo‘ladi:

```
Enter a positive number: 37
Prime? True
```

```
Enter a positive number: 34
Prime? False
```

3. “BREAK” operatori

Break operatoridan sikldan chiqib ketishda foydalaniladi. Qachonki sikl **break** operatoriga yetganida sikl darhol o‘z ishini tugatadi va siklni tana qismidan chiqib, keyingi qatorga o‘tib ishini davom ettiradi. Sikl ishlash jarayonida uning tana qismida **break** operatori ishlatilgan bo‘lsa, u sikldan chiqib ketishni ta’minlab beradi Qachonki **break** operatori ishga tushganda sikl kodining qolgan qismi bajarilmasdan, sikldan chiqib ketadi. Biz **break** operatori bilan sikldan chiqishni misollar orqali ko‘rsatib o‘tamiz.

3.1. Faktorialni hisoblash

Konsoldan kiritilgan sonning faktorial qiymatini ushbu misolda ko‘ramiz. Bu hisoblash sikl operator va **break** operatoridan foydalanib amalga oshiriladi. Keling matematikadan faktorial nima va u qanday hisoblanishini esga olaylik. **n**! shunday funksiyaki u o‘zidan oldingi barcha sonlarni ko‘paytmasiga teng. Quyidagi misollar orqali tushuntirib o‘tiladi:

- N! = 1 * 2 * 3 … (n-1) * n, for n> 1;
- 2! = 1 * 2;
- 1! = 1;
- 0! = 1.

N! ni undan kichik sonlar faktoriali bilan ifodalash mumkin va quyidagicha bo‘ladi.

N!=(**N**-1)!***N**, 0!=1 => deb qabul qilingan.

n! faktorialni qiymatini hisoblash uchun yuqoridagi ta’rifdan foydalanamiz:

```
int n = int.Parse(Console.ReadLine());

// "decimal" is the biggest C# type that can hold
```

```
integer values
    decimal factorial = 1;
    // Perform an "infinite loop"
    while (true)
    {
      if (n <= 1)
      {
          break;
      }
      factorial *= n;
      n--;
    }
    Console.WriteLine("n! = " + factorial);
```

Biz dastlab **factorial** o‘zgaruvchisiga 1 qiymatini beramiz va **n** ni konsoldan o‘qiymiz. Biz siklning shartini qaytaradigan qiymati **true** dan foydalanib, cheksiz **while** siklini qurdik. Bu dasturda **break** operatoridan **n** ning qiymati 1 dan kichik yoki teng bo‘lganda siklni to‘xtatish uchun foydalanganmiz. Aks holda biz natijaga **n** ko‘paytiraimiz va **n** ning qiymatini 1 taga kamaytiramiz. Hisoblash jarayonida **factorial** o‘zgaruvchisining dastlabki qiymati **n** ga keyingi sikl jarayonida **n** * (**n**-1) ga teng va bu **n** ning qiymati 2 ga teng bo‘lguncha davom etadi. Agar **n** ning qiymati 2 ga teng bo‘lganda olingan **factorial** o‘zgaruvchisining qiymatiga ko‘paytiriladi va keyingi qatorda **n** ning qiymati 1 ga teng bo‘ladi. Keyingi siklga o‘tadi. Bu siklda shart operatori **true** qiymat qaytarib, **break** operatoriga o‘tadi va sikl o‘z ishini tugatadi. **factorial** o‘zgaruvchisining qiymatiga quyidagicha bo‘ladi, ya’ni:

Factorial=n*(n-1)*(n-2)*…*3*2

n=1

Agar biz dastur kodiga konsoldan 10 ni kiritsak, natija quyidagicha bo‘ladi.

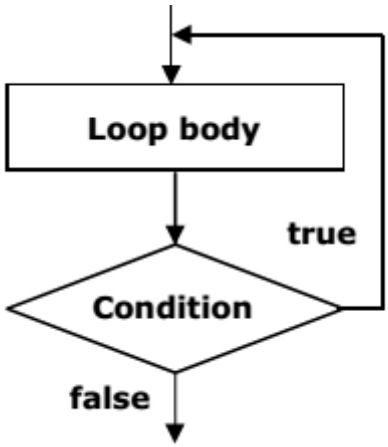
```
10
n! = 3628800
```

4. Do - While sikli

Do - **while** sikl operatori **while** sikliga o‘xshash, lekin u siklning shart qismini tana qismini bajarib bo‘lganidan keyin tekshiradi. Bu tipdagi sikl shartni oxirida tekshiruvchi sikl deyiladi. **Do** - **while** siklining ko‘rinishi quyidagicha:

```
do
{
   executable code;
} while (condition);
```

Do - **while** sikli quyidagi blok sxemadagidek ish bajaradi.



Dastlab tana qismi bajariladi. Keyin uni shart qismi tekshiriladi. Agar shart qismi rost bo‘lsa, siklning tana qismi takrorlanadi, aks holda sikl tugatiladi. Bu mantiqiy jarayon shart qismi yolg‘on qiymat qaytarmaguncha davom etadi. Siklning tana qismi hech bo‘lmaganda bir marta bajariladi. Agar shart doim **true** bo‘lsa, sikl hech qachon to‘xtamaydi va bo‘ sikl cheksiz sikl deb ataladi.

4.1. Do - while siklidan foydalanish

Do - **while** sikli operandlarni hech bo‘lmaganda bir marta ishlashini ta’minlab beruvchi sikl operatoridir.

4.2. Faktorial ni hisolashga misol

Bu misolda biz yana konsoldan kiritilgan **n** sonini faktorialini hisoblashni ko‘rib chiqamiz. Bu mantiqiy misol avvalgi misolga o‘xshish bo‘lib, bunda **while**ning o‘rniga do - **while**dan foydalanilgan.

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
decimal factorial = 1;
do
{
   factorial *= n;
   n--;
} while (n > 0);
Console.WriteLine("n! = " + factorial);
```

Dastlab biz natijani 1 bilan boshlaymiz va **n** dan kichik sonlarga ketma – ket ko‘paytiramiz va **n** ning qiymatini 1 dan kamaytirib boramiz bu jarayon **n** ning qiymati 0 ga teng bo‘lgancha davom etadi. Bu bizga quyidagi natijani beradi, ya’ni **n** * (**n**-1) * … * 1 . Nihoyat biz konsolda natijani chop etamiz. Bu algoritm har doim ko‘paytirishni kamida bir marta bajaradi va shuning uchun ham **n**<=0 bo‘lganda ishlamaydi.

Agar biz dastur kodiga konsoldan 7 ni kiritsak, natija quyidagicha bo‘ladi.

```
n = 7
n! = 5040
```

4.3. Katta sonlarni faktorialini hisoblash

Agar biz yuqoridagi dasturdagi **n** ning qiymatiga katta sonni kiritsak, masalan **n**=100 bo‘lsa, shu sonni faktorialini hisoblashni so‘rasak, biz **System.OverflowException** natija olamiz, ya’ni qiymat decimal tipidan ham oshib ketadi.

```
n = 100
```

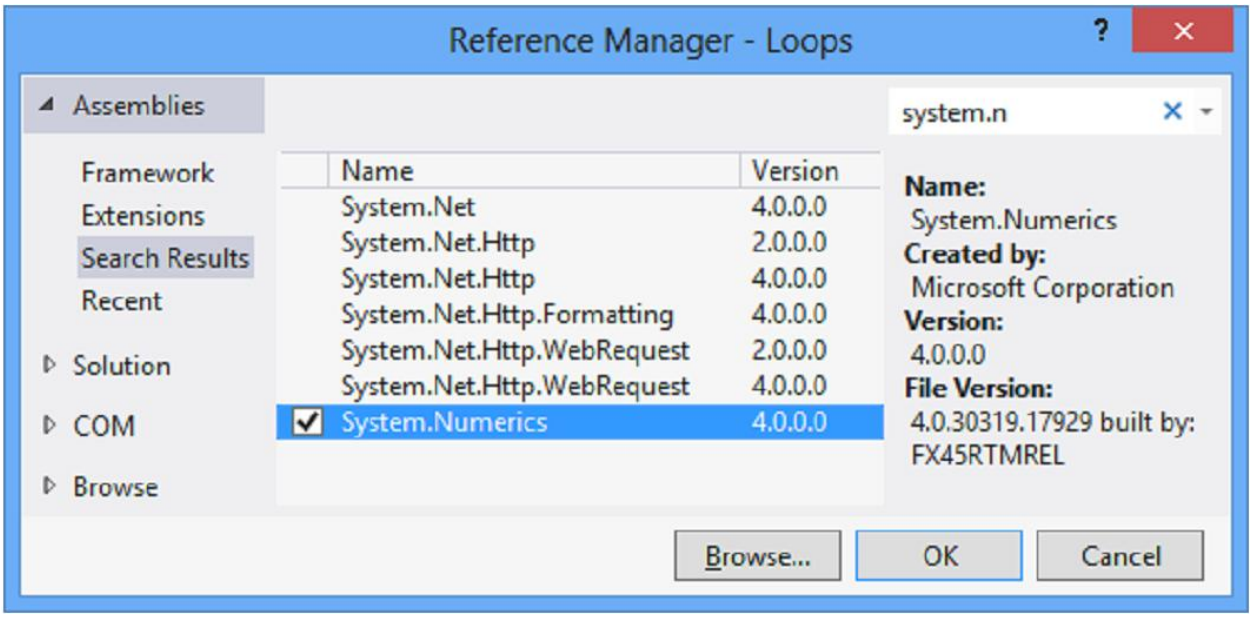
```
Unhandled Exception: System.OverflowException: Value was either
too large or too small for a Decimal.
   at System.Decimal.FCallMultiply(Decimal& result, Decimal d1,
Decimal d2)
   at System.Decimal.op_Multiply(Decimal d1, Decimal d2)
   at TestProject.Program.Main() in
C:\Projects\TestProject\Program
.cs:line 17
```

Demak biz bu kabi misollarni yechishda bu dastur kodini qo‘llay olmaymiz.

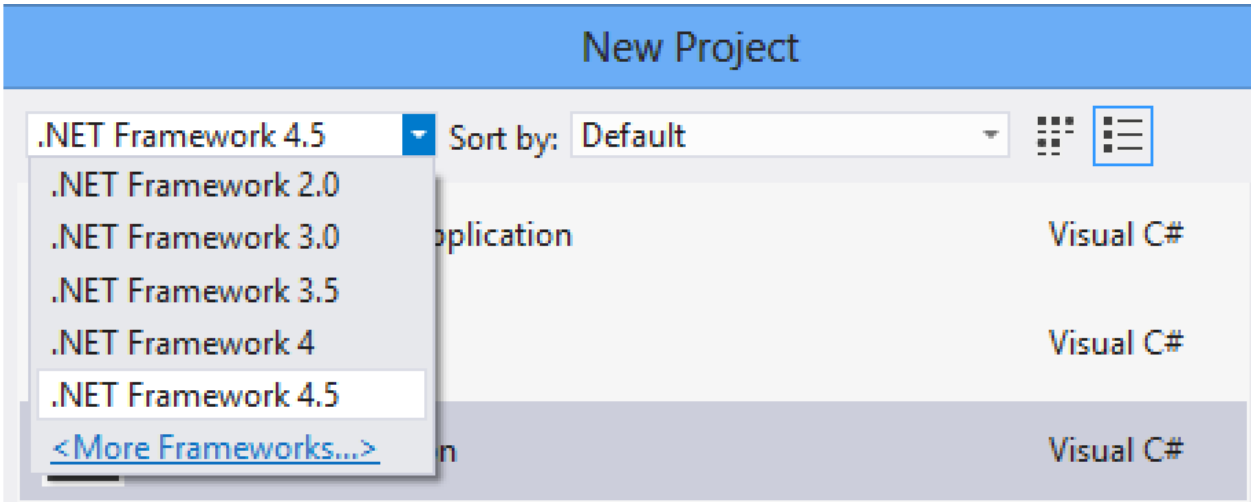
Agar biz 100! ni hisoblashni xoxlasak **BigInteger** tipidan foydalanamiz (**.NET Framework 4.0** versiyasida bor boshqa versiyalarida mavjud emas). Bu tip butun tipning (i**n**t) kengaytirilgan varianti bo‘lib (100,000 ta raqamdan iborat), u juda katta. Ushbu **BigInteger** klassida cheklanmagan miqdordagi raqamlar yozilgan.

BigInteger dan foydalanish uchun, biz o‘z loyihamizdan **add a reference** ni tanlash orqali **System.Numerics.dll** kutubxonani qo‘shishimiz kerak (bu juda katta sonlar bilan ishlashdagi .**N**ET kutubxonasini standarti bo‘lib, VS loyihasida bu ishni bu kutubxonani qo‘shmasdan bajarib bo‘lmaydi) . **Visual Studio** dagi **Solution Explorer** oynasidan sichqonchaning o‘ng tugmasini bosib, **Adding a reference** ni tanlaymiz va u quyidagicha:

Biz ro‘yxatdan **assembly System.Numerics.dll** kutubxonani topamiz va tanlaymiz:



Agar biz bu kutubxonani topa olmasak, bu shuni anglatadiki **Visual Studio** da NET FRAMEWORK 4.0 o‘rnatilmagan yoki bizning **Visual Studio** eski versiyadir va versiyasini yangi versiyaga almashtirishimiz kerak.



So‘ngra biz **“using System.Numerics;”**ni qo‘shishimiz zarur, shunda **decimal** o‘rnini **BigInteger** egallaydi. Dastur esa quyidagicha bo‘ladi:

```
using System;
using System.Numerics;

class Factorial
{
   static void Main()
   {
      Console.Write("n = ");
      int n = int.Parse(Console.ReadLine());
      BigInteger factorial = 1;
      do
      {
         factorial *= n;
         n--;
      } while (n > 0);
      Console.WriteLine("n! = " + factorial);
   }
}
```

Yuqoridagi dastur natijasi **n**=100 bo‘lsa, 100! ning qiymatini quyidagicha olamiz, natijada 158-xonali son hosil bo‘ladi.

```
n = 100
n! =
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000
```

BigInteger da siz 1000! ni ham, 10000! va hattoki 100000! hisoblay olsiz. Bu biroz vaqt oladi, ammo **OverflowException** natija paydo bo‘lmaydi. **BigInteger** klassi katta imkoniyatlarga ega, ammo i**n**t va lo**n**g ga qaraganda sekin ishlaydi. Bizda **"big decimal"** degan klass mavjud emas, lekin **.NET FRAMEWORK** da faqatgina **"big integer"** klass mavjud .

4.4. [N…M] oraliqdagi sonlarni ko‘paytirish

Do-**while** sikli bilan ishlashga doir misol ko‘rib chiqamiz. Bu misolda ko‘zlangan maqsad [**N**…M] oraliqdagi sonlarni ko‘paytirishdan iborat. Quyida ushbu masalaning yechimi ko‘rsatilgan.

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());

Console.Write("m = ");
int m = int.Parse(Console.ReadLine());

int num = n;
long product = 1;
do
{
   product *= num;
   num++;
} while (num <= m);

Console.WriteLine("product[n...m] = " + product);
```

Bu misol kodida **num** ning qiymatlarini **n** dan boshlab, m gacha ketma – ket ravishda oshirib borib, **product** o‘zgaruvchisiga ularning ko‘paytmasini berib boramiz. Biz foydalanuvchidan **n** va m sonlarini kiritishni so‘raymiz va **n,** m dan kichik bo‘lishi kerak. Aks holda dastur natijasi sifatida **n** ni chiqarib beradi.

Agar **n**=2 va m=6 sonlari uchun dasturni ishlatsak, quyidagi natijaga erishamiz:

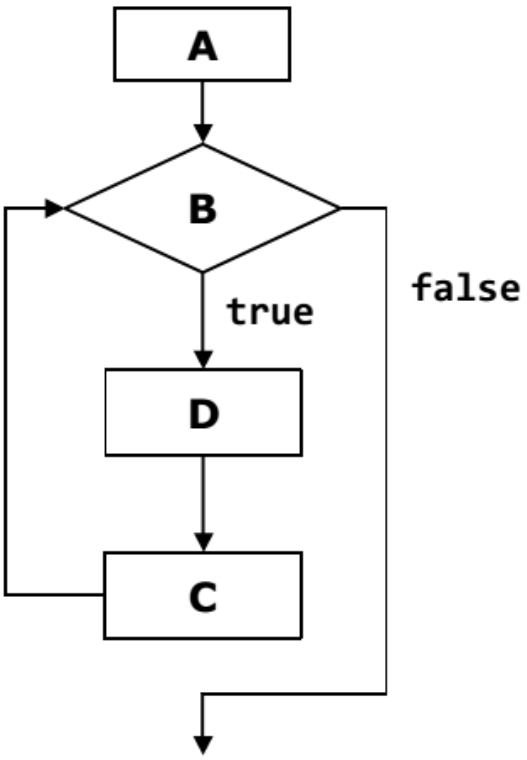
```
n = 2
m = 6
product[n...m] = 720
```

Ehtiyot bo‘ling: ko‘paytma juda tez o‘sadi, shuning uchun siz **long** tipining o‘rniga **BigInteger** dan foydalanishingiz mumkin, chunki aniq natijaga erishish uchun va **long** tipidan oshib ketishi mumkin. Ko‘rib chiqilmagan dastur kodi qiymat oshib ketganda xatolikni ko‘rsatmasdan xato javob chiqarib berishi mumkin. Bundan siz “**tekshirish kalit so‘zi**” orqali xatolikni aniqlashingiz mumkin.

5. For sikli

For sikli **while** va **do** - **while** sikliga qaraganda bir – oz murakkab, lekin boshqa tomondan **for** sikli orqali ancha murakkab masalalarni boshqa sikllarga qaraganda kam kod bilan amalga oshiriladi. Quyida sikl jarayonini sintaksisi va blok sxemasi berilgan.

```
for (A; B; C)
{
    D;
}
```




```
for (int i=0; i<10; i++)
{
    /* loop body */
}
```

Ular (A) boshlash bloki, (B) shart, (D) tana va o‘zgaruvchilarning ishga tushishi(C) bloklarini o‘z ichiga oladi. Biz ular haqida qisqacha ma’lumot beramiz. Oldin keling for siklining dastur kodini ko‘rib chiqaylik:

```
for (initialization; condition; update)
{
   loop's body;
}
```

Ushbu kod sanagichning boshlang‘ich qiymati (misol uchun **int i =0**), mantiqiy shart (**i<10**), hisoblagichning ishga tushishi (i++, yoki i-- bo‘lishi mumkin misol uchun i=i+3) va siklning tana qismidan iborat.

Bu siklning hisoblagichi siklning boshqa tiplaridan farq qiladi.

Ko‘pincha hisoblagich qiymati berilgan dastlabki qiymatdan oxirgi qiymatgacha tartib bo‘yicha o‘zgaradi. Siklda berilgan qiymatni oxiriga borguncha

hisoblaydi misol uchun 1 dan 100 gacha. Kod ishga tushishi boshlanishidan oldin odatda for sikli necha marta takrorlanishlar soni aniq. **For** sikli har birini ola oladi yoki oddiy sikl qiymatlari tartib bilan ko‘tarilishi yoki tushishi yoki biz bergan qiymat asosida oshishi yoki kamayishi mumkin. Ehtimol birinchi siklning qiymati oshadi va boshqasi kamayadi. Hattoki ikkini ko‘paytirish natijasida siklning natijasi 2 dan 1024 gacha o‘zgarishi mumkin. Keyingisida faqat sikl qo‘shishni o‘z ichiga ola olmaydi, ammo boshqa arifmetik jarayonini bajaradi.

For siklining elementlarida hech narsa bo‘lmasligi mumkin va bunda sikl cheksiz davom etadi va u quyidagicha:

```
for ( ; ; )
{
   // Loop body
}
```

5.1. For siklning initsializatsiya qilish

Bu qismda for siklining initsializatsiyasining berilishi:

```
for (int num = 0; …; …)
{
   // The variable num is visible here and it can be used
}
// Here num can not be used
```

Faqatgina birini o‘z ichiga oladi, sikl boshlanishidan oldin. Odatda initsializatsiyasida qiymatlarni hisoblashda foydalanish mumkin va dastlabki qiymatni oladi. Qiymat “**visible**” faqat siklning ichida foydalaniladi. Qismda initsializatsiya jarayoni boshlanishi mumkin va birinchiga qaraganda ko‘proq qiymat qabul qiladi.

5.2. For siklini hisoblash

For siklida hisoblash jarayoni:

```
for (int num = 0; num < 10; …)
{
   // Loop body
}
```

Siklning har biri ichma-ich joylashishidan oldin birinchi (sikl) qiymat hisoblanadi. Siklning shart natijasi **true** bo‘lsa bajariladi , false bo‘lsa sikldan chiqib ketadi va to‘xtaydi.

5.3. Sikl o‘zgaruvchilarining o‘zgarishi

Siklning qiymatining o‘zgarishi for siklining oxirgi elementida amalga oshiriladi:

```
for (int num = 0; num < 10; num++)
{
   // Loop body
}
```

Bu har biri ichma-ich joylashgan kod shundan keyin bajarilishi shart. Unda qiymatning o‘zgarishidan foydalanilgan .

5.4. Siklning asosiy tana qismi

Siklni asosiy qismi blok va koddan iborat. Sikl qiymatlari sikl ishlaydigan blokning ichida e’lon qilinadi.

5.4.1. For siklliga – Misol

for siklliga misol ko‘rib chiqiladi:

```
for (int i = 0; i <= 10; i++)
{
   Console.Write(i + " ");
}
```

Bu misolning natijasi quyidagicha:

```
0 1 2 3 4 5 6 7 8 9 10
```

Bu yerda **for** sikliga boshqa bir misol berilgan va bizda ikkita **i** va **sum** qiymat va ularning qiymati birga teng ammo biz ularni sikl ichida qiymatini oshiramiz:

```
for (int i = 1, sum = 1; i <= 128; i = i * 2, sum += i)
{
  Console.WriteLine("i={0}, sum={1}", i, sum);
}
```

Sikldan quyidagi natija chiqadi:

```
i=1, sum=1
i=2, sum=3
i=4, sum=7
i=8, sum=15
i=16, sum=31
i=32, sum=63
i=64, sum=127
i=128, sum=255
```

5.4.2. N ni M darajasini hisoblashga – misol

n ni m darajasini hisoblash dastur tuzamiz va uni hisoblash uchun **for** siklidan foydalanamiz:

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());
decimal result = 1;
for (int i = 0; i < m; i++)
{
  result *= n;
}
Console.WriteLine("n^m = " + result);
```

Bizda result o‘zgaruvchising qiymati 1 ga teng (**result=1**). Sikl qiymatlarni hisoblash uchun (**int i=0**) dan boshlaydi. Sikl bajarilishi uchun (**i<m**) deb belgilaymiz.

Bunda sikl m marta 0 dan m-1 gacha ish bajaradi. Biz siklning natijasini n gacha ko‘paytiramiz va keyin n ni m ning darajasiga ko‘taramiz (1, 2, … m). Oxirida biz dastur ishini ekranga chiqaramiz.

Bu yerda dastur qanday ishlashini ko‘rishimiz uchun n=2 va m=10 qiymatni berib ko‘ramiz:

```
n = 2
m = 10
n^m = 1024
```

5.5. Bir qancha o‘zgaruvchilar bilan for sikli

Biz for siklida qiymatlarni qanday ko‘paytirilishini ko‘rdik. Bu misolda bizda ikki o‘zgaruvchi bor. Birinchi o‘zgaruvchi 1 dan oshib boradi va ikkinchisi 10 dan kamayib boradi:

```
for (int small=1, large=10; small<large; small++, large--)
{
  Console.WriteLine(small + " " + large);
}
```

Natija quyidagicha bo‘ladi:

```
1 10
2 9
```

```
3 8
4 7
5 6
```

5.6. “continue” operatori

continue operatori xuddi **break** operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin programmani qurilmadan chiqib ketmasdan takrorlashning keyingi qadamiga “sakrab” o‘tishini tayinlaydi. Quyidagi misolda bu operatordan qanday foydalanishni ko‘rib chiqamiz.

Biz barcha 1 dan n gacha bo‘ldan toq sonlarni hisoblaymiz va 7 ga bo‘linadiganlarini olmasdan **for** sikli orqali davom ettiramiz:

```
int n = int.Parse(Console.ReadLine());
int sum = 0;
for (int i = 1; i <= n; i += 2)
{
   if (i % 7 == 0)
   {
      continue;
   }
   sum += i;
}
Console.WriteLine("sum = " + sum);
```

Dastlab siklni qiymati bilan birinchi [1…n] toq son qiymatini hisoblaymiz. Eng kichik toq son qiymati 1 ga teng. **i <= n** dan oshib ketmasligi siklning har bir amalida tekshiriladi. Bu ifodada biz qiymatni 2 ga oshiramiz va toq songa ega bщlamiz. 7 ga bo‘linishi sikl ichida tekshiriladi. Biz sikl ichida **continue** operatorini chaqirsak bu yana siklning asosiy qismiga o‘tadi. Agar o‘zgaruvchi 7 ga bo‘linmasa u **sum** ning qiymatiga **i** ga qo‘shib boraveradi.

n=11 ga bo‘lgandagi natijani ko‘rishimiz mumkin:

```
11
sum = 29
```

5.7. foreach sikl operatorlari

foreach sikli C/C++/C# dasturlash tillari oilasi uchun yangi ammo bu **VP va PHP** dasturchilari uchun yaxshi tanish. Bu dasturning konstruksiyasida massivning barcha elementlarini va ro‘yxatdan yoki elementlar yig‘indisida (**IEnumerable**) ishlatiladi. Agar elementlari indekslanmagan bo‘lsa ham to‘plamning barcha elementlari orqali ifodalash mumkin.

Biz massivlarga “**massiv**” bo‘limida to‘xtalamiz, ammo hozirga dastur uchun boshqa elementlar yoki raqamlar ketma – ketligini tassavur qilishimiz kerak.

foreach sikli tuzilishi:

```
foreach (type variable in collection)
{
   statements;
}
```

Biz ko‘rishimiz mumkin bu **for** sikliga qaraganda oddiyroq va shuningdek uning ayzalligi sizga berilga to‘plamning barcha elementlarini yozilganlarini saqlab boradi. **Foreach** opreatoridan qanday foydalanishga misol:

```
int[] numbers = { 2, 3, 5, 7, 11, 13, 17, 19 };
foreach (int i in numbers)
{
   Console.Write(" " + i);
}
Console.WriteLine();
string[] towns = { "London", "Paris", "Milan", "New York" };
foreach (string town in towns)
{
   Console.Write(" " + town);
}
```

Bu misoda biz raqamlar masssivini yaratdik va ular **foreach** sikli bilan ishlaydi va uning qiymatini konsol ga chiqaradi. Shaharlar nomi **satr** tipidagi massivda e’lon qilindi va uning elementlarini konsolga **foreach** sikli orqali chiqariladi. Misol natijasi quyidagicha bo‘ladi:

```
2 3 5 7 11 13 17 19
London Paris Milan New York
```

6. Ichma – ich joylashgan sikl operatorlari

Ichma – ich sikllar dastur konstruksiyasida sikllar bir birini ichida joylashadi. Dasturda birinchi sikl ko‘p bajariladi va undan keyingi sikllar oz bajariladi. Ichma – ich siklni ko‘ramiz:

```
for (initialization, verification, update)
{
  for (initialization, verification, update)
  {
```

```
    executable code
  }
  …
}
```

Birinchi sikl initsializatsiyadan keyin uning asosiy qismi bajarilishni boshlaydi va ikkinchi siklga o‘tadi. U qiymati initsializatsiya qilinadi va tekshiriladi keyin kod bajariladi va sikl qachon **false** qiymati qaytarmaguncha davom etadi. Birinchi **for** siklining ikkinchi ifodasi ishlaydi. Uning qiymati oshiriladi va ikkinchi sikl yana bajariladi. Tashqaridagi siklga chiqguncha ichkaridagi sikllar ko‘p marta bajariladi.

6.1. Uchburchak shaklda chiquvchi – misol

Quyidagi muammoni yechamiz: Berilgan n ta sondan konsolga raqamlarni uchburchak sifatida chiqaring:

```
1
1 2
1 2 3
…
1 2 3 … n
```

Biz ikkita sikl muammosini yechamiz. Oxirgi sikl qatorlarni va ichidagi bittasi ularning elementlari belgilaydi. Birinchi qatorda biz “1” bir chiqarishimiz kerak. Ikkinchi siklda “1 2” chiqishi kerak. Biz qatorlar va raqamlar o‘zaro bog‘liqligini ko‘ramiz. Ichki sikl qanday tuzilganini ko‘rsatib o‘tamiz:

Siklning qiymatini 1 bilan boshlaymiz (birinchi raqamni chiqaramiz): **col=1;**
Takrorlanish qatorlarga bog‘liq bo‘ladi: **col<=row;**
Har bir siklning ichida siklning qiymatini bittaga oshiramiz

Asosiysi biz **for – sikl**ni 1 dan n gacha bajaramiz va boshqa (ichki) sikl ichiga tashlaymiz, qatorlar raqamini 1 dan boshlab bittaga oshirib boramiz. Tashqi va ichki sikllar tenglikda boradi va ular ustunni tashkil qiladi.

Oxirida biz shu kodga erishamiz:

```
int n = int.Parse(Console.ReadLine());
for (int row = 1; row <= n; row++)
{
   for (int col = 1; col <= row; col++)
   {
      Console.Write(col + " ");
   }
   Console.WriteLine();
}
```

Agar hozir buni bajarsak, n=7 bo‘lgandagi qiymatni natijasini ko‘ramiz:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
```

Ye**slatma:** Agar n>9 uchburchakda kamchilik bo‘ladi. Buni o‘ylab ko‘ring!

6.2. Tub sonlarga – misol

Ichma – ich siklga boshqa bir misol ko‘ramiz. Bizning maqsadimiz [n…m] orasidagi tub sonlarni konsolga chiqarishdan iborat. Biz oraliqni **for sikli** bilan cheklaymiz va tub sonlarni tekshirish uchun **while sikl**lidan foydalanamiz:

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());

for (int num = n; num <= m; num++)
{
   bool prime = true;
   int divider = 2;
   int maxDivider = (int)Math.Sqrt(num);
   while (divider <= maxDivider)
   {
      if (num % divider == 0)
      {
         prime = false;
         break;
      }
      divider++;
   }
   if (prime)
   {
      Console.Write(" " + num);
   }
}
```

for sikli n, n+1, … m tubligini oxirida tekshiradi. **num** qiymati tubligini har bir tashqi siklda tekshiriladi. Mantiqan biz tub sonlarni bilamiz. Dastlab biz **prime** qiymatini **true (rost)** deb $[2...\sqrt{num}]$ kiritamiz. Ichki **while siklida** har bir son tekshiriladi agar **num** ning bo‘luvchisi bo‘lsa davom etadi aks holda **prime false** ga teng bo‘ladi. **while sikli** tugagandan keyin mantiqiy **prime** qiymati boshqa son yo‘qligini ko‘rsatadi va o‘sha son konsolga chiqariladi.

Agar n=3 va m=75 bo‘lsa quyidagi natija chiqadi:

```
n = 3
m = 75
 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
```

6.3. Baxtli sonlarga – misol

Boshqa misol orqali biz ikki va undan ortiq ichma – ich joylashgan sikllarni ko‘rishimiz mumkin. Bizning maqsadimiz ABCD sonlarini aniqlash. Agar A+B=C+D bo‘lsa (biz bularni omadli raqamlar) deb ataymiz. Biz buni to‘rtta **for** siklida bajaramiz. Birinchi sikl minglarni tasvirlaydi. Sikl 1 dan boshlanadi va 0 da to‘xtaydi. Biz yuzlar, o‘nlar va minglar deb belgilaymiz. Biz omadli raqamlarni ichki sikldan tekshirib kelamiz va oxirida konsolga chiqaramiz. Bu yerda dastur kodini кщrib chiьiamiz:

```
for (int a = 1; a <= 9; a++)
{
  for (int b = 0; b <= 9; b++)
```

```
  {
    for (int c = 0; c <= 9; c++)
    {
      for (int d = 0; d <= 9; d++)
      {
        if ((a + b) == (c + d))
        {
          Console.WriteLine(
            " " + a + " " + b + " " + c + " " + d);
        }
      }
    }
  }
}
```

Bu yerda natija qisman chiqarilgan (butub natija juda uzun):

```
1 0 0 1
1 0 1 0
1 1 0 2
1 1 1 1
1 1 2 0
1 2 0 3
1 2 1 2
1 2 2 1
…
```

6.4. 6/49 Lotereya masalasi

Bu misolda biz “6/49” o‘yinida mumkin bo‘lgan barcha kombinatsiyalar mavjudligini topamiz. Biz 6 dan farq qiluvchi hamma sonlarni topamiz va ko‘rsatamiz [1…49] oralig‘idagi 6 talik turli raqamlar o‘rin almashtirishlar qiymatlar sonini topib chop etishimiz kerak. Biz 6 ta for siklidan foydalanamiz. Oldingi misolga o‘xshamagan holda sonlar takrorlanmaydi. Sonlar takrorlanmasligi uchun har bir n – ketma – ketlik n-1 – ketma –ketlikdan katta bo‘ladigan qilib yozishimiz kerak. Shuning uchun mavjud sikl 1 dan boshlamaydi lekin oldingi siklga +1 ni qo‘shgan sondan boshlaydi. Biz birinchi siklni 44 ga yetguncha olib borishimiz kerak.keyingisi 45 ga yetguncha va hakozo…agar biz hamma sikllarni 49 gacha qilsak biz bu kombinatsiyalarda 1 xil qiymat olamiz. Shu sababli har bir sikl o‘zidan oldingi siklga 1 ni qo‘shib oladi. Dastur quyidagicha

```
for (int i1 = 1; i1 <= 44; i1++)
{
```

```
   for (int i2 = i1 + 1; i2 <= 45; i2++)
   {
      for (int i3 = i2 + 1; i3 <= 46; i3++)
      {
         for (int i4 = i3 + 1; i4 <= 47; i4++)
         {
            for (int i5 = i4 + 1; i5 <= 48; i5++)
            {
               for (int i6 = i5 + 1; i6 <= 49; i6++)
               {
                  Console.WriteLine(i1 + " " + i2 + " " +
                     i3 + " " + i4 + " " + i5 + " " + i6);
               }
            }
         }
      }
   }
}
```

Hammasi to‘g‘ri berilgan. Keling yuqoridagi dasturni ishlatib ko‘ramiz. Bu dastur ishlaydiganga o‘xshaydi ammo bir muammo bor. Kombinatsiya juda ko‘p va dastur to‘xtamaydi (u juda sekin va juda ko‘p vaqt oladi). Shundan “6/49” tanlanganini tushinish mumkin. Chunki bunda juda ko‘p kombinatsiyalar mavjud. Yuqoridagi o‘yinda hamma kombinatsiyalarning sonini hisoblashning tezligini kutib

turibmiz, vaholanki uni chop etish emas. Bu tezlik chop qilingan natijalarni qisqartirishi natijasida dastur kutilmagan darajada tez tugaydi.

Katta textlarni konsolda chiqarish juda sekin amalga oshadi. Zamonaviy kompyuterlarda bir sekundda 300,000,000 tezlikda ishlaydi (2012) ammo sekundiga 10,000-20,000 oralig‘idagi text qatorlarni chiqarib beradi.

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

- **Sikl operatori** – takrorlash sharti deb nomlanuvchi mantiqiy ifodani rost **(true)** qiymatida programmaning ma’lum bir qismlaridagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajarishni amalga oshiradi (itaratsiya)
- **Break** – operatori sikldan chiqib ketishda foydalaniladi
- **BigInteger –** katta sonlar bilan ishlashga mo‘ljallangan klass
- **continue** – bu operatori xuddi **break** operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin programmani qurilmadan chiqib ketmasdan takrorlashning keyingi qadamiga “sakrab” o‘tishini tayinlaydi
- **foreach –** bu sikl operatori **C/C++/C#** dasturlash tillari oilasi uchun yangi ammo bu **VP va PHP** dasturchilari uchun yaxshi tanish dasturning konstruksiyasida massivning barcha elementlarini va ro‘yxatdan yoki elementlar yig‘indisida (**IEnumerable**) ishlatiladi
- **Write(…) –** konsolga argumentlarni chiqarish
- **WriteLine(…) –** konsolga ma’lumotlarni chiqarish va keyingi qatorga o‘tish

- **Console.ReadKey() –** klavish bosilguncha ekranni ushlab turadi
- **KeyChar –** kiritilgan belgini ushlab turadi
- **int.Parse(string) –** satrni butun tipga o‘tkazadi
- **Convert klassi –** metodidan foydalangan holda boshqa tipga o‘tkazish uchun ishlatiladi
- **try-catch –** hatoliklarni ushlab qolish uchun ishlatiladi
- **while, do – while, for, foreach** – sikl operatorlari

Topshiriqlar

1. Konsolda 1 dan n gacha sonlarni chiqaradigan dastur tuzing. (n kiritilsin).
2. 1 dan n gacha sonlar berilgan. Ekranga 3 ga va 7 ga bo‘linmaydigan sonlarni bir vaqtda chiьiaruvchi dastur tuzing. (n soni kiritilsin).
3. Konsolda bir qancha butun son tipidagi sonlarni eng kichik va eng kattasini chiqaradigan dastur tuzing.
4. Kartaning standard doskasidan barcha ehtimoliylarini chiqaradigan dastur tuzing. Jokerslarsiz (u yerda 52 karta bor: 13 juft kartaning 4 tasi Jokers).
5. N ta raqamdan iborat fibonachchi sonlarini chiqaruvchi dastur tuzing. Fibanochchi ketma – ketligi: 0,1,1,2,3,5,8,13,21,34,55,89,144,233,337,…
6. N!/C! ni hisoblaydigan dastur tuzing. (1<C<N).
7. N!*C!/(N-C)! hisoblaydigan dastur tuzing. (1<N<C).
8. Quyidagi formulani hisoblovchi dastur tuzing.

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} \quad n \geq 0$$

9. N va x berilgan S ni toping.

$$S = 1 + {}^{1!}/_{x} + {}^{2!}/_{x^2} + \ldots + {}^{n!}/_{x^n}$$

10. Quyida matritsa berilgan (N<20).

N = 3

1	2	3
2	3	4
3	4	5

N = 4

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

11. N kiritilganda shu ko‘rinishada hosil bo‘ladigan matritsa dasturini tuzing.

12. Oxirgi raqamlari 0 lar bilan tugaydigan faktorial ga dastur tuzing. Masalan:

N = 10 -> N! = 3628800 -> 2

N = 20 -> N! = 2432902008176640000 -> 4

13. O‘nlikdan ikkilikka o‘tuvchi dastur tuzing.
14. Ikkilikdan o‘nlikka o‘tuvchi dastur tuzing.
15. O‘nlikdan o‘n oltilikka o‘tuvchi dastur tuzing.
16. O‘n oltilikdan o‘nlikka o‘tuvchi dastur tuzing.
17. N berilgan. 1 dan N gacha bo‘lgan sonlarni ixtiyoriy tarzda chiqaradigan dastur tuzing.
18. Berilgan a va b ni EKUB va EKUKkini toping. EKUK (a,b)=|a*b|/EKUB(a,b).
19. Spiralsimon matritsa dasturini tuzing. Masalan:

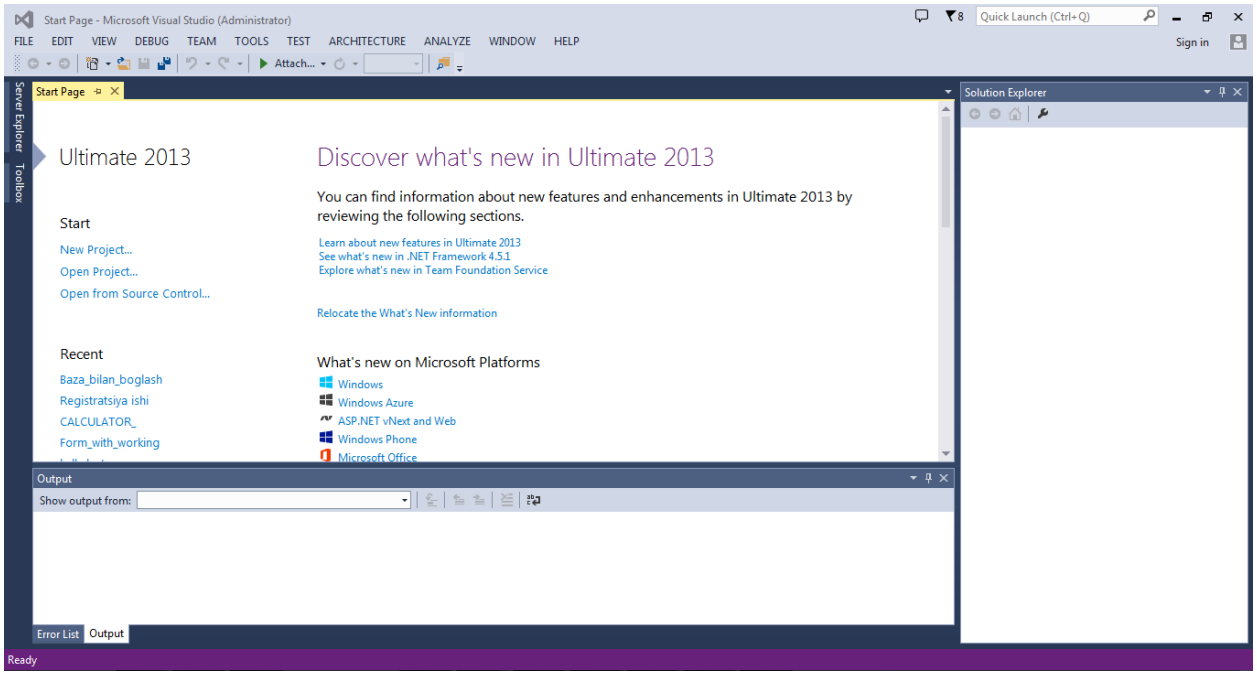
n=4:

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Amaliy mashg‘ulot.

1-misol. $S = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{50}$ yig‘indini hisoblash dasturini tuzing.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi va yangi loyiha yaratamiz.



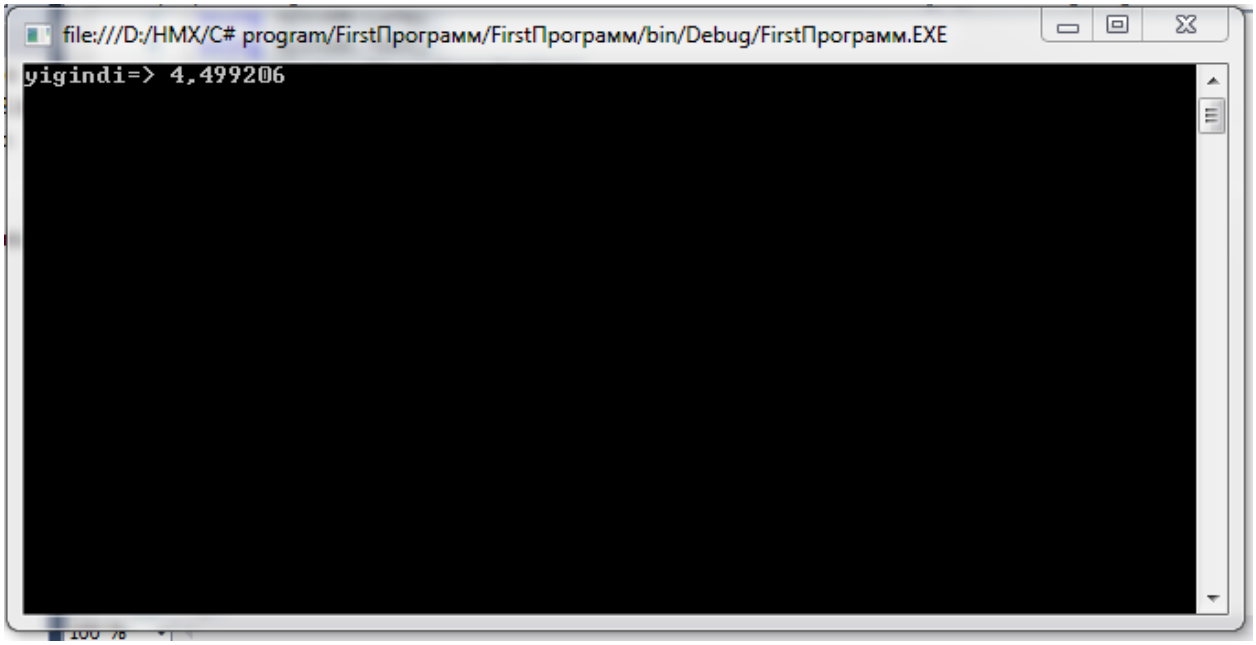
Endi berilgan masala kodini

```
static void Main(string[] args)
 {
     }
```

S# dagi asosiy metod blokiga yoziladi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
         float s = 0;
         for (float i = 1; i <= 50; i++ )
         {
            s += 1 / i;
         }
         Console.WriteLine("yigindi=> {0}",s);
         Console.ReadKey();
      }
   }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasida quyidagi 1.1-rasm rasmda keltirilgan natijaga erishamiz.

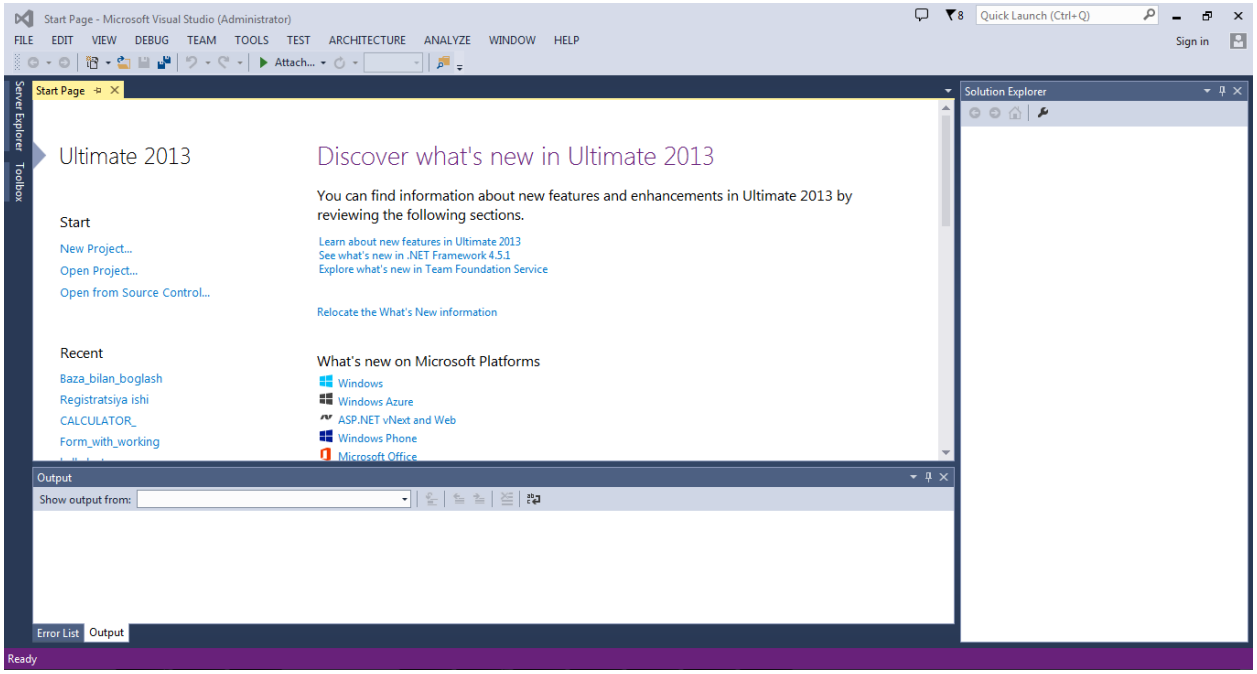


1.1-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

2-misol. Konsoldan 2 ta son kiritiladi. Sonlar ustida amallar bajaruvchi menyu xosil bo‘lsin va foydalanuvchi tomonidan tanlangan menyu asosida amal bajaruvchi dastur tuzing.

Visual Studio 2013 (VS 2013) tizim ishga tushiriladi, **2.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



2.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **SecondProgramm** kabi kiritib, **OK** tugmasini bosamiz.

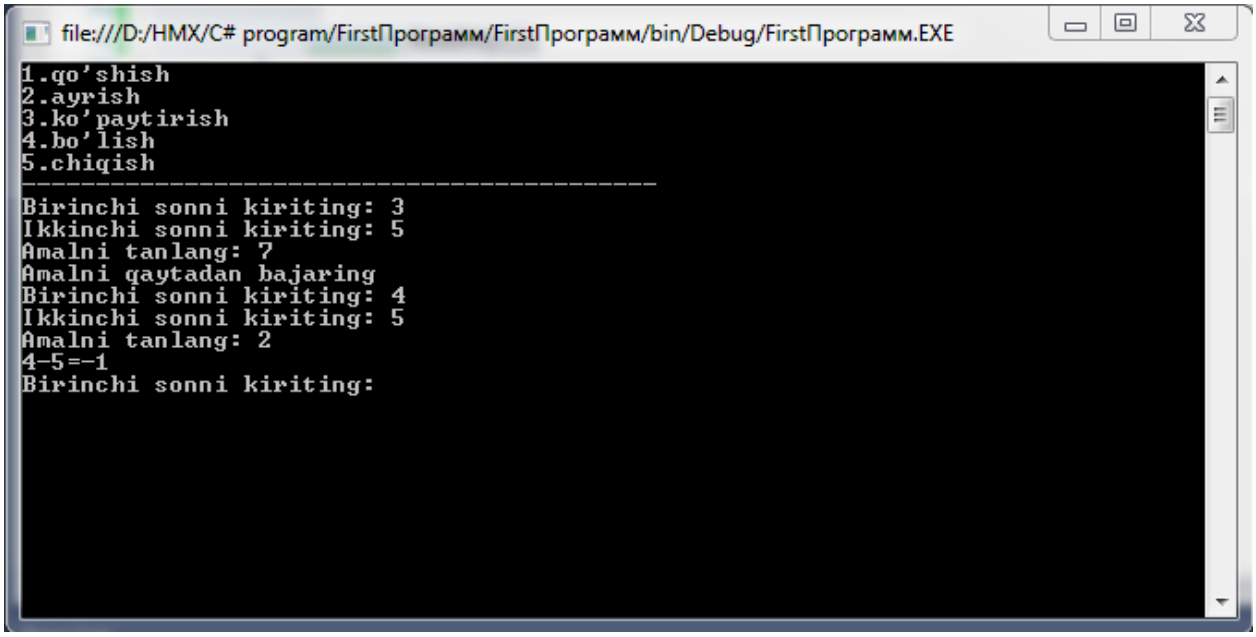
Endi berilgan masala kodini kiritamiz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SecondProgramm
{
```

```
class Program
{
    static void Main(string[] args)
    {

Console.WriteLine("1.qo'shish\n2.ayrish\n3.ko'paytirish\n4.bo'lish\n5.chiqish\n----------
-----------------------------------");
        while(true)
        {
            Console.Write("Birinchi sonni kiriting: ");
            float a = float.Parse(Console.ReadLine());
            Console.Write("Ikkinchi sonni kiriting: ");
            float b = float.Parse(Console.ReadLine());
            Console.Write("Amalni tanlang: ");
            int s=int.Parse(Console.ReadLine());
            if(s==1)
            {
                Console.WriteLine("{0}+{1}={2}",a,b,a+b);
            }
            else
            {
                if(s==2)
                {
                    Console.WriteLine("{0}-{1}={2}",a,b,a-b);
                }
                else
                {
                    if (s == 3)
                    {
                        Console.WriteLine("{0}*{1}={2}", a, b, a * b);
                    }
                    else
                    {

                        if (s == 4)
```

```
                {
                    Console.WriteLine("{0}/{1}={2}", a, b, a / b);
                }
                else
                {
                    if(s==5)
                    {
                        System.Environment.Exit(-1);
                    }
                    else
                    {
                        Console.WriteLine("Amalni qaytadan bajaring");
                    }

                }

            }

        }

    }
    Console.ReadKey();
    }
  }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. ***a*** *va* ***b*** sonlarni kiritamiz va 1.3-rasmda keltirilgan natijaga erishamiz.



2.2-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliy mashg‘ulot topshiriqlari

Boshlang‘ich bosqich

Topshiriq: bloksxema va tanlash operatorini qo‘llab masalaga muvofiq dastur yozing.

1. A sonini butun **N** darajaga oshiring
2. Berilgan sonni faktorialini hisoblang. **N** sonining faktorialini quyidagi formula bo‘yicha xisoblang: N!=1*2*3*…N
3. 1 dan **N** gacha bo‘lgan sonlar kvadratlari summasi S ni xisoblang.
4. 1 dan **N** gacha bo‘lgan juft sonlar kvadratlari va toq sonlar kublari summasi S ni xisoblang.

5. 5 ga karrali bo‘lmagan va 3 ga karrali bo‘lgan sonlarni toping, shuningdek 5 ga karrali bo‘lmagan va 3 ga karrali bo‘lgan sonlarni summasini toping.
6. 1 dan **N** gacha berilgan sonlarning 5 ga karralilarini toping
7. Berilgan natural son ikkining darajasi bo‘la oladimi?
8. Berilgan sonni ko‘paytuvchilarga ajrating.
9. Bir sonini ham inobatga olgan holda, bo‘luvchilari yig‘indisiga teng bo‘lgan son tub son deyiladi. 2 dan h oralig‘igacha bo‘lgan sonlar ichidan tub sonlarni toping va chop qiling.
10. M dan **N** gacha bo‘lgan sonlarning kvaratlari yig‘indisini toping.
11. Berilgan **M** va **N** sonlari oralig‘idagi toq sonlarning kvaratlari yig‘indisini toping.
12. -80 dan 80 gacha bo‘lgan, sonlardan 7 ga karrali butun sonlarning toqlarini ko‘paytmasini toping.
13. -10 dan 10 gacha bo‘lgan, 9 ga karrali butun musbat sonlarning yig‘indisini toping.
14. 100 dan 800 gacha diapazonda bo‘lgan **N** natural soni berilgan. **N** dan katta bo‘lgan 3 xonali sonlarni qanchaligini aniqlang.
15. **N** natural soni berilgan. **N** dan kichik bo‘lgan barcha natural sonlarni va o‘zaro tub(agar sonni ±1 dan boshqa bo‘luvchisi bo‘lmasa) sonlarni aniqlang.
16. **p** va **q** butun sonlar berilgan. **p** bilan o‘zaro tub bo‘lgan **q** sonini barcha bo‘luvchilarini toping.
17. **N** natural soni berilgan. Bu sonni barcha oddiy bo‘luvchilarini toping.
18. Birinchi 100 ta tub sonni toping. (tub son o‘ziga va 1 ga bo‘linadi)
19. **M** va **N** qiymatlar oralig‘idagi juft sonlarning kvadratini ko‘paytmasini toping.
20. Natural son **n** berilgan. xisoblang: $1\cdot2+2\cdot3\cdot4+\ldots+n(n+1)\cdot\ldots\cdot2n$.
21. Natural sonlardan **N** toq sonlarni kubini bo‘linmasini toping.
22. **M** dan **N** gacha bo‘lgan natural sonlarning bo‘linmasini kvadratini toping.
23. -20 dan 20 gacha bo‘lgan, 5 ga karrali butun manfiy sonlarning yig‘indisini toping.
24. 4 ga karrali va 100 dan kichik bo‘lgan butun musbat sonlarning yig‘indisini toping.
25. 1 dan **N** gacha bo‘lgan barcha sonlarning kvadratlari bo‘linmasini hisoblovchi dastur tuzing.
26. 100 dan 200 ni ham hisobga olgan holda barcha toq sonlarni chiqaruvchi dastur tuzing.
27. **M** dan **N** gacha bo‘lgan sonlarning kublarini ko‘paytmasini toping.

28. **n**, **m** natural sonlari berilgan. Barcha **n** dan kichik, kvadratlari yig‘indisi **m** ga teng bo‘lgan natural sonlari aniqlang.
29. Berilgan **M** va **N** o‘zgaruvchilarning qiymatlari oralig‘idagi toq sonlarning kvadratlari ko‘paytmasini toping.
30. Berilgan **M** va **N** o‘zgaruvchilarning qiymatlari oralig‘idagi juft sonlarning kvadratlari yig‘indisini toping.

Testlar

1. Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System.IO;
using System;
 class Program
{
   static void Main()
   {
      object a = 1;
      object b = 10;
      object s = 0;
      while(b == a)
      {
            s=s+a;
            a++;
      }
            Console.WriteLine(s);

   }
}
```

a) 0
b) 55
c) Kompilyatsiyaning bajarilishida xatolik sodir bo‘ladi
d) Xech qanday natija chiqmaydi

2. Siklning shart ifodasi qanday qiymat qaytaradi

a) faqat mantiqiy qiymat

b) butun qiymat

s) satr qiymat

d) faqat butun va mantiqiy qiymat

3. Cheksiz sikl nima?

a) asosiy dasturning kodini takrorlanishini bajaradi

b) Hech qachon tugamaydigan sikllar

s) bunday sikllar mavjud emas

d) yuqoridagi a) va b) javoblar to‘g‘ri

4. While sikli dastlab nimani bajaradi?

a) mantiqiy ifodani

b) siklni tana qismini

s) sonlar yig‘indisini

d) a) va b)

5. **Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
            int counter = 0;
            while (counter < 9)
```

```
            {
                Console.WriteLine("Number : " + counter);
                counter++;
            }
        }
    }
}
```

a) 0 dan 9 gacha sonlarni tartib bo‘yicha chiqaradi

b) 0 dan 8 gacha sonlarni tartib bo‘yicha chiqaradi

s) 1 dan 9 gacha sonlarni yig‘indisini chiqaradi

d) 0 dan 7 gacha sonlarni tartib bo‘yicha chiqaradi

6. Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
    class Program
    {
        static void Main(string[] args)
        {

            Console.Write("n = ");
            int n = int.Parse(Console.ReadLine());
            int num = 1;
            int sum = 1;
            Console.Write("The sum 1");
            while (num < n)
```

```
        {
            num++;
            sum += num;
            Console.Write(" + " + num);
        }
        Console.WriteLine(" = " + sum);

    }
  }
}
```

a) n tub sonmi yoki yo‘qligini tekshiradi

b) Kompilyatsiya xatoligi vujudga keladi

s) 0

d) 1

7. while (true)

{

Operatorlar;

}

Bu opreator nima deb ataladi

e) Oddiy sikl opreatori
f) Cheksiz sikl operatori
g) Shartli sikl opreatori
h) True sikl opreatori

8. **Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
```

```
{
   class Program
   {
      static void Main(string[] args)
      {

         int n = int.Parse(Console.ReadLine());

         decimal factorial = 1;

         while (true)
         {
          if (n <= 1)
          {

            break;
          }
          factorial *= n;
          n--;
         }
         Console.WriteLine("n! = " + factorial);

      }
   }
}
```

a) Faktorialni hisoblash programmasi

b) Kompilyatsiya xatoligi vujudga keladi

s) n=3 kiritsak, n!= 6 natijani qaytaradi

d) a) va s) javoblar to‘g‘ri

9. **Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {

         int n = 5;

         decimal factorial = 1;

         while (true)
         {
          if (n <= 1)
          {

            break;
          }
          factorial *= n;
          n--;
         }
         Console.WriteLine(factorial);

      }
   }
}
```

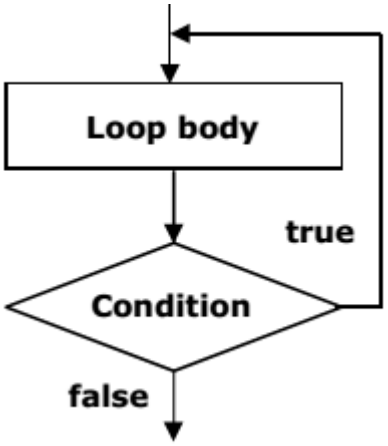
a) n!=120

b) Kompilyatsiya xatoligi vujudga keladi

s) 120

d) Unknown number!

10. **Do** - **while** sikl operatori **while** sikl operatorini farqi nimada.
    a) **Do** - **while** sikl operatori **while** sikliga o‘xshash, lekin u siklning shart qismini tana qismini bajarib bo‘lganidan keyin tekshiradi
    b) **while** sikl operatori **Do** - **while** sikliga o‘xshash, lekin u siklning shart qismini tana qismini bajarib bo‘lganidan keyin tekshiradi
    c) Bu sikl operatorlarni hech qanday farqi yo‘q

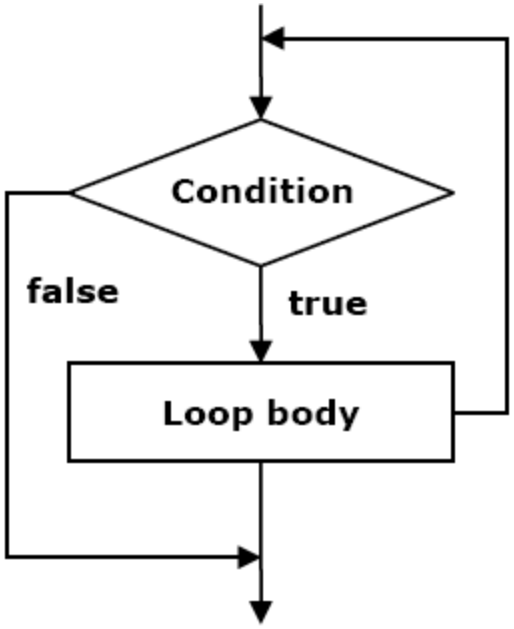


    d) To‘g‘ri javob berilmagan

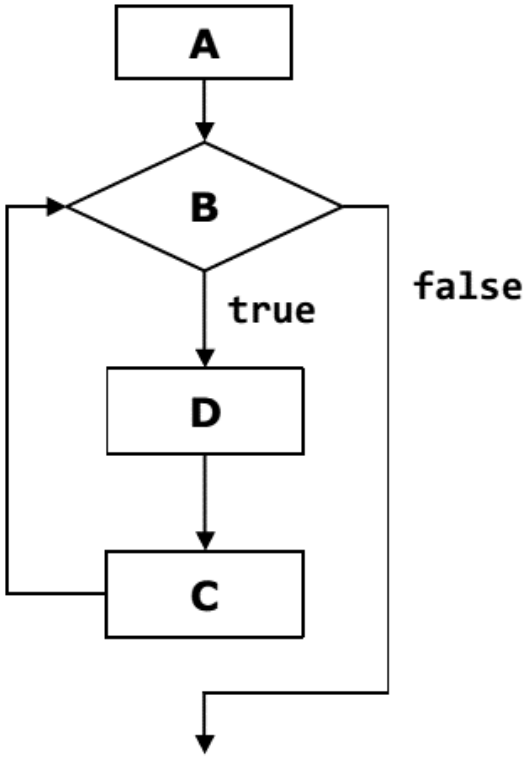
11. Do - while sikli qaysi blok sxemadagidek ish bajaradi.

a)

b)



c)



d) To‘g‘ri javob berilmagan

12.Katta sonlar bilan ishlashda qaysi tipdan foydalaniladi

a) BigInteger
b) Integer
c) Numerics
d) BigDecimal

13.BigInteger dan foydalanish uchun qaysi kutubxonani qo‘shishimiz kerak

a) Reference.dll
b) System.Numeric.dll
c) System.IO.dll
d) System.Numerics.dll

14.**Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

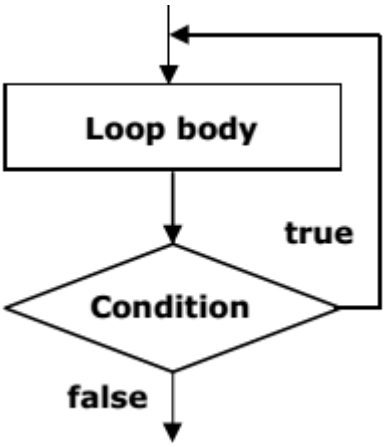
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

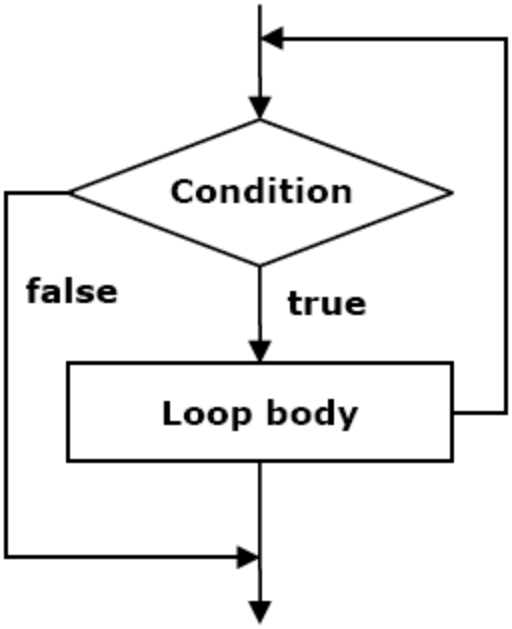
```
namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {


         int n =0;

         int m =5;
         int num = n;
         long product = 1;
         do
         {
            product *= num;
            num++;
         } while (num <= m);
         Console.WriteLine(product);
      }
   }
}
```

a) 1
b) 0
c) 120
d) 125

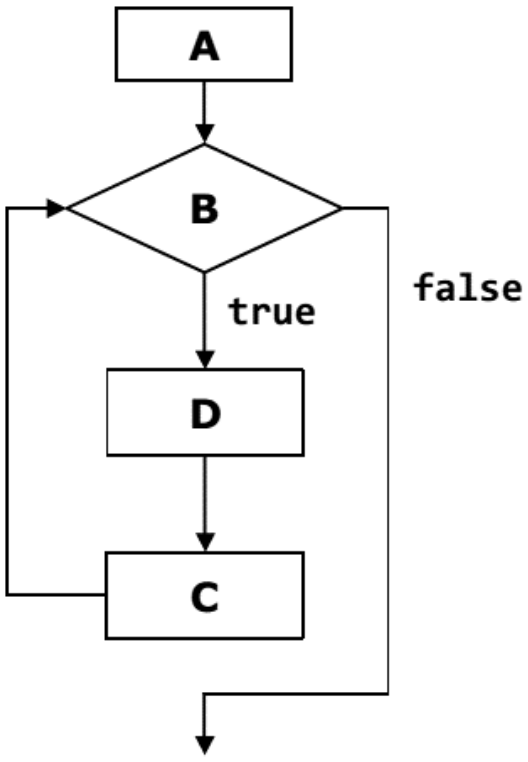
For sikli qaysi blok sxemadagidek ish bajaradi

a)
b)





c)



d) To‘g‘ri javob

berilmagan

15.Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {

         for (int i = 0; i <= 10; i++)
         {
            Console.Write(i + " ");
         }
      }
   }
}
```

a) 0 1 2 3 4 5 6 7 8 9 10

b) 0 1 2 3 4 5 6 7 8 9

s) Kompilyatsiya xatoligi vujudga keladi

d) 1 2 3 4 5 6 7 8 9 10

16. Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
         for (int i = 1, sum = 1; i <= 128; i = i * 2, sum += i)
         {
            Console.WriteLine("i={0}, sum={1}", i, sum);
         }
      }
   }
}
```

a) i=1, sum=1

i=2, sum=3

i=4, sum=7

i=8, sum=15

i=16, sum=31

i=32, sum=63

i=64, sum=127

i=128, sum=255

b) i=2, sum=3

i=4, sum=7

i=8, sum=15

i=16, sum=31

i=32, sum=63

i=64, sum=127

i=128, sum=255

s) i=1, sum=1

i=2, sum=3

i=4, sum=7

i=8, sum=15

i=16, sum=31

i=32, sum=63

i=64, sum=127

d) Kompilyatsiya xatoligi vujudga keladi

17.**Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
         int n = 5;
         int m = 2;
         decimal result = 1;
         for (int i = 0; i < m; i++)
         {
```

```
            result *= n;
        }
        Console.WriteLine("{0}^{1} = {2}", n, m, result);

    }
  }
}
```

a) n^m =25

b) 2^5 =32

s) Kompilyatsiya xatoligi vujudga keladi

d) 5^2 =25

18.**Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
         for (int small = 1, large = 5; small < large; small++, large--)
         {
            Console.WriteLine(small + " " + large);
         }
         Console.ReadKey();
      }
   }
```

```
}
```

a) 1 10

2 9

b) 3 8

4 7

s) Kompilyatsiya xatoligi vujudga keladi

d) 1 10 2 9

19.continue operatori vazifasi to‘liq berilgan javobni toping?

a) takrorlash operatori tanasini bajarishni to‘xtatadi, lekin programmani qurilmadan chiqib ketmasdan takrorlashning keyingi qadamiga “sakrab” o‘tishini tayinlaydi
b) takrorlash operatori tanasini bajarishni to‘xtatadi
c) break operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi
d) takrorlashning keyingi qadamiga “sakrab” o‘tishini tayinlaydi

20. **Quyidagi kodning kompilyatsiya va ijro natijasi qanday bo‘ladi:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
         int n = 10;
         int sum = 0;
         for (int i = 1; i <= n; i += 2)
```

```
            {
               if (i % 7 == 0)
               {
                  continue;
               }
               sum += i;
            }
            Console.WriteLine("sum = " + sum);
            Console.ReadKey();
         }
      }
   }
```

a) 0

b) sum = 55

s) Kompilyatsiya xatoligi vujudga keladi

d) sum =18

7-mavzu. Massivlar

Reja

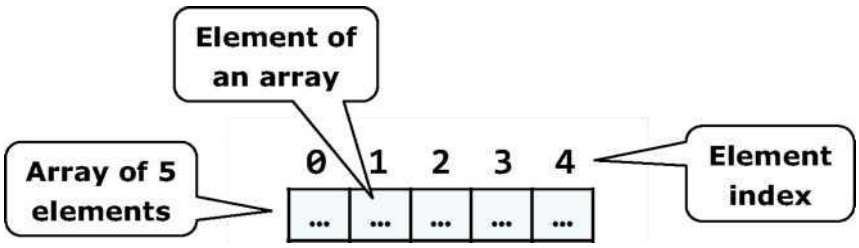
1. Kirish ([2] 235-betlar)
2. Massivlarni e’lon qilishi uchun xotiradan joy ajratish ([2] 236- betdan)
3. Massivning elementlarini joylashtirish ([2] 239-betdan)
4. Konsoldan massivni o‘qish ([2] 242-betdan)
5. Ko‘p o‘lchamli massivlar ([2] 247-betdan)
6. Massivning massivi ([2] 253-betdan)

Kirish

Biz bu bo‘limda bir xil tipdagi elementlar ketma-ketligi bilan ishlashni massivlar orqali o‘rganamiz. Biz massivlar nima ekanligini, ularni qanday qilib e’lon qilinishini, son kiritilishini va foydalanishni tushuntirib o‘tamiz. Biz bir - o‘lchovli va ko‘p o‘lchovli massivlarni ko‘rib chikamiz. Biz massiv orkali konsoldan kiritish va standart o‘qib olish jarayoni ketma-ketligi usullarini o‘rganamiz. Biz massivlar orqali ishlanadigan ko‘plab misollar orqali ularning qanchalik foydaliligini ko‘rsatamiz.

1.1 Massiv nima

Massivlar dasturlash tillari uchun eng muhim hisoblanadi . Ular biz elementlar deb ataydigan o‘zgaruvchilarning massividan iborat.



Massivlarning bir qator elementlari S# da 0, 1, 2, ... N- 1 bilan nomerlanadi. Ana shu raqamlar indekslar deb ataladi. Berilgan massivdagi qator elementlarning umumiy soni esa bu massiv uzunligi deyiladi. Massiv elementlari bir xil tipda bo‘ladi. Bu bizga bir guruh o‘xshash elementlarning belgilangan ketma – ketlikda namoyon bo‘lishi va umuman ular ustida ishlash imkonini beradi.

Massivlar o‘lchami turli bo‘lishi mumkin, lekin eng ko‘p foydalaniladigan massivlar bir o‘lchovli va ikki o‘lchovlilardir. Bir – o‘lchovli massivlar vektor va ikki o‘lchovli massivlar esa matritsalar deyiladi.

Massivlarni e’loni uchun xotiradan joy ajratish

Ushbu S# tilida massivlar uzunligi massiv e’loni paytida aniq bo‘ladi va aynan shu paytda elementlar soni ham aniq bo‘ladi. Bir marta massivning uzunligi kiritildimi, uni boshqa o‘zgartirib bo‘lmaydi.

2 Massivlarni e’lon qilishi uchun xotiradan joy ajratish

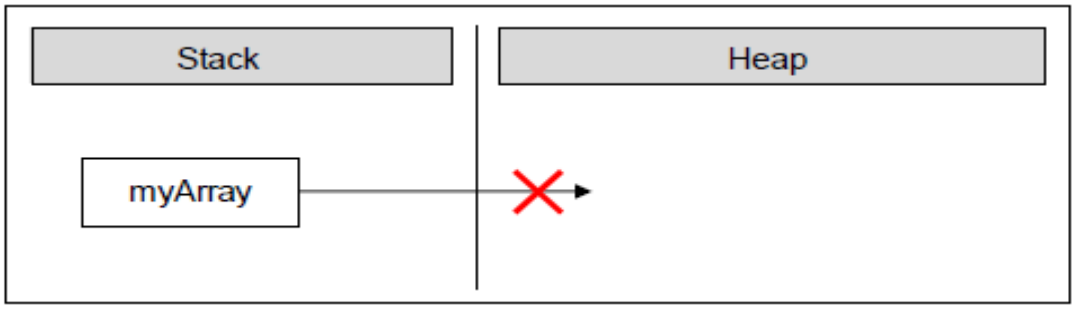
Biz massivni quyidagicha e’lon qilamiz:

```
int[] myArray;
```

Ushbu misolda myArray o‘zgaruvchisi massivning nomi hisoblanadi, qaysiki butun sonlar tipiga(int) tegishli. Bu shuni anglatadiki biz massiv elementlarini butun sonlar tipidan olamiz. Ushbu belgi [] orqali biz bu o‘zgaruvchi massiv elemetlari yagona emasligini ko‘rsatamiz.

Biz massiv o‘zgaruvchisini tipini e’lon qilganimizda u qiymatga ega bo‘lmaydi (bo‘sh). Chunki, unda hali elementlar uchun xotira ajratilmagan bo‘ladi.

Quyidagi rasmda e’lon qilinmagan massiv elementlari uchun xotira ajratilmagan paytida qanaqa ko‘rinishda bo‘lishini ko‘rsatadi.



Dasturning amalga oshirish jarayonida myArray o‘zgaruvchisi kiritiladi va unga qiymati nolga teng(qiymatga ega emas) bo‘ladi.

2.1 Massiv yaratish - "new" opreatori

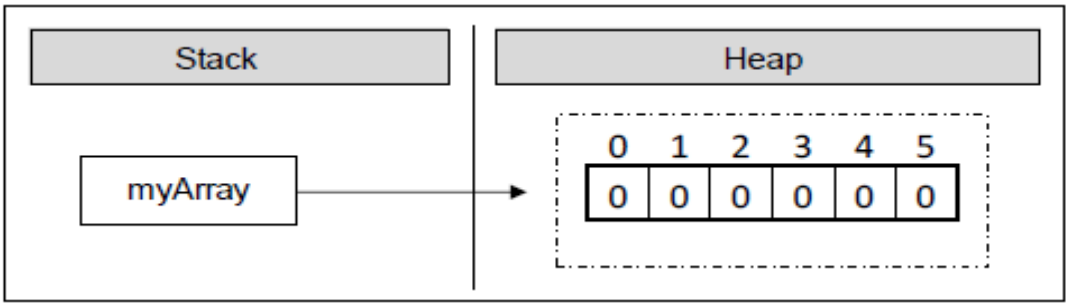
Biz C# da "new" kalit so‘zi orqali massivni e’lon qilamiz. Bu xotiradan joy olishda foydalaniladi

.

int [] myArray = new int [6] ;

```
int[] myArray = new int[6];
```

Bu misolda biz elementlari int tipiga tegishli uzunligi 6 ga teng massiv yaratdik. Bunda 6 ta butun tipli son uchun dinamik xotira(heap) joy ajratiladi va ular 0 qiymati bilan initsializatsiya qilinadi:



Bu rasm shuni ko‘rsatadiki, massiv elemetlari uchun xotiradan olingan joy qattiq xotirada bo‘ladi. C# da massivning elementlari doimo qattiq (dinamik) xotirada saqlanadi. Massiv uchun xotiradan joy ajratish jarayonida qavs ichida elementlar sonini kiritamiz*(manfiy bo‘lmagan butun son), bu uning uzunligini bildiradi. Elementning tipi **new** kalit so‘zidan keyin yoziladi, biz bu orqali xotiradan qanday tipdagi elementlar uchun joy ajratilishini ko‘rsatamiz.

2.2 Massiv ishga tushishi va mavjud qiymatlar

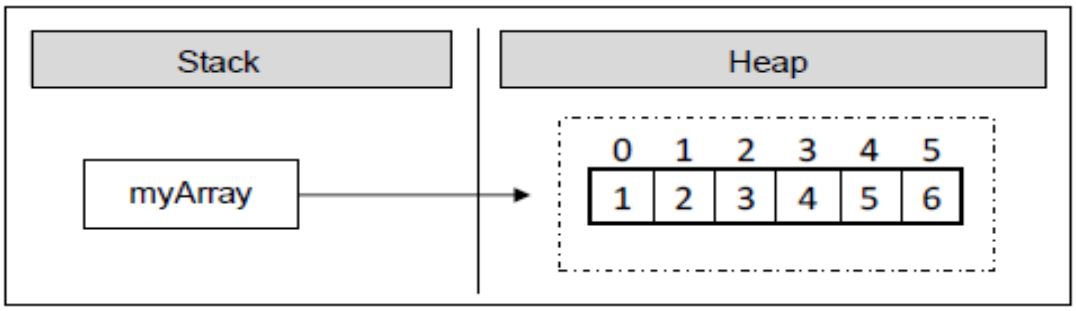
Berilgan massiv elementlarini qo‘llashdan oldin, uning qiymatlarini kiritishimiz kerak. Bazi dasturlash tillarida massiv elementlarining boshlang‘ich qiymati

bo‘lmaganligi sababli , biz undan foydalanmoqchi bo‘lsak ,xatolik beradi. S# da massiv elementlarining boshlang‘ich qiymati bor,xoh uning qiymati 0 yoki bo‘sh bo‘lsin(ular ham tiplarga tegishli). Albatta boshlang‘ich qiymatlarni ochiqchasiga kiritishimiz mumkun. Buni turli yo‘llar orqali amalga oshirish mumkun.

Mana ulardan biri:

```
int[] myArray = { 1, 2, 3, 4, 5, 6 };
```

Ushbu holatda biz massiv va uning boshlang‘ich qiymatlarini e’lon qilayotgan paytimizda kiritamiz. Quyidagi shakl orqali biz massivning qanday qilib uning elementlari qiymatlari to‘g‘ridan-to‘g‘ri e’lon qilinayotgan paytda xotiradan joy olishini ko‘rishimiz mumkin.



Ushbu sintaksisda biz **new** operator o‘rniga figurali qavslardan foydalandik. Qavslar orasiga vergullar bilan ajratgan g‘olda massiv elementlarining boshlang‘ich qiymatini kiritdik. Elementlar soni massiv uzunligini bildiradi.

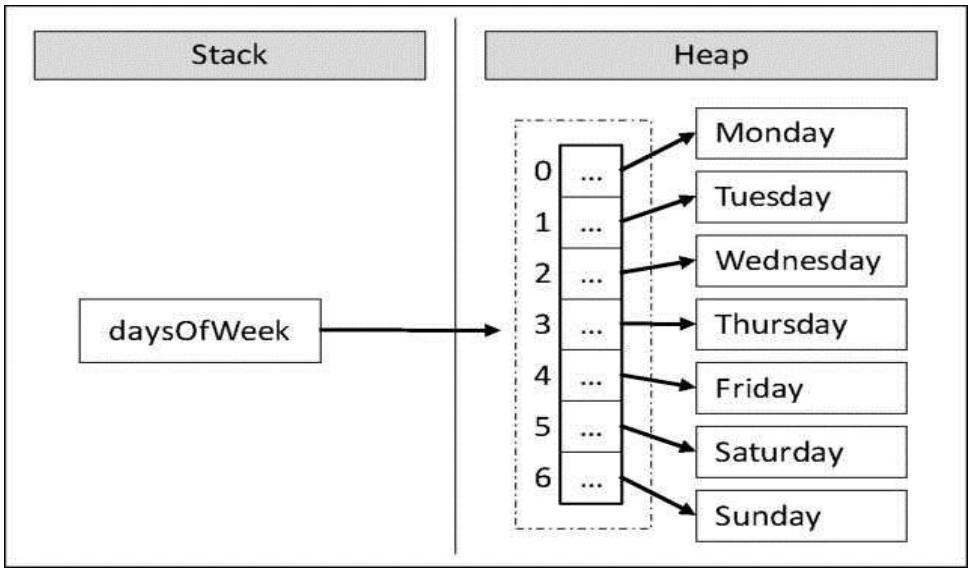
2.3 Massivni e’lon qilish va initsializatsiya qilishga –misol

Bu yerdagi misol massivni e’lon qilish va initsializatsiya qilishga oid:

```
string[] daysOfWeek =
  { "Monday", "Tuesday", "Wednesday","Thursday", "Friday",
  "Saturday", "Sunday" };
```

Bu misolda biz massivni satr tipiga tegishli 7 ta elementni xotiraga joylashtirdik. Satr tipi refrens tipiga tegishli bo‘lib , ularning qiymati dinamik xotirada saqlanadi. **daysOfWeek** o‘zgaruvchi stak xotirasida saqlanadi va uning elementlari esa dinamik xotirada saqlanadi. Massivning har bir elementi satr tipiga tegishli bo‘ladi va har bir massiv elementi uchun dinamik xotiradan alohida joy ajratiladi.

Ushbu rasmda massivning elementlari qanday qilib xotiradan joy olganini ko‘rishingiz mumkin:



3 Massivning chegarasi

Massiv elementlari 0 dan boshlab indekslanadi. Birinchi element 0 bilan, 2-element 1 bilan va shu tartibda n-element n-1bilan indekslanadi.

3.1 Massiv elementlarini kiritish

Biz massiv elementlarini ularni indekslaridan foydalangan holda kiritamiz. Har bir elementning indeksi massivning nomidan keyin va to‘rtburchak qavs ichiga joylashadi. Berilgan massivning elementlariga o‘qish va yozish uchun kira olasiz.

Massiv elementlarini kiritish doir misol:

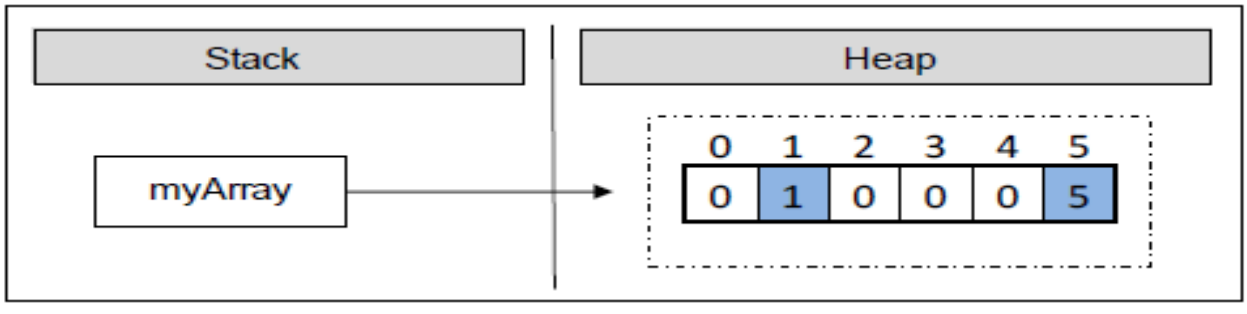
```
myArray[index] = 100;
```

Yuqoridagi misolda, massivning indeks o‘rinda turgan elementiga 100 qiymatini beramiz.

Quyidagi misolda, biz massivni nomerladik va bazi elementlariga qiymat kiritdik.

```
int[] myArray = new int[6];
myArray[1] = 1;
myArray[5] = 5;
```

Yuqoridagi o‘zgarishdan keyin massiv quyidagi ko‘rinishda xotiradan joy oladi.



Rasmdan ko‘rish mumkinki xotirada biz qiymat bergan 2 ta elementdan tashqari barcha elementlar 0 bilan to‘ldirilgan. Biz massiv elementlarini ketma – ketma sikl operatoridan foydalangan holda kiritishimiz mumkin. Massiv elementlarini kiritishni eng ko‘p tarqalgan usuli **for** sikl operatoridan foydalanishdir.

```
int[] arr = new int[5];
for (int i = 0; i < arr.Length; i++)
{
   arr[i] = i;
}
```

3.2 Massivni chegaralash

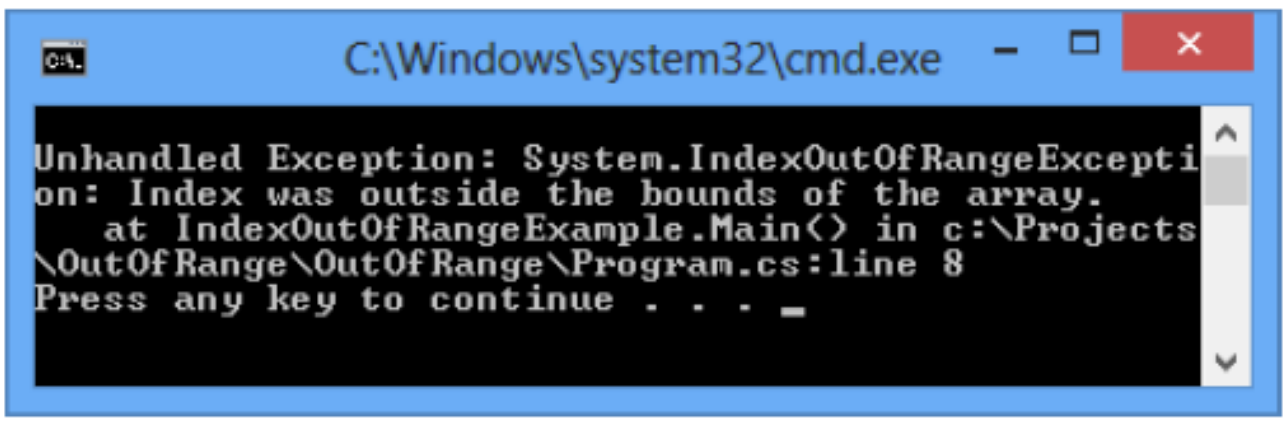
.NET Framework kiritilgan indeks massivga tegishli yoki tegishli emasligini **automatic check** (avtomatik tekshirish) orqali tekshirib beradi. Biz agar massivda mavjud bo‘lmagan element kiritsak, **System.IndexOutOfRangeException** yozuv chiqadi. Bu tekshirish foydalanuvchilarga massiv bilan ishlashda qayerda xatolik bo‘layotganligi ko‘rsatib turadi. Lekin bu pullik. Bu tekshirishda qayerda xatolik bo‘lganini aniqlash mumkin.

Quyidagi misolda biz massivda mavjud bo‘lmagan elementni kiritamiz.

IndexOutOfRangeExample.cs

```
class IndexOutOfRangeExample
{
   static void Main()
   {
      int[] myArray = { 1, 2, 3, 4, 5, 6 };
      Console.WriteLine(myArray[6]);
   }
}
```

Yuqorida misolda 6 ta butun sonlardan iborat bo‘lgan massiv uchun xotiradan joy ajratdik. Birinchi indeksi 0 va oxirgi indeksi 5 ga teng. Biz konsolga massivning



indeksi 6 ga teng bo‘lgan elementni chop etishga harakat qilyapmiz lekin bunday element mavjud bo‘lmaganligi uchun quyidagi istisno holatni olib keladi.

3.3 Massivni teskari tartiblash

Keyingi misolda massiv elementlarini kiritamiz va ularning indekslaridan foydalangan holda o‘zgartiramiz. Ushbu misolda massiv elementlarini teskari tartibda chop etish talab qilingan. Biz massiv elementlarini 2-yordamchi massivdan foydalangan holda teskari tartibda chop etamiz. Bunda 1-massiv elementlari teskari tartibda 2-massivning elementlari ham bo‘ladi. Har ikki massivlarning ham uzunligi 1

ArrayReverseExample.cs

```
class ArrayReverseExample
{
  static void Main()
  {
    int[] array = { 1, 2, 3, 4, 5 };
    // Get array size
    int length = array.Length;
    // Declare and create the reversed array
    int[] reversed = new int[length];

    // Initialize the reversed array
    for (int index = 0; index < length; index++)
    {
      reversed[length - index - 1] = array[index];
    }

    // Print the reversed array
    for (int index = 0; index < length; index++)
    {
      Console.Write(reversed[index] + " ");
    }
  }
}
// Output: 5 4 3 2 1
```

xil bo‘ladi.

Bu misol quyidagicha ishlaydi. Dastlab biz elementlari int tipiga tegishli bir o‘lchami 5 ga teng bo‘lgan massiv e’lon qilamiz. elementlarga 1 dan 5 gacha bo‘lgan qiymatlarni ketma-ket beramiz. Undan so‘ng massiv uzunligini **length**

o‘zgaruvchisiga saqlaymiz. Ushbu qiymatni xosil qilishda **Length** xossasidan foydalanamiz va bunda u massiv elementlarining umumiy sonini qaytaradi. C# da har bir massiv uzunligini aniqlashda **Length** xossasidan foydalaniladi.

Bundan so‘ng, dastlabki massivni uzunligi bilan bir hil bo‘lgan **reversed** nomli massiv e’lon qilamiz. **reversed** nomli massiv elementlari dastlabki massiv elementlaridek bo‘lib, faqat teskari tartibda bo‘ladi.

Elementlarni testari tartiblash uchun **for**-sikl operatoridan foydalanamiz. Har bir jarayon ketma – ketligida biz indeks o‘zgaruvchisini birma – bir oshirib boramiz va bu orqali biz dastlabki massiv elementining hech bir elementi tushib qolmayotganiga ishonch hosil qilamiz. Sikl operatori ishlash ketma – ketligi soni massiv uzunligiga teng.

Biz bu ketma – ketlik jarayonida massiv ustida nima sodir bo‘lganini kuzatamiz?

Jarayon boshida indeksni qiymati 0 ga teng bo‘ldi. Massivni **array[index]** dan foydalangan holda massivning birinchi elementini kiramiz va bu elementni yangi **reversed** massivining **[length - index - 1]** elementi bo‘lib, yangi massiv **reversed** ning oxirgi elementi bo‘lib xotiraga joylashadi. Shunday qilib biz dastlabki massivning birinchi elementini **reversed** massivining oxirgi elementiga mos qo‘yamiz. Har bir jarayon ketma – ketligida indeksni qiymati bittaga oshadi. Bu yo‘l orqali biz keyingi element **array** massivining to‘g‘ri tartibida **reversed** massivining teskari tartibida joylashadi.

Natijada, biz massivni teskar tartibda joylashtirdik va shu tartibda chop etamiz. Bu misolda biz **for** sikli orqali amalga oshirdik, lekin bu jarayonni boshqa sikl operatorlari (while va **foreach** sikl operatorlari) dan foydalangan holda bajarish mumkin.

4 Konsoldan massivni o‘qish

Konsoldan massiv qiymatlarini qanday o‘qishimiz mumkinligi ko‘rib chiqamiz. Biz konsolidan o‘qish uchun NET Framework vositalari va sikl operatorlardan foydalanamiz.

Dastlab biz **Console.ReadLine( )** yordamida konsoldan qatorni o‘qib, keyin biz **int.Parse** yordamida qatorni butun soni tipiga o‘tkazamiz va uni **n** o‘zgaruvchisiga o‘zlashtiramiz. Biz bu **n** o‘zgaruvchisidan massivni uzunligi sifatida foydalanamiz.

```
int n = int.Parse(Console.ReadLine());
int[] array = new int[n];
```

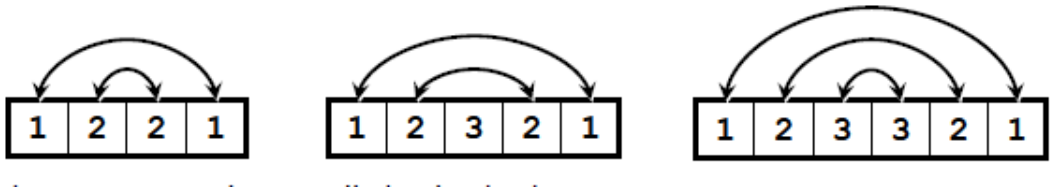
Biz yana massiv ustida takrorlash uchun sikl operatorlaridan foydalanamiz. Har bir takrorlash davomida biz konsoldan o‘qib olingan joriy elementni massivga yuklaymiz. Massivning barcha elementlarini o‘qib olish uchun sikl **n** marta takrorlanadi .

```
for (int i = 0; i < n; i++)
{
   array[i] = int.Parse(Console.ReadLine());
}
```

4.1 Simmetrik massivni tekshirish

Agar massivning birinchi va oxirgi elementi, ikinchi elementi unga mos ravishda massivning oxirgi elementdan bitta oldingi elementi va shu tartibda tenglik mos ravishda davom etsa, bu simmetrik massiv deyiladi. Quyidagi shakllarda simmetrik massivlarga misollar keltirilgan.

1



Keyingi misolda massiv simmetrik yoki simmetrik emasligini tekshiramiz :

```
Console.Write("Enter a positive integer: ");
int n = int.Parse(Console.ReadLine());
int[] array = new int[n];

Console.WriteLine("Enter the values of the array:");

for (int i = 0; i < n; i++)
{
   array[i] = int.Parse(Console.ReadLine());
}

bool symmetric = true;
for (int i = 0; i < array.Length / 2; i++)
{
   if (array[i] != array[n - i - 1])
   {
      symmetric = false;
      break;
   }
}

Console.WriteLine("Is symmetric? {0}", symmetric);
```

Biz massivni initsializatsiya qilamiz va uning elementlarini konsoldan o‘qiymiz. Simmetrik ekanligini tekshirish uchun bizga massivning yarmi orqali takrorlashni amalga oshirishimiz kerak bo‘ladi. Massivning o‘rtadagi elementi **array.Length / 2** indeksidir. Agar uzunlik toq son bo‘lsa, bu indeks o‘rta qiymatidan bitta kam bo‘ladi, lekin, agar u juf son bo‘lsa, indeks o‘rtasi (ikki elementning o‘rtasi)dan boshlanadi. Shunday qilib, sikl 0 dan **array.Length / 2** gacha takrorlashni bajaradi.

Massivning simmetrik ekanligini tekshirish uchun, biz **bool** o‘zgaruvchisidan foydalanamiz va dastlab biz massivni simmetrik deb tasavur qilamiz. Takrorlashlar davomida biz birinchi elementni oxirgi element bilan, ikkinchi elementni oxiridan bitta oldingisi bilan taqqoslaymiz. Agar qaysidir paytda elementlar teng bo‘lmasa, keyin **bool** o‘zgaruvchisi **false** qiymatini oladi va massiv simmetrik emasligi aniq bo‘ladi.

Va nixoyat konsolga **bool** o‘zgaruvchisini qiymati chop ettiriladi.

4.2 Konsolga massivni chop etish

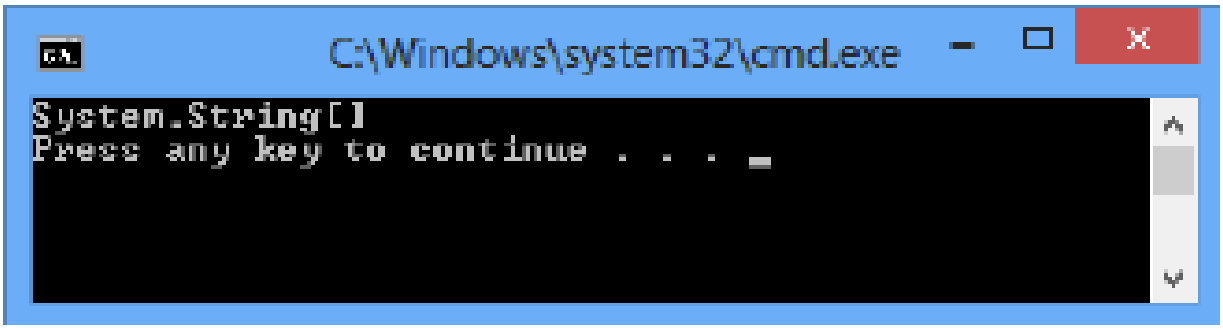
Ko‘p xollarda massiv bilan ishlashni tugatganimizdan keyin massiv elementlarini konsolga chop etishimiz kerak bo‘ladi.

Biz initsializatsiya qilingan (kiritilgan) massiv elmentlarini konsolga chop etamiz, masalan sikl operatorlaridan foydalangan holda. Massivlarni chop etishda aniq bir qoida yo‘q amma bir qancha tanlab olingan usullardan foydalaniladi.

Ko‘pincha tez – tez uchrab turadigan xatolardan biri son sifatida chop etish hisoblanadi.

```
string[] array = { "one", "two", "three", "four" };
Console.WriteLine(array);
```

Afsuski bu kod massiv elementlarini chiqarmaydi faqatgina uni tipini chiqarib beradi. Kod natijasini ko‘rishimiz mumkin.



Biz **for** sikli yordamida birma – bir massiv elementlarini chop etish.

```
string[] array = { "one", "two", "three", "four" };

for (int index = 0; index < array.Length; index++)
{
   // Print each element on a separate line
   Console.WriteLine("Element[{0}] = {1}", index, array[index]);
}
```

Biz **for**-siklidan foydalangan holda massiv elementlari ko‘rib chiqiladi va bu jarayon **array.Length** davom etadi va satr formatida **Console.WriteLine()** dan foydalanib joriy elementlarini chop etamiz. Natijasi quyidagicha:

```
Element[0] = one
Element[1] = two
Element[2] = three
Element[3] = four
```

4.3 Massiv elementlarini ketma – ketligi

Ko‘rib turganimizdek, massiv bilan ishlashda eng ko‘p ishlatiladigan texnikalardan biri bu massiv elementlaridan foydalanishdir. Sikl operatori orqali jarayonni ketma – ket amalga oshirish bizga massiv bilan ishlashda ya’ni ularni

indeks orqali kiritishda qulaylik yaratib beradi va ularni hoxlaganimizdek boshqara olamiz. Biz buni turli sikl operatorlari yordamida amalga oshirishimiz mumkin lekin bizga mos keladigani **for** sikl operatoridi. Biz bu ketma – ketlik qanday ishlashini birma – bir ko‘rib chiqamiz.

4.4 "for" sikli bilan jarayon ketma - ketligi

Massiv strukturasi va indekslari bilan ishlaganimizda **for** siklini qo‘llash yaxshi natija beradi. Biz quyidagi misolda barcha massiv elementlarini ikkilantirib chop etamiz.

```
int[] array = new int[] { 1, 2, 3, 4, 5 };

Console.Write("Output: ");
for (int index = 0; index < array.Length; index++)
{
   // Doubling the number
   array[index] = 2 * array[index];
   // Print the number
   Console.Write(array[index] + " ");
}
// Output: 2 4 6 8 10
```

For yordamida massiv indeksining joriy qiymati saqlanadi va indeksini qiymatini ham kiritib boriladi. Bu ketma – ketlikni bajarishini for siklini o‘zi amalga oshiradi. Quyidagi misolda biz massiv elementlarini bazilarini ketma – ketlikda olib boramiz lekin barchasini emas.

```
int[] array = new int[] { 1, 2, 3, 4, 5 };

Console.Write("Output: ");
```

```
for (int index = 0; index < array.Length; index += 2)
{
   array[index] = array[index] * array[index];
   Console.Write(array[index] + " ");
}
// Output: 1 9 25
```

Bu misolda massivni juf o‘rindagi barcha elementlarini kvadratga oshirib konsolga chop etadi.

Biz bazan massivni teskari tartibda olishni hoxlaymiz. **For** sikli massivning oxirgi indeksdan ishini boshlaydi va har bir qadamda indeksning qiymati 0 ga teng

```
int[] array = new int[] { 1, 2, 3, 4, 5 };

Console.Write("Reversed: ");
for (int index = array.Length - 1; index >= 0; index--)
{
   Console.Write(array[index] + " ");
}
// Reversed: 5 4 3 2 1
```

bo‘lguncha bittaga kamayadi. U quyidagicha:

Bu misolda massivni teskari tartibda joylashtirish olib borilmoqda va har bir qiymat konsolga chop etilmoqda.

4.5 "foreach" sikli bilan takrorlash

Massiv elementlari orqali takrorlanishni eng ko‘p qo‘llaniluvchi konstruksiyalaridan biri bu **foreach** operatoridan foydalanishdir. **Foreach** siklini C# dagi konstruksiyasiyasi quyidagicha:

```
foreach (var item in collection)
{
   // Process the value here
}
```

Bu dasturlash konstruksiyasida **var** takrorlanishlar ketma – ketligidagi elementlar tipidir. **collection** massiv (elementlar to‘plami) va **item** o‘zgaruvchisi esa har qadamdagi massivning joriy elementlari.

Umuman olganda **foreach** sikl konstruksiyasi **for** sikl bilan bir xil bo‘lgan qulayliklar mavjud. Asosiy farq shundaki jarayon ketma – ketligi barcha massiv elementlarining boshdan oxirigacha bajariladi. Biz joriy indeksni kirita olmaymiz o‘zini – o‘zi yeg‘ish usuli orqali jarayon ketma – ket olib boriladi. Massivlar uchun bu jarayon ketma – ketligi birinchi elementdan oxirgi elementga tartib bo‘yicha olib boriladi. **Foreach** siklidagi o‘zgaruvchilar faqat o‘qib olinadi shuning uchun siklning tana qismidagi joriy sikl o‘zgaruvchilarini o‘zgartirib bo‘lmaydi.

foreach-siklidan elementlarini o‘zgartirmasdan faqat o‘qib olish uchun ishlatilinsa foydalaniladi. Foreach bilan ketma – ketlik

Quyidagi misolda massiv ketma – ketligidan foydalanish **foreach** sikli orqali

```
string[] capitals =
   { "Sofia", "Washington", "London", "Paris" };

foreach (string capital in capitals)
{
   Console.WriteLine(capital);
}
```

ko‘rsatilgan

Bu misolda satr tipidagi **capitals** nomli massiv e’lon qilinganidan keyin **foreach** sikl operatori yordamida massiv elementlarini konsolga chop etiladi. Har bir element har bir qadamda **capital** o‘zgaruvchisiga joylashadi. Bu kod bajarilgandan so‘ng quyidagi natijaga erishiladi.

```
Sofia
Washington
London
Paris
```

6 Ko‘p o‘lchamli massivlar

Bir o‘lchamli massivlar matematikada vector sifatida ma’lum. Ko‘pincha ko‘p o‘lchamli massivlarga extiyoj seziladi. Masalan, shaxmat taxtasini o‘lchami 8 ga 8 bo‘lgan ikki o‘lchamli massiv sifatida qarash mumkin (8 ta katak gorizontal bo‘yicha va 8 ta katak vertikal bo‘yicha)

6.1 Ko‘p o‘lchamli massiv va matritsa

C# dagi xar bir mavjud tip massiv tipi sifatida ishlatilishi mumkin. Bir o‘lchamli elementlari butun son bo‘lgan massiv int[], ikki o‘lchamli massiv esa int[,] ko‘rinishda e’lon qilinadi. Quyidagi misolda ikki o‘lchamli massivni e’lon qilish keltirilgan:

```
int[,] twoDimensionalArray;
```

Ushbu massivni ikki o‘lchamli deb ataladi, chunki uning o‘lchami ikkiga teng. Ular matematik terminga ko‘ra matritsalar deb ataladi. Umuman olganda, bir o‘lchamdan ko‘p bo‘lgan massivlarni ko‘p o‘lchamli massiv deb ataladi.

Quyida uch o‘lchamli massivni e’lon qilish ko‘rsatilgan:

```
int[,,] threeDimensionalArray;
```

Matematikadagi teoremaga asosan massiv o‘lchami uchun chegara yo‘q ammo amaliyotda ikki o‘lchamli massivdan yuqori tartibi ishlatilmaydi.

6.2 Ko‘p o‘lchamli massivni e’lon qilish va xotiraga joylashtirish

Ko‘p o‘lchamli massivlar bir o‘lchamli massivlarga o‘xshash yo‘l orqali e’lon qilinadi. Bir o‘lchamli massivlardan tashqari qolgan massivlarda massiv o‘lchamidan

```
int[,] intMatrix;
float[,] floatMatrix;
string[,,] strCube;
```

bitta kam vergul bilan ajratiladi.

Yuqoridagi misolda, ikki o‘lchamli va uch o‘lchamli massivlarni yaratish keltirib o‘tilgan. Har bir o‘lcham vergul bilan to‘rtburchak qavslar orqali belgilanadi [].

Ko‘p o‘lchamli massivlarni xotiraga joylashtirishda **new** kalit so‘zidan foydalaniladi va har bir o‘lchami uchun uzunligi aniqlanadi. Massivni e’lon qilish quyida misolda keltirilgan:

```
int[,] intMatrix = new int[3, 4];
float[,] floatMatrix = new float[8, 2];
string[,,] stringCube = new string[5, 5, 5];
```

Bu misolda **intMatrix** ikki o‘lchamli massiv bo‘lib, int[] tipidagi 3 element va ularni har biri 4 uzunlikdan iborat. Ikki o‘lchamli massivlar turli usullarda turlicha tushuntiriladi. Tasavvur qilish uchun ikki o‘lchamli massivlarni ustuni va qatori quyidagi rasmda keltirilgan:

	0	1	2	3
0	1	3	6	2
1	8	5	9	1
2	4	7	3	0

Kvadrat massivning satr va ustunlari indeksi 0 dan n-1 gacha sonlar bilan nomerlanadi. Agar ikki o‘lchamli massivni o‘lchami ustun bo‘yicha – *m* va qator bo‘yicha *n* ga teng bo‘lsa, u holda elementlari m*n ta bo‘ladi.

6.3 Ikki-o‘lchamli massivlarni initsializatsiya qilish

Ikki o‘lchamli massivlar bir o‘lchamli massivlar kabi bir xil initsializatsiya qilinadi. Quyidagi misolda massiv elementlarini to‘g‘ridan to‘g‘ri e’lon qilish ro‘yhati

```
int[,] matrix =
{
  {1, 2, 3, 4}, // row 0 values
  {5, 6, 7, 8}, // row 1 values
};
// The matrix size is 2 x 4 (2 rows, 4 cols)
```

keltirilgan:

Yuqoridagi misolda 2 qator va 4 ustundan iborat ikki o‘lchamli massiv initsializatsiya qilingan. Birinchi fugurali qavsda yozilgan sonlar birinchi qator qiymatlarini ikkinchisi esa ikkinchi qator qiymatlarini bildiradi. Har bir qator bir o‘lchamli massivni o‘z ichiga oladi.

6.4 Ko‘p o‘lchamli massiv elementlarini kiritish

Ikki o‘lchamli matritsaning elementlari ikkita indeksdan foydalangan holda kiritiladi. Bunda birinchi indeks qator elementini, ikkinchi indeks ustun elementini bildiradi. Ko‘p o‘lchamli massivlarda har bir o‘lcham uchun turlicha indeks bor.

Ko‘p o‘lchamli massivning har bir o‘lchamining indeksi 0 dan boshlanadi

Quyidagi misolda keltirilgan:

```
int[,] matrix =
{
  {1, 2, 3, 4},
  {5, 6, 7, 8},
};
```

matrix nomli massivda 8 ta elementi mavjud bo‘lib, 2 qator va 4 ta ustundan

```
matrix[0, 0]  matrix[0, 1]  matrix[0, 2]  matrix[0, 3]
matrix[1, 0]  matrix[1, 1]  matrix[1, 2]  matrix[1, 3]
```

iborat. Har bir element quyidagicha kiritiladi.

Bu misolda har bir element indekslar yordamida kiritiladi. Agar qator indekslari uchun **row**, ustun indekslarini **col** deb olinsa, u quyidagicha bo‘ladi.

```
matrix[row, col]
```

Ko‘p o‘lchamli massiv elementlari o‘zlarining indeks nomerlari bilan farqlanadi va u quyidagicha bo‘ladi:

```
nDimensionalArray[index1, …, indexN]
```

6.5 Ko‘p o‘lchamli massivlaring uzunligi

Ko‘p o‘lchamli massivning har bir o‘lchamini uzunligi bo‘lib, u dastur bajarilishi jarayonida aniqlanadi. Ikki o‘lchamli massiv uchun quyidagi misolda ko‘rsatilgan.

```
int[,] matrix =
{
  {1, 2, 3, 4},
  {5, 6, 7, 8},
};
```

Ikki o‘lchamli massiv qatorlari soni **matrix.GetLength(0)** dan, ustunlar soni esa **matrix.GetLength(1)** foydalanib olinadi. Yuqoridagi misolda **matrix.GetLength(0) 2** ni va **matrix.GetLength(1) esa 4** ni qaytaradi. Demak massiv 2 ta qator va 4 ta ustundan iborat.

6.6 Matritsani chop etish

Keyingi misolda qanday qilib ikki o‘lchovli massiv elementlarini konsolga chop etish mumkinligi ko‘rsatib o‘tilgan:

```
// Declare and initialize a matrix of size 2 x 4
int[,] matrix =
{
  {1, 2, 3, 4}, // row 0 values
  {5, 6, 7, 8}, // row 1 value
};

// Print the matrix on the console
for (int row = 0; row < matrix.GetLength(0); row++)
{
  for (int col = 0; col < matrix.GetLength(1); col++)
  {
    Console.Write(matrix[row, col]);
  }
  Console.WriteLine();
}
```

Dastlab massiv e'lon qilindi va unga qiymat berildi. Bu ikki o'lchamli massiv elementlarini ekranga chop etish uchun **for** sikl operatoridan foydalanildi. Birinchi **for** sikli qator bo'yicha ketma – ketlikni ta'minlab bersa, ichma – ich joylashgan ikkinchi **for** sikli esa o'z navbatida har bir qator bo'yicha ustunlar ketma – ketligini

```
1 2 3 4
5 6 7 8
```

ta'minlab beradi. Har bir jarayon ketma –ketligida bu ikki indeksdan foydalangan holda elementlar konsolga chop etiladi (qator va ustun bo'yicha). Bu kod ishga tushishi natijasi quyidagicha:

6.7 Matritsa elementlarini konsoldan o'qish

Bu misolda ikki o'lchamli massivni konsoldan qanday qilib o'qib olish ko'rsatilgan. Dastlab ikki o'lchamli massiv qiymati (uzunligi) o'qib olinadi. Ichma – ich joylashgan sikl operatoridan foydalangan holda har bir element qiymati kiritiladi (va kiritilgan qiymatlar yana ekranga chiqariladi).

```
Console.Write("Enter the number of the rows: ");
int rows = int.Parse(Console.ReadLine());

Console.Write("Enter the number of the columns: ");
int cols = int.Parse(Console.ReadLine());

int[,] matrix = new int[rows, cols];

Console.WriteLine("Enter the cells of the matrix:");

for (int row = 0; row < rows; row++)
{
   for (int col = 0; col < cols; col++)
   {
      Console.Write("matrix[{0},{1}] = ",row, col);
      matrix[row, col] = int.Parse(Console.ReadLine());
   }
}

for (int row = 0; row < matrix.GetLength(0); row++)
{
   for (int col = 0; col < matrix.GetLength(1); col++)
   {
      Console.Write(" " + matrix[row, col]);
   }
   Console.WriteLine();
}
```

Yuqoridagi dastur natijasi quyidagicha bo‘ladi (bu misolda massiv 3 ta qator va 2 ta ustunlardan iborat):

```
Enter the number of the rows: 3
Enter the number of the columns: 2
Enter the cells of the matrix:
matrix[0,0] = 2
matrix[0,1] = 3
matrix[1,0] = 5
matrix[1,1] = 10
matrix[2,0] = 8
matrix[2,1] = 9
 2 3
 5 10
 8 9
```

6.8 Matritsani maksimal qiymat

Keyingi misolda yana bir boshqa bir qiziqarli misolni ko‘ramiz. Bizga elementlari butun son bo‘lgan ikki o‘lchamli to‘rtburchakli massiv berilgan bo‘lib,bu ko‘p o‘lchamli massiv ichidan shunday ikkiga ikki massiv olishimiz kerakki ushbu massiv elementlari boshqa bir ikki o‘lchamli massiv elementlaridan katta bo‘lsin, va uni hamda elementlari yig‘indisini chop etishimiz kerak.

Bu masal1aning yechimi quyidagicha:

```
            long sum = matrix[row, col] + matrix[row, col + 1] +
              matrix[row + 1, col] + matrix[row + 1, col + 1];
            if (sum > bestSum)
            {
               bestSum = sum;
               bestRow = row;
               bestCol = col;
            }
         }
      }

      // Print the result
      Console.WriteLine("The best platform is:");
      Console.WriteLine("  {0} {1}",
         matrix[bestRow, bestCol],
         matrix[bestRow, bestCol + 1]);
      Console.WriteLine("  {0} {1}",
         matrix[bestRow + 1, bestCol],
         matrix[bestRow + 1, bestCol + 1]);
      Console.WriteLine("The maximal sum is: {0}", bestSum);
   }
}
```

Agar dastur amalga oshirilsa, quyidagi natija olinadi

```
The best platform is:
  9 8
  7 9
The maximal sum is: 33
```

Algoritm quyidagicha tushintiriladi. Dastlab, elementlari butun son bo‘lgan o‘lchamli massiv yaratiladi. bestSum, bestRow, bestCol nomlanuvchi yordamchi o‘zgaruvchilar e’lon qilinadi va bestSum ning qiymati sifatida long tipli eng kichik son beriladi. 4 ta butun sonni yig‘indisi **int** tipga sig‘masligini inobatga olgan holda, **long** tipidan foydalanildi.

bestRow, bestCol o‘zgaruvchilarida elementlari boshqa ikki o‘lchamli massiv elementlaridan katta bo‘lgan massiv elementlari ularni yig‘indisi esa **bestSum** o‘zgaruvchisida saqlanadi.

2 x 2 massiv elementlarini kiritish uchun birinchi element indekslari kerak bo‘ladi. Ularni amalga oshirgach qolgan 3 ta elementlarini ham kiritiladi:

```
matrix[row, col]
matrix[row, col + 1]
```

```
matrix[row + 1, col]
matrix[row + 1, col + 1]
```

Ushbu misolda row va col o‘zgaruvchilari **matrix** nomli massivga tegishli 2 ga 2 matritsani elementlarini indekslaridir.

Bu orqali 2 x 2 matritsaning birinchi elementining indekslarini bilgan holda qolgan elementlarni topish mumkin. Bu algoritm elementlari boshqa 2 x 2 massiv elementlaridan katta bo‘lgan massivni aniqlash uchun foydalaniladi. Bu 2 ga 2 matritsani topish jarayoni matritsani elementlarining yig‘indisi eng katta bo‘lguncha davom etadi. Bu ish 2 ta ichma – ich joylashgan **for** sikl operatori hamda **row** va **col** orqali amalga oshiriladi. Shuni yodda tutish kerakki jarayon butun matritsa bo‘yicha amalga oshirilmaydi chunki agar **row + 1** yoki **col + 1** elementlar kiritilsa, matritsa o‘lchamidan oshib ketishi mumkin va xatolikka olib keladi va **System.IndexOutOfRangeException** istisno holatga olib keladi.

Joriy elementning qo‘shni elementlari bo‘yicha yangi matritsa tuziladi va ularning yig‘indisi hisoblanadi va bu qiymat oldingi qiymatlar bilan taqqoslanadi. Agar bu qiymat katta bo‘lsa, joriy yig‘indi eng katta bo‘ladi va bu elementlar indekslari **bestRow** va **bestCol** ga joylashadi. Bu jarayonni asosiy matritsa bo‘yicha ko‘rib chiqqanimizdan keyin elementlari eng katta bo‘lgan 2 ga 2 matritsa topiladi. Agar asosiy matritsada o‘lchami 2 ga 2 bo‘lgan, elementlar eng katta va yig‘indisi teng bo‘lgan yana bitta matritsa topilsa natija sifatida birinchi massivni olinadi.

Jarayon oxirida topilgan 2 x 2 matritsa elementlari va ularning yig‘indisi konsolga chop etiladi.

7 Massivlarning massivi

C# dasturlash tilida massivlarning massivi bo‘lib, u **jagged** massiv deb ataladi.

Jagged massiv massivning massivi (ichma –ich joylashgan) bo‘lib, massiv qatori elementlari massivlardan iborat bo‘ladi.

7.1 Massivning massivini e’lon qilish va hotiradan joy ajratish

Ko‘p o‘lchamli massiv bilan **jagged** massivni e’lon qilishdagi asosiy bitta farq bir juft qavsning ko‘p bo‘lishidadir. Massivning har bir o‘lchami uchun juft qavslar bor va u quyidagicha joylashadi:

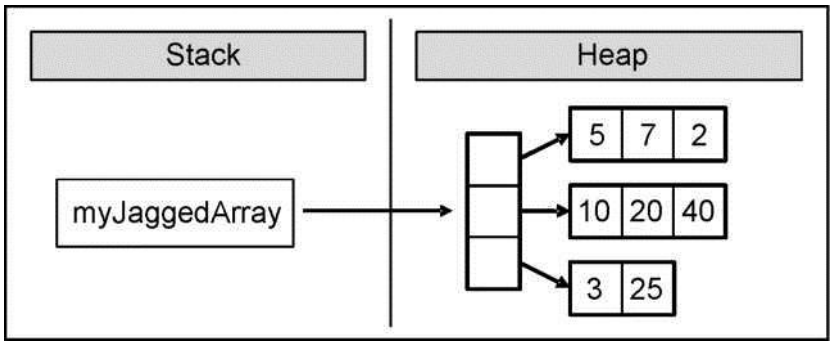
```
int[][] jaggedArray;
jaggedArray = new int[2][];
jaggedArray[0] = new int[5];
jaggedArray[1] = new int[3];
```

Bu yerda jagged massivining qanday qilib e’lon qilish, elementlariga qiymat berish va xotiraga joylashtirish ko‘rsatilagan (jagged massiv elementlari elementlari butun tipli massivdan iborat):

```
int[][] myJaggedArray = {
   new int[] {5, 7, 2},
   new int[] {10, 20, 40},
   new int[] {3, 25}
};
```

7.2 Xotiraga joylashtirish

myJaggedArray nomli **jagged** massivini qanday qilib e’lon qilish xotiraga joylashtirish quyidagi rasmda ko‘rsatilgan. Yuqorida ko‘rganimizdek **jagged** massiv massivlarni to‘plami hisoblanadi. **jagged** massivi hech qanday massivni o‘z ichiga olmaydi lekin shunga qaramasdan ularni elementlarini ko‘rsatib turadi. O‘lchamini bilib bo‘lmaydi shuning uchun ham ular haqidagi massivning mavjud ma’lumotlari CLR da saqlanadi. Keyin xotiradan **jagged** massivining elementlari uchun alohida massiv ajratiladi, keyin ushbu ma’lumotlar dinamik xotiradan ajratilgan yangi blokka joylashadi. **myJaggedArray** statik xotirada saqlanadi va u dinamik xotiraga qaratilgan bo‘ladi. Ushbu dinamik xotirada ketma-ketlikdagi 3 ma’lumot 3 ta blokda saqlanadi.har bir blok elementlari butun son bo‘lgan massivni o‘z ichiga oladi.



7.3 Elementlarni e’lon qilish va initsializatsiya qilish

Biz **jagged** massiv elementlar (o‘z navbatida massiv) ini ularning indekslaridan foydalangan holda kiritamiz. Quyidagi misolda massivning 0 –indeksida joylashgan

```
myJaggedArray[0][2] = 45;
```

```
int[][,] jaggedOfMulti = new int[2][,];
```

```
jaggedOfMulti[0] = new int[,] { { 5, 15 }, { 125, 206 } };
jaggedOfMulti[1] = new int[,] { { 3, 4, 5 }, { 7, 8, 9 } };
```

massivning 3-indeksida turgan elementi quyidagicha e’lon qilingan:

Paskal uchburchagi- misol

Keyingi misolda biz jagged massividan paskal uchburchagini yaratish va formatlash uchun foydalanamiz. Matematikadan ma’lumki uchburchakning birinchi qatori 1 dan boshlanadi va keyingi qatorlar iborat. Matimatikadan bilganimizdek, uchburchakning har bir qatori 1 raqamidan boshlanadi va har bir keyingi raqam esa undan yuqoridagi qator elementlari yig‘indisiga teng.paskal uchburchagi quyidagicha ko‘rinishda bo‘ladi

Berilgan balandlikdagi paskal uchburchagini hosil qilish uchun biz 0-qatori 1

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
      . . .
```

dan iborat jagged massivini e’lon qilamiz.elementlar soni qatorlar bilan mos ravishda o‘sadi.dastlab biz **triangle[0][0] = 1** ni beramiz, qolgan katakchalardagi qiymat 0 ga nisbatan joylashadi. Keyin sikl operatori orqali **row** qiymati birma – bir oshib boradi. Bu ichma - ich joylashgan sikl orqali joriy sikl ustunlar bilan ham ishlaydi va paskal uchburchagining qiymati quyidagicha, ya’ni joriy qator **(triangle[row][col])** katakchaga, pastagi **(triangle[row+1][col])** katakchaga va pastdan o‘ng tarafdagi **(triangle [row+1][col+1])** qiymat kiritiladi. Bo‘sh katakchadagi mos qiymatlarni (satr tipiga tegishli **PadLeft ()** metodidan foydalanib) ekranga chop etamiz.

Quyida algoritmining kodi tasvirlangan:

PascalTriangle.cs

```
class PascalTriangle
{
  static void Main()
  {
    const int HEIGHT = 12;

    // Allocate the array in a triangle form
    long[][] triangle = new long[HEIGHT + 1][];

    for (int row = 0; row < HEIGHT; row++)
    {
```

Agar bu dasturni amalga oshirsak, uni to‘liq darajada ishlayotganiga va Paskal uchburchagini berilgan qator nomeri bo‘yicha yaratadi (bu misolda balandlik 12 ga teng).

```
      triangle[row] = new long[row + 1];
    }

    // Calculate the Pascal's triangle
    triangle[0][0] = 1;
    for (int row = 0; row < HEIGHT - 1; row++)
    {
      for (int col = 0; col <= row; col++)
      {
        triangle[row + 1][col] += triangle[row][col];
        triangle[row + 1][col + 1] += triangle[row][col];
      }
    }

    // Print the Pascal's triangle
    for (int row = 0; row < HEIGHT; row++)
    {
      Console.Write("".PadLeft((HEIGHT - row) * 2));
      for (int col = 0; col <= row; col++)
      {
        Console.Write("{0,3} ", triangle[row][col]);
      }
      Console.WriteLine();
    }
  }
}
```

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1
  1   9  36  84 126 126  84  36   9   1
1  10  45 120 210 252 210 120  45  10   1
1  11  55 165 330 462 462 330 165  55  11   1
```

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

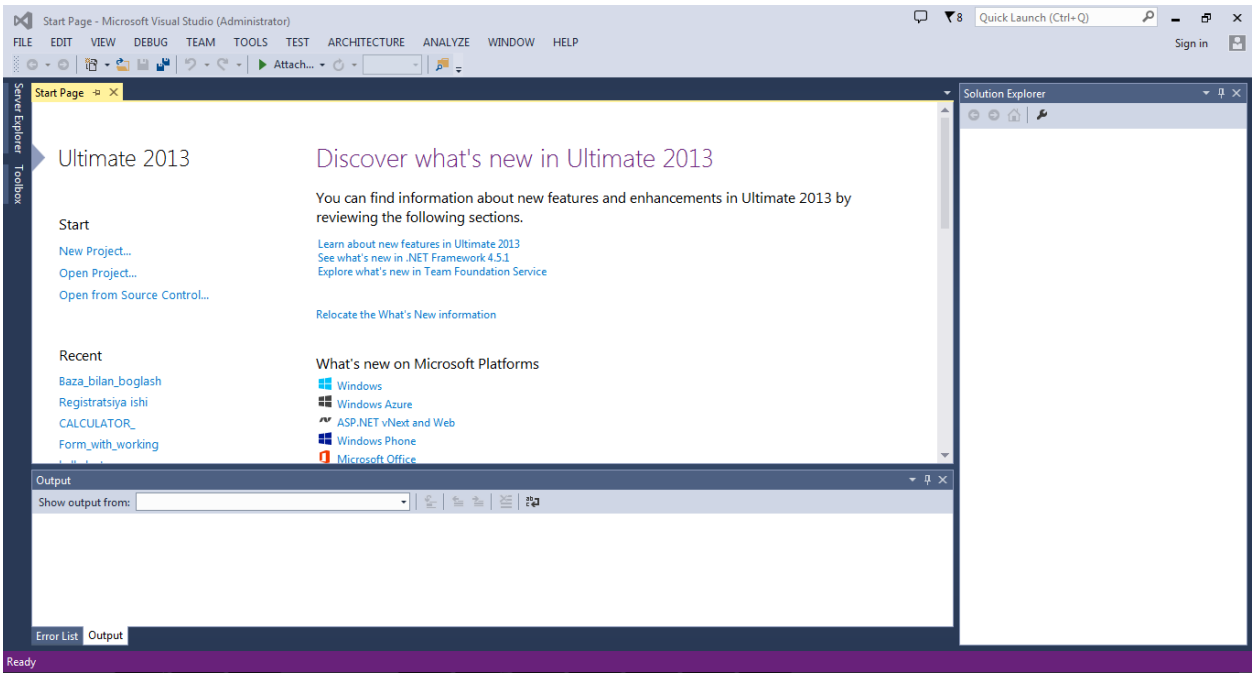
- **Massiv** - bir xil tipdagi elementlar ketma-ketligi
- **Length** – massiv uzunligi (o‘lchami) ni aniqlash funksiyasi
- **New** - massiv yaratishda bu operatordan foydalaniladi
- **Sikl operatori** – takrorlash sharti deb nomlanuvchi mantiqiy ifodani rost (true) qiymatida programmaning ma’lum bir qismlaridagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajarishni amalga oshiradi (itaratsiya)
- **Break –** operatori sikldan chiqib ketishda foydalaniladi
- **BigInteger –** katta sonlar bilan ishlashga mo‘ljallangan klass
- **continue** – bu operatori xuddi break operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin programmani qurilmadan chiqib ketmasdan takrorlashning keyingi qadamiga “sakrab” o‘tishini tayinlaydi

- **foreach –** bu sikl operatori C/C++/C# dasturlash tillari oilasi uchun yangi ammo bu **VP va PHP** dasturchilari uchun yaxshi tanish dasturning konstruksiyasida massivning barcha elementlarini va ro‘yxatdan yoki elementlar yig‘indisida (**IEnumerable**) ishlatiladi
- **Write(…) –** konsolga argumentlarni chiqarish
- **WriteLine(…) –** konsolga ma’lumotlarni chiqarish va keyingi qatorga o‘tish
- **Console.ReadKey() –** klavish bosilguncha ekranni ushlab turadi
- **KeyChar –** kiritilgan belgini ushlab turadi
- **int.Parse(string) –** satrni butun tipga o‘tkazadi
- **Convert klassi –** metodidan foydalangan holda boshqa tipga o‘tkazish uchun ishlatiladi
- **try-catch –** hatoliklarni ushlab qolish uchun ishlatiladi

Amaliy mashg‘ulot

1-misol. Massivlarni konsol orqali kiriting va maksimal elementini topish dasturini tuzing.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi va yangi loyiha yaratamiz.



Endi berilgan masala kodini

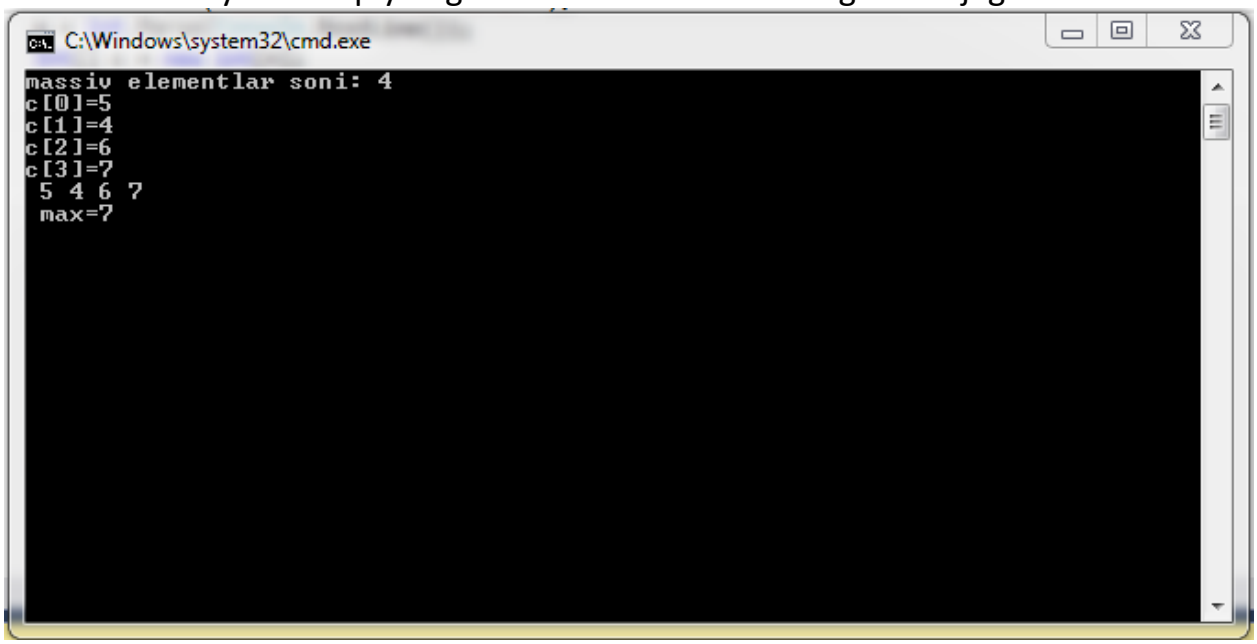
```
static void Main(string[] args)
      {
      }
```

S# dagi asosiy metod blokiga yoziladi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
         int i, n, max;
         Console.WriteLine("massiv elementlar soni");
         n = int.Parse(Console.ReadLine());
         int[] c = new int[n];

         for (i = 0; i < n; i++)
         {
```

```
            Console.Write("c[{0}]=", i);
            c[i] = int.Parse(Console.ReadLine());
        }
        max = c[0];
        for (i = 0; i < n; i++)
        {
            if (max < c[i])
                max = c[i];
        }

        for (i = 0; i < n; i++)
            Console.Write(" " + c[i]);

        Console.WriteLine("\n max={0} ", max);
 Console.ReadKey();
     }
   }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasida quyidagi 1.1-rasm rasmda keltirilgan natijaga erishamiz.



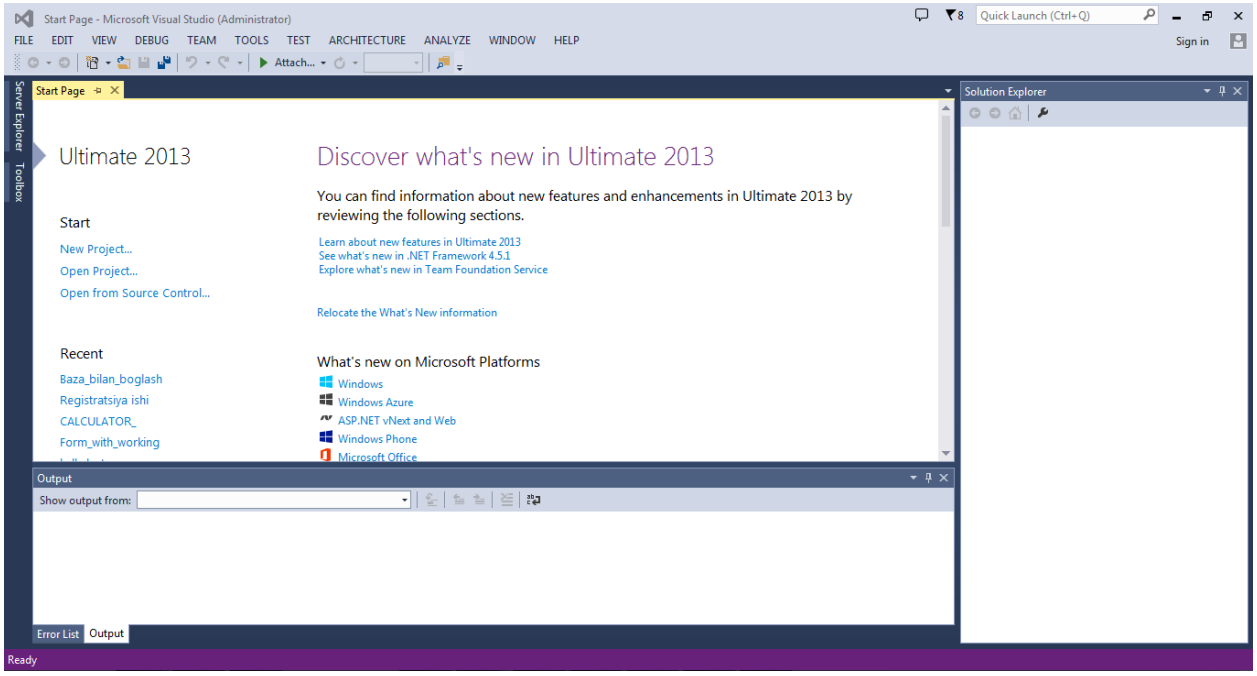
1.1-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun

nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

2-misol. Konsoldan n o‘lchamli massiv berilgan. Massivni juft elementlirini indeksi bo‘yicha tartiblab chop etish dastur tuzing.

Visual Studio 2013 (VS 2013) tizim ishga tushiriladi, **2.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



2.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

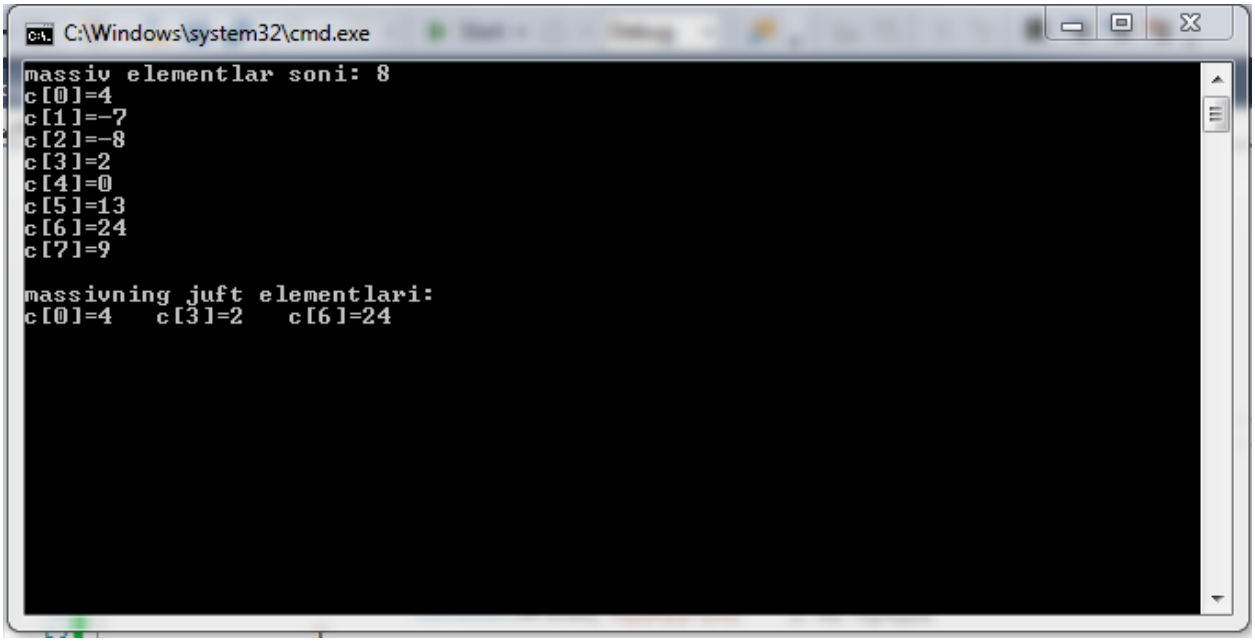
VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **SecondProgramm** kabi kiritib, **OK** tugmasini bosamiz.

Endi berilgan masala kodini kiritamiz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace SecondProgramm
{
   class Program
   {
      static void Main(string[] args)
      {
          int i;
         Console.Write("massiv elementlar soni: ");
         int n = int.Parse(Console.ReadLine());
         int[] c = new int[n];

         for (i = 0; i < n; i++)
         {
            Console.Write("c[{0}]=", i);
            c[i] = int.Parse(Console.ReadLine());
         }
         Console.WriteLine("\nmassivning juft elementlari:");
         for (i = 0; i < n; i++)
         {
            if (c[i] % 2 == 0)
            {
               if (c[i] != 0 && c[i]>0)
                  Console.Write("c[{0}]={1}   ", i, c[i]);
            }

         }

         Console.ReadKey();
      }
   }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. Massiv o‘lchamini kiritamiz va 1.3-rasmda keltirilgan natijaga erishamiz.



2.2-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliyot topshiriqlari

Boshlang‘ich bosqich

Topshiriq: bloksxema va tanlash operatorini qo‘llab masalaga muvofiq dastur yozing:

Elementlari 14 ta butun son bo‘lgan massiv kiriting. Qiymati juft bo‘lgan elementlari sonini toping.

Elementlari 12 ta butun son bo‘lgan massiv kiriting. 5-elementini shu massiv elementlarini o‘rta arifmetigi qiymatiga almashtirib yangi massiv hosil qiling.

Butun tipli, 11 ta elementdan iborat massiv berilgan. Absolyut qiymati massiv elementlarining o‘rta arifmetigi qiymatidan katta bo‘lgan elementlari sonini toping.

Butun tipli, 10 ta elementdan iborat massiv berilgan. Maksimal elementini birinchi elementi bilan joyini almashtiring.

Butun tipli, 9 ta elementdan iborat massiv berilgan. Maksimal elementini minimal elementi bilan joyini almashtiring.

20 ta elementdan iborat butun tipli massiv kiriting. Qiymatlari juft va toq

elementlaridan eng kattalarini toping.

Haqiqiy tipli, 15 ta elementdan iborat massiv berilgan. qiymati birinchi elementnikidan katta bo‘lgan elementlari sonini toping.

Butun tipli, 15 ta elementdan iborat massiv berilgan. Massiv elementlarini uning o‘rta arifmetigiga bo‘lib yangi massiv xosil qiling.

Butun tipli, 15 ta elementdan iborat massiv berilgan. Massiv elementlarini uning o‘rta arifmetigiga bo‘lib yangi massiv xosil qiling.

Butun tipli, 17 ta elementdan iborat massiv kiriting (elementlari musbat va manfiy). Manfiy sonlar modulining o‘rta arifmetigidan absolyut qiymati katta bo‘lgan elementlari yig‘indisini toping.

Butun tipli, 14 ta elementdan iborat massiv kiriting. Musbat sonlarning toqlari yig‘indisini va sonini xisoblang.

Haqiqiy tipli, 12 ta elementdan iborat massiv kiriting. Elementlarini kamayish tartibida joylashtiring. Maksimal va minimal elementlari yig‘indisini toping.

Butun tipli, 15 ta elementdan iborat massiv kiriting. Maksimal va minimal elementlari yig‘indisi va bo‘linmasini toping.

Butun tipli, 17 ta elementdan iborat massiv kiriting. 3 ga karralisini qiymati toq bo‘lgan elementlarini yig‘indisiga almashtiring.

Haqiqiy tipli, 14 ta elementdan iborat massiv kiriting. 1 dan 7 gacha bo‘lgan elementlarini ko‘payish, 8 dan 14 gacha bo‘lgan elementlarini kamayish tartibida joylashtiring.

Haqiqiy tipli, 12 ta elementdan iborat massiv kiriting. Maksimal va minimal elementlari orasida nechta element borligini aniqlang.

Butun tipli, 15 ta elementdan iborat massiv kiriting. Manfiy sonlar nechtaligini, musbat sonlarining ko‘paytmasini, qiymati nolga tenglari nechtaligini aniqlang.

Haqiqiy tipli, 12 ta elementdan iborat massiv kiriting. Elementlari qiymati joylashgan oraliq chegarasini aniqlang.

19 ta butun tipli elementdan iborat massiv berilgan. Birinchi manfiy songacha bo‘lgan elementlari yig‘indisini aniqlang. Agar manfiy son bo‘lmasa bu haqda habar chop etilsin.

Butun tipli, 16 ta elementdan iborat massiv kiriting. 3 ga karrali barcha elementlarini nol bilan almashtiring, almashtirishlar sonini aniqlang.

Berilgan M(12) haqiqiy sonli massivning minimal elementini 3 martta oshiring va oxirgi element bilan o‘rnini almashtiring.

M(15 haqiqiy sonli massiv berilgan. Elementlarini teskari tartibda chiqaring.

Butun tipli 14 ta elementdan iborat massiv kiriting. Indeksi juft bo‘lgan elementlarini yig‘indisini va toq elementlari ko‘paytmasin toping.

Haqiqiy tipli 12 ta elementdan iborat massiv kiriting. Oxirgi elementdan qiymati kichik bo‘lgan elementlari soni va yig‘indisini toping.

Butun tipli, 15 ta elementdan iborat massiv kiriting (elementlari musbat va manfiy). Berilgan massiv elementlari bo‘linmasidan va musbat sonlari yig‘indisidan yangi massiv hosil qiling.

Elementlari 15 ta haqiqiy tipli massiv berilgan. Barcha musbat sonlari ko‘paytmalari va barcha manfiy sonlari modulining ko‘paytmalari bo‘linmasini aniqlang.

Elementlari 19 ta butun son bo‘lgan massivdan maksimal elementini aniqlang va uni barcha juft bo‘lganlari qiymatiga o‘zlashtiring.

Butun tipli, 17 ta elementdan iborat massiv kiriting. Qiymati musbat sonlarining o‘rta arifmetigidan katta bo‘lgan elementlarining yig‘indisini va nechtaligini toping.

Haqiqiy tipli, 18 ta elementdan iborat massiv berilgan. Barcha musbat sonlari ko‘paytmasi va barcha manfiy sonlari modulli yig‘indisini bo‘linmasini butun qismini toping.

Haqiqiy sonli M(12) massiv berilgan. minimal elementi summasini va indeksini toping.

O‘rta bosqich

Topshiriq: masalaga mos ravishda blok sxema tuzing va dastur yozing.

Har biri 10 ta elementdan iborat butun sonli massiv kiriting. Yangi massivga juft indekslariga 1-massivning toq indeksli elementini, toq indeksga 2-massivni juft indeksli elementini chop eting.

Butun sonli 8 ta (ikki xonali sonlar) elementdan iborat massiv kiriting. Massivning sonlarini kichik xonasidagi raqamlarni yangi massivda chop eting.

Butun sonli 17 ta (ikki xonali sonlar) elementdan iborat massiv kiriting. Shu massiv raqamlari yig‘indisini toping.

Ikki ta haqiqiy sonlardan iborat 9 ta va 7 ta elementli massiv kiriting. Ikkila massiv sonlarini o‘sish tartibida uchinchi massivga chiqaring.

10 ta elementlari butun tipli bo‘lgan X va Y massivlarini kiriting. Ikkila massivni bir xil elementlaridan iborat S massivni hosil qiling.

Y massivni 12 ta elementini $yi=i^2+2i+19,3 \cos i$ formula bo‘yicha qiymatini hisoblang. Ekranga shu massivni va birinchi bo‘lib qiymati o‘rta arifmetigidan kichiklarini,qolganlarni o‘rnini o‘zgartirmay yangi massivga yozib chop eting.

Haqiqiy tipli Z(16 massivi berilgan. juft indeksdagi sonlar yig‘indisi va indeksi 3 ga karrali sonlar yig‘indisi ayirmasini toping.

Berilgan butun sonli R(9) massivida musbat toq sonlarning eng katta indeksini toping.

Klaviaturadan butun tipli 15 ta elementdan iborat massiv X ni kiriting. Massiv Y ni elementlarini yi= cos xi^2 + 2,97 lg^2 i^2 formula bo‘yicha hisoblang. Ikkala massivlarini elementlarini kamayish tartibida uchinchi massivni hosil qiling.

Klaviaturadan butun tipli 17 ta elementdan iborat massiv X ni kiriting. Massiv Y ni elementlarini formula bo‘yicha xisoblang:Y massivni o‘sish tartibida, X ni kamayish tartibida saralang, X va Y massivni juft indekslaridagi elementdan yangi R massivni hosil qiling.

Butun tipli 9 ta (ikki xonali sonlar) elementdan iborat massiv kiriting. Shu massiv raqamlari yig‘indilaridan yangi massiv hosil qiling.

12 ta elementdan iborat, haqiqiy tipli massiv kiriting. Elementlarini kamayish tartibida joylashtiring. Joylashtirilayotgandagi qadamlar sonini aniqlang.

Butun sonli 11 ta elementdan iborat massiv kiriting. Ishorasi manfiy bo‘lgan toq sonlar yig‘indisini hisoblang va uchga karrali sonlarini shu yig‘indi qiymatiga o‘zgartiring.

Haqiqiy tipli 14 ta elementdan iborat massiv kiriting. Birinchi yarmidagi sonlarni ikkinchi yarmi bilan o‘rnini almashtiring. Bundagi qadamlar sonini aniqlang.

Haqiqiy sonlar massivi berilgan. Berilgan S haqiqiy sondan eng uzoqda joylashgan massiv elementini qiymatini va indeksini aniqlang.

10 ta elementdan tashkil topgan, butun sonli massiv berilgan. Birinchi manfiy songacha bo‘lgan elementlari soni nechtaligini va ularni yig‘indisini aniqlang.

Berilgan massivdagi lokal minimumlar sonini aniqlang. (Lokal minimumlar - bu yonma-yon turgan 3 ta sonlarni, o‘rasidagisi o‘ng va chapdagidan kichik bo‘lgani).

Berilgan sonli massivdagi lokalni maksimumlarni sonini toping.(Lokal maksimumlar - bu yonma-yon turgan 3 ta sonlarni, o‘rasidagisi o‘ng va chapdagidan katta bo‘lgani).

Butun sonli berilgan Z(15) massividan musbat, manfiy va nol sonlarini barchasini yig‘indisini toping va birinchi manfiy son va nol soni oralig‘ida joylashgan elementlar ketma-ketligini chiqaring.

Berilgan sonli massivdan monoton kamayadigan(har bir keyingi son o‘zidan oldingidan kichik) sonlar ketma-ketliginidagi sonlar indeksini chiqaring.

Berilgan sonlar massividan ikki marttadan ko‘p takrorlanadigan sonlarni o‘chirib tashlang.

Butun tipli 10 ta elementdan tashkil topgan massiv kiriting. Yangi massivni avval manfiy sonlarni keyin nollarni keyin musbat sonlarni ularni ketma-ketligini saqlagan holda kiritib hosil qiling.

Butun tipli 10 ta (ikki xonali sonlar) elementdan iborat massiv kiriting. Kiritilgan massiv sonlari raqamlarini ayirmasidan yangi massiv hosil qiling.

Butun tipli 15 ta elementdan iborat massiv kiriting. Shunday saralangki massivni manfiy sonlari boshida o‘sish tartibida, musbat sonlari oxirida kamayish tartibida bo‘lsin.

Har biri haqiqiy tipli 12 ta elementdan iborat 2 ta massiv berilgan. Birinch massivni ikkinchi massivda ham bor elementlarini nollar bilan almashtiring.

Butun sonli massiv berilgan. Elementlari monotonno ortadigan (har bir keyingi son oldingisidan katta) masssiv uchastkasi sonini aniqlang.

Butun sonli massiv berilgan. Juft indeksli elementlari yig‘indisini toq indeksli elementlari yig‘indisiga bo‘lishdan qolgan qoldiqni aniqlang.

Butun sonli massiv berilgan. Barcha elementlarining o‘rta arifmetigidan ortuvchi sonlar qanchanigini foizini aniqlang.

Ikkita massiv kiriting. Har bir massivni maksimal elementini aniqlang va ularni o‘rnini almashtirib qo‘ying.

Yuqori bosqich

Topshiriq: algoritmni sxemasini va masalani yechish dasturini tuzing.

Ikkilik sanoq sistemasidagi butun son berilgan, yani 0 va 1 ketma-ketligidan iborat son. Shu sonni o‘nlik sanoq sistemasiga o‘tkazuvchi dastur tuzing.

Ikkilik sanoq sistemasidagi butun son berilgan, yani 0 va 1 ketma-ketligidan iborat son. Shu sonni sakkizlik sanoq sistemasiga o‘tkazuvchi dastur tuzing.

Ikkilik sanoq sistemasidagi butun son berilgan, yani 0 va 1 ketma-ketligidan iborat son. Shu sonni o‘n oltilik sanoq sistemasiga o‘tkazuvchi dastur tuzing.

Ikkilik sanoq sistemasidagi kasr son berilgan, yani 0 va 1 ketma- ketligidan iborat nuqta bilan ajratilgan son. Shu sonni o‘nlik sanoq sistemasiga o‘tkazuvchi dastur tuzing.

Ikkilik sanoq sistemasidagi kasr son berilgan, yani 0 va 1 ketma- ketligidan iborat nuqta bilan ajratilgan son. Shu sonni sakkizlik sanoq sistemasiga o‘tkazuvchi dastur tuzing.

Ikkilik sanoq sistemasidagi kasr son berilgan, yani 0 va 1 ketma- ketligidan iborat nuqta bilan ajratilgan son. Shu sonni o‘n oltilik sanoq sistemasiga o‘tkazuvchi dastur tuzing.

Ikki xonali butun sonlardan iborat 15 ta elementdan tashkil topgan massiv kiriting. Berilgan massiv elementlaridagi sonlarni raqamlari xonasini almashtirib yangi massivga chop eting. Masalan: berilgan massiv 25 71 84... ,

yangi massiv 52 17 48...

9 ta elementdan (9 ta ikki xonali sonlar 8 lik sanoq sisitemasidagi) tashkil topgan massiv kiriting. Shu massiv elementlarini o‘nlik sanoq sistemasiga o‘girib yangi massivga chop eting.

Butun sonli 7 ta elementdan (7 ta ikki xonali sonlar) tashkil topgan massiv kiriting. Elementlarning katta xonalarida joylashgan raqamlardan yangi massivni chop eting.

7 ta va 9 ta haqiqiy sonlardan tashkil topgan 2 ta massiv kiriting.Ikkala massiv elementlarini kamayish tartibida uchinchi massivga chop etib yangi massivni hosil qiling.

12 ta ikkilik sanoq sistemasidagi sonlardan tashkil topgan massiv berilgan. Massivda ikki marttadan ko‘p takrorlanadigan elementlarini o‘chiring.

Faqat 2 ta bir xil son qatnashadigan massiv kiriting. Shu sonlarni joylashgan joyini aniqlang.

Ikkilik sanoq sistemasidagi butun son berilgan, yani 0 va 1 ketma-ketligidan iborat. Massiv elementlarini siklik tarzda chapga ikki xona chapga suring. Berilgan va hosil bo‘lgan sonni ayirmasini toping.

Sonlari ikkilik sanoq sistemasidagi sonlardan iborat massiv berilgan. Massivni kamayish tartibida saralang. Sonlar yig‘indisini aniqlang.

Sonlari ikkilik sanoq sistemasidagi sonlardan iborat massiv berilgan. Massivni o‘sish tartibida saralang. Sonlarni o‘rtacha qiymatini aniqlang.

Sonlari ikkilik sanoq sistemasidagi sonlardan iborat massiv berilgan. maksimal va minimal elementlari o‘rnini almashtiring.

Ikkilik sanoq sistemasidagi butun son berilgan, yani 0 va 1 ketma-ketligidan iborat. Massiv elementlarini siklik tarzda chapga ikki xona o‘ngga suring. Berilgan va hosil bo‘lgan sonlar yig‘indisini toping.

Butun sonli massiv berilgan. Elementlari monotonno ortadigan (har bir keyingi son oldingisidan katta) joylardagi sonlar yig‘indisi va monoton kamayadigan(har bir keyingi son o‘zidan oldingidan kichik) joylardagi sonlar yig‘indisini o‘rtasidagi ayirmani toping.

Butun sonli massiv berilgan. Uning elementlari qiymati arifmetik progressiyani xolis qila oladim aniqlang. Agar “xa” bo‘lsa progressiya ayirmasini chiqaring, agar “yo‘q” bo‘lsa - javob “xosil qilmaydi”.

Butun sonli massiv berilgan. Uning elementlari qiymati geometrik progressiyani xolis qila oladim aniqlang. Agar “xa” bo‘lsa progressiya majrajini chiqaring, agar “yo‘q” bo‘lsa - javob “xosil qilmaydi”.

Butun sonli massiv berilgan. Qiymati o‘zidan o‘ngda turgan sondan katta bo‘lgan elementlarini indeksini chiqaring, bunday sonlar nechtaligini aniqlang.

Ikkilik sonlardan tashkil topgan massivni maksimal va minimal elementlari orasida nechta son borligini aniqlang. Agar bunday sonlar bo‘lmasa, habar

chop eting.

Berilgan ikkilik sanoq sistemasidagi massivda barcha elementlarini siklik bir xona o‘ngga suring.

Berilgan ikkilik sanoq sistemasidagi massivda barcha elementlarini siklik bir xona chapga suring. O‘zgartirishdan oldingi va keyingi sonlarning yig‘indisini toping.

Berilgan ikkilik sanoq sistemasidagi massivda barcha elementlarini qiymatini ikkilik sanoq sistemasidagi 1010 soniga oshiring.

Haqiqiy sonlar massivi berilgan. Berilgan R haqiqiy soniga eng yaqin bo‘lgan massiv elementini aniqlang(qiymati va indeksini).

Ikkilik sanoq sistemasidagi massiv berilgan. Berilgan D ikkilik sanoq sistemasidagi songa eng uzoq bo‘lgan massiv elementini aniqlang(qiymati va indeksini).

Ikkilik sanoq sistemasida manfiy va musbat sonlar berilgan. Shu sonlar yig‘indisini xisoblovchi dastur tuzing.

O‘nlik sanoq sistemasidagi 3 ta butun sondan iborat massiv berilgan. berilgan massivdagi sonlarni ikkilik sanoq sistemasiga o‘girib massiv hosil qiling.

Ikki o‘lchovli massivlar

Boshlang‘ich bosqich.

Toshiriq: algoritm blok-sxemasini va masalani xisoblovchi dasturini tuzing.

Butun sonli 3x4 ikki o‘lchovli massivni oxirgi qatoridagi elementlarini o‘sish tartibida saralang.

7x7 ikki o‘lchovli massiv berilgan. Manfiy toq elementlari modulini yig‘indisini toping.

5x6 ikki o‘lchovli massiv berilgan. har bir ustunning musbat sonlarini o‘rta arifmetigini aniqlang.

5 -tartibdagi haqiqiy sonlardan iborat kvadrat matritsa berilgan. Yon diaganalidagi eng kichik elementni aniqlang.

5x4 ikki o‘lchovli butun sonli massivni oxirgi ustun elementlarini kamayish tartibida saralang.

A matritsada(4 ta qator, 3 ta ustun) birinchi va uchinchi ustunlardagi eng katta elementlarni joylarini almashtiring.

B matritsada(3 ta qator, 4 ta ustun) birinchi va uchinchi ustunlardagi eng kichik elementlarni joylarini almashtiring.

B matritsada(3 ta qator, 4 ta ustun) birinchi va uchinchi ustunlardagi eng kichik elementlarni joylarini almashaHaqiqiy sonlardan tashkil topgan NxN kvadrat matritsa berilgan (N<=10). Har bir ustunning eng kichik sonlarini

ko‘paytmasini toping.tiring.

Ikki o‘lchovli 5x6 massiv berilgan. Har bir ustunni o‘rta arifmetigini va har bir qatorni minimum va maksimumlarini aniqlang.

Ikki o‘lchovli 7x8 massiv berilgan. har bir qatordagi toq sonlar qanchaligini aniqlang.

Ikki o‘lchovli nxm elementli massiv berilgan, massivdagi juft va toq sonlar nechtaligini aniqlang.

Ikki o‘lchovli nxm elementli massiv berilgan. Massiv elementlari orasida 7 nechi martta uchrashini aniqlang.

Ikki o‘lchovli nxm elementli massiv berilgan. Har bir ustundagi eng katta sonni aniqlang.

Ikki o‘lchovli nxm elementli massiv berilgan. Birinchi eng kichik elementini indeksini aniqlang.

N ta elementli kvadrat massiv berilgan. Oxirgi ustun elementlarini yig‘indisini toping.

N ta elementli kvadrat massiv berilgan. Birinchi satr elementlarini ko‘paytmasini toping.

Butun sonli 10x10 kvadrat massiv berilgan. Har bir qator elementlarini yig‘indisini toping.

Butun sonli 4x4 kvadrat massiv berilgan. Elementlari yig‘indisi eng kichik bo‘lgan qatorni toping.

Butun sonli 7x7 kvadrat massiv berilgan. Elementlari yig‘indisi eng katta bo‘lgan qatorni toping.

Butun sonli 6x8 kvadrat matritsa berilgan. Birinchi ustundagi musbat sonlarning ko‘paytmasini toping.

Butun sonli 4x6 kvadrat matritsa berilgan. Matritsaning har bir ustuni yig‘indisini toping.

Butun sonli 4x6 kvadrat matritsa berilgan. Qatorlarining hamma elementlari yig‘indilari orasidan eng kichigini qiymatini toping.

Butun sonli A[n, m] matritsa berilgan. Matritsa elementlarining o‘rta arifmatigidan katta bo‘lgan elementlari sonini xisoblang. n=4, m=5

N qator va M ustunlardan iborat ikki o‘lchovli massiv berilgan. Massivning ikkinchi qatoridagi elementlari yig‘ indisini toping.

Butun sonli 4x4 o‘lchamdagi matritsa berilgan. Shu matritsani ikkinchi ustunida joylashgan manfiy elementlar nechtaligini toping.

Butun sonli matritsa berilgan. 7 ta ustun va 3 ta qatordan iborat massivning har bir ustunida nechta element borligini aniqlang.

Ixtiyoriy matritsa uchun toq sonlarini yig‘indisini hisoblovchi dastur tuzing.

5x5 o‘lchovli butun sonli matritsa berilgan. 4 chi va 5 chi qatorlarini o‘rnini almashtiring.

Ixtiyoriy matritsa uchun elementlari yig‘indisi eng kichik bo‘lgan ustunni aniqlang.

Matritsaning ikkinchi ustunidagi musbat sonlar nechtaligini aniqlang.

O‘rta bosqich

Topshiriq: masalaga mos ravishda blok sxema tuzing va dastur yozing.

Ixtiyoriy matritsada ikkinchi manfiy sondan keyin joylashgan elementlarni kamayish tartibida saralang.

Ikki o‘lchovli massivni 0 va birlar bilan to‘ldirish kerak. Kiritilishdan so‘ng massiv quyidagi ko‘rinishga ega bo‘lsin:

0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0

Ikki o‘lchovli massivni to‘ldirish kerak. Kiritilishdan so‘ng massiv quyidagi ko‘rinishga ega bo‘lsin:

01 02 03 04 12 13 14 05 11 16 15 06 10 09 08 07

A(n, m) massiv berilgan. Bitta ham takrorlanuvchi elementi bo‘lmagan massiv qatorini o‘chiring.

3x3 massivni o‘sish tartibida spiral shaklida joylashtiring.

7 8 9 6 1 2 5 4 3

A matritsa elementlarini tasodifiy sonlar generatori yordamida hosil qiling. Bosh diagonal bo‘yicha minimal element joylashgan qatorni o‘chirib, B matritsani hosil qiling.

Kvadrat matritsani n natural sonlar 1, 2, 3, ... , n2 tartibida soat strelkasiga qarshi spiral bo‘yicha yozib to‘ldiruvchi dastur tuzing.

Kvadrat matritsani n natural sonlar 1, 2, 3, ... , n2 tartibida soat strelkasiga qarab spiral bo‘yicha yozib to‘ldiruvchi dastur tuzing.

Ikki o‘lchovli butun sonli massiv berilgan A(M,N). Shu massiv qatorlari nomerlaridan bir o‘lchovli B massivni tuzing.

A (N,M) sonlar matritsidan berilgan B sonidan qiymati yuqori bo‘lgan barcha elementlarini aniqlovchi dastur yozing. Bunday elementlar nechtaligini sanang va ularni K massivga yozing.

A (N,M) sonlar matritsidan klaviaturadan kiritilgan songa teng bo‘lgan barcha elementlarini aniqlovchi dastur yozing. Bunday elementlar nechtaligini sanang.

Ikki o‘lchovli A[5,10] massiv berilgan. Matritsa elementlarini modul bo‘yicha eng kattasiga bo‘lib yangi matritsani hosil qiling.

Ikki o‘lchovli massiv berilgan. Birinchi qatorni minimal element joylashgan

qatordan keying qatorga qo‘ying.

Butun sonli B[1.5, 1.5] massiv berilgan. Chap diagonaldan pastda joylashgan elementlarini ko‘paytmasini toping.

Butun sonli B[1.5, 1.5] massiv berilgan. Chap diagonaldan yuqorida joylashgan elementlarini yig‘indisini toping.

Butun sonli 5x5 matritsa berilgan. Shu matritsadagi birinchi qatorning barcha manfiy sonlarni 0 bilan almashtiring.

Ikki o‘lchovli 5x5 massiv berilgan. Matritsa elementlarini modul bo‘yicha eng kattasiga bo‘lish yo‘li bilan yangi matritsani hosil qiling.

Butun sonli to‘g‘ri to‘rtburchakli M-N o‘lchovli matritsa berlgan. Juft raqamdagi ustunni o‘sish, toq raqamdagisini kamayish tartibida saralang.

Butun sonli 8x5 o‘lchovli matritsa berilgan. Aniqlang:

a) massivning 2-ustuni elementlari yig‘indisini;

b) massivning 3-ustuni elementlari yig‘indisini;

Butun sonli to‘g‘ri to‘rtburchakli M-N o‘lchovli matritsa berilgan. [1,20] oraliqda joylashgan elementlaridan bir o‘lchovli massiv xosil qiling. Xosil qilingan bir o‘lchovli massivning elementlarini o‘rta arifmetigini toping.

Butun sonli to‘g‘ri to‘rtburchakli M-N o‘lchovli matritsa berilgan. [1,20] oraliqda joylashgan elementlaridan bir o‘lchovli massiv xosil qiling. Xosil qilingan bir o‘lchovli massivning elementlari ko‘paytmasini toping.

Butun sonli kvadrat matritsa berilgan. Har bir qatordagi eng katta elementini topib, uni asosiy diagonal elementi bilan o‘rnini almashtiring.

Butun sonli kvadrat matritsa berilgan. Indekslari yig‘indisiga karrali bo‘lgan minimal qiymatdagi elementlar joylashgan ustunni (ustun raqamini) ko‘rsating.

Butun sonli kvadrat matritsa berilgan. Asosiy diagonaldan yuqorida joylashgan elementlari summasini toping.

Berilgan kvadrat massiv o‘zining asosiy diaganaliga nisbatan simmetrikmi aniqlang.

Berilgan kvadrat massiv o‘zining asosiy diaganaliga nisbatan simmetrik emasmi aniqlang.

Ikkita n va m sonlar berilgan. Ikki o‘lchovli int A[n][m] massiv hosil qiling. Uni ko‘paytirish jadvali bilan to‘ldiring A[i][j]=i*j va ekranga chiqaring. Bunda ichma ich sikldan foydalanmang, massivni to‘ldirishlar faqat bitta sikl bilan amalga oshirilsin. Masalan: for(i=0; i<n; ++i).

Butun sonli NxM o‘lchovli matritsa berilgan. Bir hil elementlardan eng kichigi joylashgan qator nomerini chiqaring.

Butun sonli kvadrat matritsa berilgan. Asosiy diagonaldan pastda joylashgan elementlari ko‘paytmasini toping.

Butun sonli NxM o‘lchovli matritsa berilgan. Bir hil elementlardan eng kattasi joylashgan qator nomerini chiqaring.

Yuqori bosqich:

Topshiriq: algoritmni tizimli sxemasini va masalani yechish dasturini tuzing, kiruvchi qiymatlarni klaviaturadan kiriting va natijani ekranga chiqaring.

A(xi, yi) nuqtadagi tekkislikda berilgan massivni eng ko‘p nuqtalar soni yotadigan doirani radiusi va markazini aniqlang.

1-kurs studentlaridan iborat massiv va ularning 100 metrga yugirish bo‘yicha natijalaridan iborat massiv berilgan. Estafetada qatnashish uchun 4 ta eng yaxshi yuguruvchilardan iborat jamoa tuzing.

Tasodifiy sonlardan L(I,J) massiv tuzing. Massivni har bir elementini 3 martta oshiring va ishorasini qarama-qarshisiga almashtiring. Massivni ekranga jadval ko‘rinishida chiqaring.

Kvadrat matritsa berilgan. Asosiy diagonaldan pastda joylashgan elementlaridan bir o‘lchovli massiv hosil qiling. Bir o‘lchovli massivni saralang. Bir o‘lchovli massivni saralashdan oldingi va saralashdan keying xolatini chop eting.

Ikki o‘lchovli ixtiyoriy massivni uch xil (pufaksimon, qo‘yish, tanlash) usulda saralashni amalga oshiruvchi dastur tuzing.

Butun sonli MxN o‘lchovli, elementlari 0 dan 100 gacha bo‘lgan massiv berilgan. Agar matritsa har xil qatorlarining ko‘p sonlari o‘xshash bo‘lsa, matritsa qatorlarini o‘xshash diymiz. Birinchi qatorga o‘xshash qatorlar nechtaligini toping.

Matritsaning har bir qatorida k ga teng koordinata elementlarini toping(agar ular bo‘lsa). Qidirish usuli- barer bilan ketma-ketlikda.

Matritsaning har bir qatorida k ga teng koordinata elementlarini toping(agar ular bo‘lsa). Qidirish usuli- barer bilaMassiv A[3][3] berilgan. elementlari yig‘indisi minimal bo‘lgan qatorni aniqlang va minimal yig‘indini matritsaning barcha elementlariga ko‘paytirib chiqing.an ketma-ketlikda.

N tartibli butun sonli kvadrat matritsa berilgan. Matritsa elementlarini qayta shunday joylashtiringki, uning oxirgi elementlati o‘sish tartibida joylashmasin. Sanab saralash.

5x10 o‘lchamdagi matritsa berilgan. elementlari monoton kamayadigan ustunlar sonini chiqaring.

Barcha elementlari har xil nxm tartibli matritsa berilgan. Har bir qatordagi qiymati eng kichik elementni tanlang, so‘ng bular orasidan eng kattasini. Shu elementni indeksini chop eting.

O‘sish yoki kamayish tartibida joylashgam matritsa ustunlari elementidan maksimalini aniqlang. Agar bunday ustun bo‘lmasa, 0 ni chop eting.

Natural sonlardan iborat, NxN kvadrat matritsa berilgan. Uning

elementlarini yon diagonalga nisbatan aksini chiqaring.

Natural sonlardan iborat, NxN kvadrat matritsa berilgan. Uning elementlarini yon diagonalga nisbatan natural sonlardan iborat, NxM matritsa berilgan. Qatorlarda eng chapdagi kichik elementni tanlang va uni birinchi ustunga qo‘ying. aksini chiqaring.

8-mavzu. Qism dasturlar

Reja

1. Kirish
2. Qism dasturlarning o‘zagi
3. Qism dasturlar uchun loyixalar tuzish
4. Lokal yo‘naltiruvchi muxitlar
5. Vaqtinchalik usullar parametri
6. Qism dastur parametrlari
7. O‘rnatilgan qism dasturlar
8. Alohida va mustaqil kompilyatsiya

Qism dasturlar dasturlarning asosiy o‘zagi xisoblanib dasturlash tilida eng muhum loyiha hisoblanadi. Biz xozir qism dasturlar loyihasini, uning ichidagi parametr metodlarini, lokal ma’lumot muxitini, ko‘plab umumiy qism dasturlarni va boshqalarni shuningdek qism dastur bilan bog‘liq muammoli vaziyat tasirlarni o‘rganamiz. Biz shuningdek qism dasturlar paneli bilan bog‘liq muxokamalarni o‘rganamiz.

Qism dasturlarni amalga oshirish usullari keyingi mavzularda batafsil yoritilgan.

1 Kirish

Dasturlash tili ikkiga ajralmaydigan abstraksiyalarni o‘z ichiga oladi: jarayon abstraksiyasi va ma’lumot abstraksiyasiyalarini yuqori dasturlash tilining dastlabki bosqichlarida faqat jarayon abstraksiyasidan foydalanilgan. Jarayon abstraksiyasi qism dasturlarning shakli bo‘lib, xamma dasturlash tillarida asosiy tushuncha xisoblanadi. Biroq 1980-yilda ko‘p odamlar ma’lumot abstraksiyasiga ishondi. Malumot abstraksiyasi batafsil keyingi mavzularda muxokama qilinadi.

Dasturlashtirilgan 1-kompyuter “Babbage’s Analytical Engine” ni bo‘lib, 1840-yilda tuzilgan. Dasturning bir necha turli joylarida qo‘llanma kartalarning qayta ishlangan to‘plamlarning qobilyati bor edi. Zamonaviy dasturlash tilida bunday operatorlar to‘plami qism dasturlarday yoziladi. Bunday qayta ishlangan natijalar bir necha turdagi saqlashlarda ya’ni xotira maydonini va kodlash vaqtini tejashda ishlatiladi. Operator tufayli dasturda qism dasturlarning o‘z joylariga qo‘yilgan hisoblash tavsilotlari uchun qayta ishlangan natijalar abstraksiyasi bo‘lib xam xisoblanadi. Tasvirlash o‘rniga dasturda bajariladigan bir necha xisoblashlar ya’ni tavsifi (qism dasturdagi operatorlar to‘plami) chaqirilgan operator tomonidan qabul qilingan samarali abstraktning tavsilotlari hisoblanadi. Bu jarayon programmani ifodalash ya’ni past darajadagi tavsilotlarni yashirishda mantiqiy tuzilishni ifodalaydi.

Ob’ekt yo‘naltirilgan tillar metodlari qism dasturlarga yaqindan bog‘liqligi bu bobda muhokama qilindi. Asosiy metod yo‘li qism dasturlardan o‘rnining birlashgan turkumi va ob’ekt bilan farq qiladi. Metodlarning bu maxsus xususiyatlari ya’ni qism dasturlar bilan bog‘liq qism xususiyatlari, shuningdek parametrlar va lokal o‘zgaruvchilar keyingi mavzularda muxokama qilinadi.

2 Qism dasturlarning o‘zagi

2.1 Qism dasturlarning umumiy xusisiyatlari

Bu mavzuda xar bir dasturlash tilida qism dasturlar muxokama qilingan va ularning quyidagi hususiyatlari bor.

- Har bir qism dasturni kirish nuqtasi bor.
- Qism dasturda murojaat qilinayotgan paytda faqat murojaat qilinayotgan qism dasturdan tashqari dasturning boshqa qismlari to‘xtatiladi.
- Qism dasturni bajarish to‘xtaltilganda nazorat murojaatchiga qaytadi.

Bir biriga bog‘liq qismlar va boshqarishlar natijalarini tanlash (keyingi mavzularda).

Ko‘pchilik qism dasutlarni nomi bor xattoki nomalumlarini ham. 9.12 qismda noma’lum qism dasturlarga misollar berilgan.

2.2 Asosiy ta’riflar

Qism dastur ta’rifi abstraksion qism dasturlar harakati va interfeysini o‘zida aks ettiradi. Murojaat qilinayotgan qism dasturdan o‘ziga hos aniq ishlash talab qilinadi. Qism dastur faol holatda bo‘lishi uchun murojaat qilingan ishlarni bajarishi va faoliyatini to‘htatmasligi kerak. Qism dasturlarni 2 ta asoiy turlari bor: protsedura va funksiyalar va ular 2.4 qismda muxokama qilingan.

Qism dastur sarlavhasi ya’ni ta’riflarning birinchi qismi bir necha hizmat maqsadlarni o‘zida qamrab oladi. Birinchidan u quyidagi sintaktik birlik ayrim konkret turdagi bir qism dastur ta’rifi ekanligini bildiradi. Bir turdan ortiq qism dasturlar, qism dasturlar turlari mahsus so‘z bilan aniqlanadi. Ikkinchi qism dastur anonim (yashirin) bo‘lmasa, qism dastur uchun nom beradi. Uchinchidan ularni parametrlarni aniq ro‘yhati deb belgilash mumkin.

Boshqaruvni quyidagi misollar bilan ko‘rib chiqishimiz mumkin:

Def adder (PARAMETRI)

Bu Python nomidagi qism dastur sarlavhasi hisoblanadi. Ruby deb ataluvchi qism dastur sarlavhasi ham “Def” bilan boshlanadi. Javascript qism dastur sarlavhasi esa « function» bilan boshlanadi.

S dasturlash tilida funksiya sarlovhasi quyidagicha nomlanishi mumkin:

Void adder(parametrlari)

Void so‘zi bu sarlavhasini ko‘rsatadi ya’ni qism dastur orqaga qaytmasligi uchun belgilanadi.

Qism dastur tanasi uning harakatini belgilaydi. S tilida asosiy tillarda (masalan Javascript) qism dastur tanasi qo‘shtirnoq bilan ajratiladi. Ruby da esa qism dastur tanasi “end” termini bilan yoziladi. Operatorlar tarkibida ya’ni Python funksiyasi tanasida nazarda tutilgan operator va tananing ohiridagi operator nazarda tutilmaydi.

Python funksiyasi hususiyatlaridan biri boshqa dasturlash tillari funksiyalaridan farqi def operator so‘zi bilan yoziladi. Qachonki def operatori bajarilsa, berilgan nom berilgan funksiya tanasiga tasdiqlanadi. DEF funksiyasi bajarilgunga qadar, funksiyani nomlash mumkin emas. Quyidagi misolda ko‘rishimiz mumkin:

```
if...
deffun(...):
...
else
deffun(...):
...
```

Agar keyingi punkt bajarilagn bo‘lsa, funksiyaning boshqa versiyasida nomlanishi mumkin, lekin bu versiyada qayta nomlanmaydi. Agar keyingi qator yana tanlansa , funksiyaning bu versiyasida nomlash mumkin lekin bir banddan keyingisida mumkin emas.

Rybu qism dasturining usullari boshqa dasturlash tillaridan farq qiladi. Rybu usullari asosan belgilangan sinflarda beligilanadi, bundan tashqari belilangan sinf tashqarisida ham belgilanishi mumkin, bu holda ular ildiz ob’ekt usullari bilan ko‘riladi. Bunday usullar qabul qiluvchi ob’ektdan tashqari deb ataladi, xuddi S yoki S ++ kabi. Agar Rybu usullari ob’ektdan tashqari deb atalsa, Self deb qabul qilinadi. Agar sinfda metod bo‘lmasa, sinf ichidan izlanadi.

Lue funksiyasi ham anonoimlar hisoblanadi, garchi ularning nomlari sintaksis yordamida paydo bo‘lgan bo‘lsa ham. Misol uchun kub funksiyaning belgilangan ta’riflarini ko‘rib chiqamiz.

```
function cube(x) return x * x * x end
cube = function (x) return x * x * x end
```

Birinchi kelishilgan sintaksis ikkinchi aniq bezalgan nomsiz funksiyani shakli hisoblanadi.

Qism dasturlarning qisqacha parametrlari raqamli va rasmiy parametrlar turlarini o‘z ichiga oladi. Qism dasturning protakoli qisqacha parametrlarga qo‘shimcha ravishda qaytib kelgan turi uchun qo‘shimcha hisoblanadi. Dasturlash tillarida qism dasturlash turlari bor, bu turlar qism dasturning protakolida belgilanadi.

Qism dastur e’lon qilishlari huddi ta’riflar kabi hisobga olinadi. Parallel o‘zgaruvchan ta’riflar va e’lon qilishlar shakli S da qaysiki e’lon qilishlar ma’lumot turlarini aniqlashda ammo o‘zgaruvchan turlarini aniqlashda emas. Qism dastur e’lon qilishlari qism dastur protakoli bilan taminlanadi lekin ularning tanasini o‘z ichiga olmaydi. Ular kerakli tillarda qaysiki qism dasturlarda ruhsat berilmaydigan. Ikkala ya’ni o‘zgaruvchan voqealar va qism dasturlar, e’lon qilishlar dasturning statik turi uchun tekshirilishi kerak. Qism dasturlarda, parametrlarning bu turlari tekshirilishi kerak. Yuqorida ko‘rib o‘tilgan funksiya e’loni S va S++ dasturlash tillarida ko‘p uchraydi va bu prototiplar deb ataladi. Bunday e’lon qilishlar odatda bosh fayllarda joylashtiriladi.

Ko‘pchilik boshqa tillarda (S va S++ dan tashqari) qism dastur e’lon qilishlari kerak emas, chunki ular nomlanishidan oldin qism dasturlarni belgilash talab qilinmaydi.

3 Qism dasturlar uchun loyixalar tuzish…

Qism dastur murakkab tuzilmalari dasturlash tillarida bor, ya’ni ularning loyixalari ishtirok etgan uzun ruyxatda bir yoki bir necha parametr usullarini tanlash ochiq oydin masalalarda ishlatiladi. Turli tillarda yozilgan mavzular turlicha fikrlarni aks ettiradi. Yaqindan bog‘liq masala haqiqiy parametrlar turlari nazorat qilinadi mos rasmiy parametr turlari bilan.

Qism dastur lokal atrof muxit tabiatiqism dastur tabiatiga bog‘liq. Eng muxim savol bu lokal o‘zgaruvchilar doimiy yoki dinamik usulda ajratilganmi yuqmi shu hisoblanadi.

Keyingi savol bu qism dasturlar bor yuqligi yana bir qism masala qism dastur nomlari parametrlar sifatida berilishi mumkin emasligi. Agar qism dastur nomlari parametrlar sifatida o‘tsa, qism dasturlarga tilda ruhsat etiladi, to‘g‘ri yo‘naltirilgan savol bor, qism dasturning muhiti parametr sifatida qabul qilinadi.

Nixoyat savollar bor ya’ni xaddan ortik qism dasturlar bormi yukmi degan. Xaddan ortik qism dasturlar bor bo‘lsa ular bir xil nom bilan yuritiladi xuddi boshka

dasturlar kabi. Umumiy qism dasturda xar xil turdagi turli xil ma’lumotlarni xisoblash mumkin. Qism dastur bazasi va uning tasnifi muxit kaysiki dasturda qism dastur bilan birgalikda nomlanadigan.

Kuyidagi loyixalar tuzish qism dastur uchun umumiy. Ko‘shimcha vazifalari bilan borgli loyixalar 9.10 kismda muxokama kilinadi.

- Mahalliu o‘zgaruvchilar doimo dinamik ravishda ajratilganmi?
- Qism dastur ta’riflari boshka qism dastur tariflarida paydo bulgan bulishi mumkinmi ?
- Parametrda kanday usul va usullar ishlatiladi?
- Xakikiy parametr turlari rasmiy parametr turlariga qarshi tekshiriladimi ?
- Agar qism dasturlar parametr sifatida o‘tsa qism dastur muxitiga alokasi bo‘lishi mumkinmi ?
- Qism dasturni xaddan tashkari olish mumkinmi ?
- Qism dastur umumiy bo‘lishi mumkinmi ?
- Agar qism dasturlar til bilan bog‘liq bo‘lsa, yopilish ko‘llab- kuvatlanadimi ?

Ushbu masalalar va misollar kuyidagi bo‘limda muxokama kilinadi.

4 Lokal yo‘naltiruvchi muxitlar

Ushbu bo‘limda qism dastur doirasida belgilangan parametrlarga erishish bilan bog‘liq masalalar muxokama qilinadi. Qism dastur masalasi xam o‘rin olgan.

4.1 Lokal o‘zgaruvchilar

Qism dastur o‘zining o‘zgaruvchilarni aniqlay oladi, shuningdek lokal muxit o‘zgaruvchilarini xam. Qism dasturlar ichida belgilangan o‘zgaruvchilar lokal o‘zgaruvchilar deyiladi, chunki ular unda belgilangan ya’ni ularning kulami qism dasturlar organi hisoblanadi. 5 bob terminalogiyada lokal o‘zgaruvchilar statik yoki dinamik bo‘lishi mumkin.

Lokal o‘zgaruvchilar dinamik bo‘lsa, qism dastur ijrosi boshlanadi va saqlash qachondan erkin bo‘lsa, ular saklash uchun bog‘langan deb yakun yasaydi. Dinamik lokal o‘zgaruvchilarni bir necha afzalliklari bor ya’ni asosiysi qism dastur taqdimiga moslashuvchan bo‘ladi. Qism dasturlar dinamik lokal o‘zgaruvchilarga ega bo‘liщi muhimdir. Dinamik qism dasturlarni yana bir afzalligi shundaki barcha harakatsiz lokal o‘zgaruvchilar bilan faol qism dasturlar orasida aloqa o‘rnatishdir. Bu kata afzalligi emas qachonki kompyuterni qism xotirasi bo‘lganda.

Dinamik lokal o‘zgaruvchilarning asosiy kamchiliklari kuyidagilardan iborat: birinchi joylashtirish uchun vakt qiymati boshlangich va har bir qism dasturga

chaqiruv uchun joylashtirish. Ikkinchidan murojaat uchun dinamik maxlliy o‘zgaruvchilar bilvosita bo‘lishi kerak, statik lokal o‘zaruvchilar yesa to‘g‘ridan to‘g‘ri.

Bu indirectness talab qilinadi chunki faqat ishlayotgan paytda (keyingi mavzuda berilgan) lokal o‘zgaruvchilar aniklashi mumkin. Nixoyat barcha lokal o‘zgaruvchilar dinamik bo‘lsa, qism dastur tarixi sezgir bo‘lishi mumkin emas, ya’ni ular maxlliy o‘zgaruvchilar va chakiruvlar orasidagi ma’lumotlarni saqlab qola olmaydi. Bu bazan qism dasturlarga yozish imkoniyatiga ega bo‘ladi. Qism dastur tarixi uchun quyidagi misolni ko‘ramiz. pseudorandom raqamlari uchun bir vazifani, hisoblashlardan foydalanib, shunday qism dastur pseudorandom raqamlari. shuning uchun xar bir statik maxlliy ma’lumotlarni saqlash kerak. Tartib va qism dasturlar iterator strukturasida (oldingi mavzudda muxokama kilingan) da foydalaniladi ya’ni dastularning boshka misollari kiradi.

Statik lokal o‘zgarvchilarni afzalligi dinamik maxalliy o‘zgaruvchilardan ko‘ra joylashtirish va qaytarishga ko‘prok vakt sarflaydi. Shuningdek to‘g‘ridan to‘g‘ri murojaat qilish mumkin bo‘lsa, bu yanada samarali bo‘ladi va albatta ular bir qism dastur tarixi bo‘lish imkonini beradi. Statik lokal o‘zgaruvchilarning yana bir afzalligi ularning qobilyatini qaytadan qo‘llab quvatlaydi. Shuningdek ularni saqlash boshka lokal qism o‘zgaruvchilar bilan birgalikda bo‘lmaydi.

Ko‘pchilik zamonaviy dasturlash tillarida qism dastur lokal o‘zgaruvchilarni dinamik vazifasini bajarmaydi. S va S++ vazifalari lokal dinamik o‘zgaruvchilar ayniqsa statik o‘zgaruvchilar deb e’lon qilinadi. Masalan quyidagi S va S++ vazifalarida o‘zgaruvchilar sum statik va count dinamik o‘zgaruvchilardir.

```
int adder(int list[], int listlen) {
static int sum = 0;
int count;
for (count = 0; count < listlen; count ++)
sum += list [count];
return sum;
```

C++, Java, and C# fakat dinamik lokal o‘zgaruvchilar. Python da fakat e’lon qilishlar va ta’rif usullari uchun ishlatiladi. E’lon xar kanday global o‘zgaruvchan bo‘lishi uchun usul xam belgilangan tartibda o‘zgaruvchan bo‘lishi kerak. O‘zgaruvchilar usuldan tashqarida belgilanishi mumkin global deb e’lon qilinmasa ham. Agar global o‘zgaruvchining nomi usulda belgilangan bo‘lsa, lokal va global deb e’lon qilinmagan deyish mumkin.

Hamma lokal o‘zgaruvchilar phuton da dinamik hisoblanadi. Faqat lua da o‘zgaruvilar cheklangan deb e’lon qilinadi. Har qanday blok lokal o‘zgaruvchilarni kuyidagicha e’lon qilishi mumkin:

Local sum

Barcha e’lon qilinmagan o‘zgaruvchilar lua da globaldir. Lua DA lokal o‘zgaruvchilar global o‘zgaruvchilardan ko‘ra tezroq. (2006)

4.2 Ichki qism dasturlar

Ichki qism dasturlar g‘oyasi Algol 60 bilan tashkillashtirildi. Bu turtki mantiq va sohalar nazaryasini paydo qildi. Agar qism dasturi boshqa qism dasturlarsiz kerak bo‘lsa, nimaga u joyida yo‘q yoki boshka programmalardan yashirilgan? Chunki statik soxa odatda tillarda foydalaniladi qaysiki qism dastur ichiga ruxsat berilgan, shuningdek yuqori strukturali yo‘llar lokal bo‘lmagan o‘zgaruvchilarni qism dastur ichiga taminlaydi. Eslatib o‘tamiz 5-darsda joriy muammolar muhokama qilinadi. Uzoq vaqt davomida ichki qism dasturlar ya’ni tillarda ruxsat berilgan: Algol 60, Algol 68, Pascal, and Ada. Ko‘p boshqa tillar jumladan S ning avlodlari ichki qism dasturlarga ruxsat bermaydi. Yaqinda yana bazi bir tillarga ruxsat berildi. Ular orasida JavaScript,Python, Ruby, and Lua lar bor. Shuningdek bazi funksional dasturlash tillari ichki qism dasturga ruxsat berdi.

5 Vaqtinchalik usullar parametri

Vaqtincha usullar parametri yo‘llar qaysiki parametr uzatiladigan qism dasturlar nomlaydigan. Birinchidan biz vaqtincha usullar parametrining xar xil semantik modellarini o‘ylaymiz. Keyin biz semantik modellar uchun til loyihalari tomonidan kashf qilingan usullarni muxokama qilamiz. Nixoyat biz usullar ichidan tanlangan til loyihalari loyihalarini ko‘rib chiqamiz.

5.1 Vaqtincha parametrlarning semantik modellari

Rasmiy parametrlar 3 turdagi semantik modellarning biri bilan ifodalanadi. Ular haqiqiy parametr ma’lumotlarini olish mumkin. Ular haqiqiy parametrlarga ma’lumotlar uzatishi mumkin yoki ikkalasini ham. Bu modellar ichki, tashqi va ichki va tashqi modellarga bo‘linadi. Masalan qism dastur uchun olingan parametr qiymatlarini ko‘rib chiqamiz.

—list1 va list2.

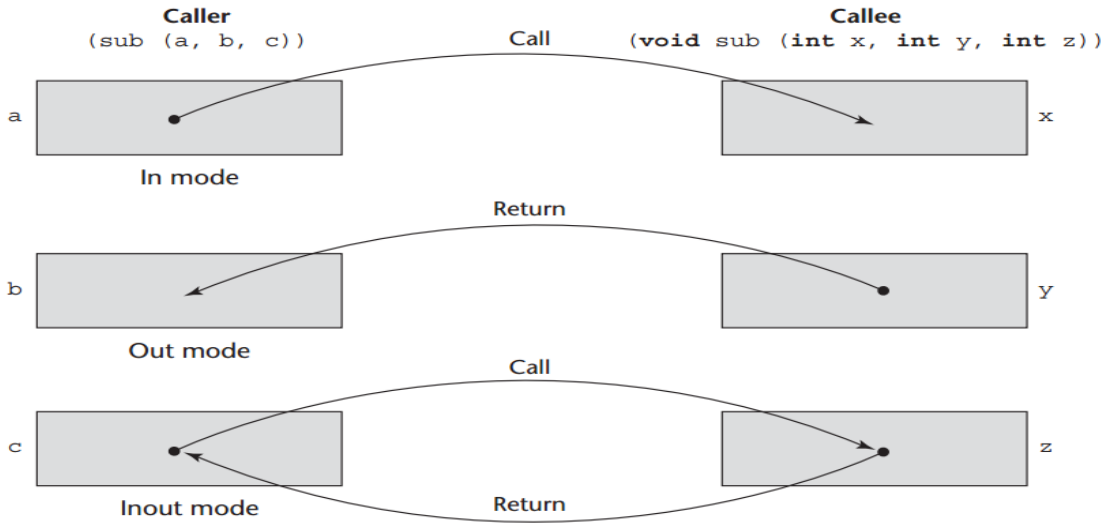
Qism dasturga list 1 ni list 2 ga qo‘shing va list 2 natijasini qayta ko‘rib chiqamiz. Bundan tashqari qism dastur yangi qator yaratishi kerak ikkita berilgan

qatorlar va ularni qaytarib. Bu qism dastur uchun list 1 bo‘lishi kerak emas, u qism dastur tomonidan yaratilishi kerak. Bizga list 2 kerak chunki list 2 yangi qiymat beradi va uni qaytarishi kerak. Uchinchi qator tashqarida bo‘lishi kerak chunki bu qator uchun boshlang‘ich qiymat yuq va foydalanuvchiga qiymatni hisoblab berishi kerak.

Ma’lumotlarni parametrga uzatish uchun 2 ta model bor: yoki haqiqiy qiymat ko‘chiriladi yoki bir erkin foydalanish yo‘li uzatiladi. Eng keng tarqalgan yo‘li oddiy ko‘rsatkich yoki mos yozuvlar bo‘ladi. 9.1 shaklda qachonki qiymatlar ko‘chirilganda parametrlarning 3 ta semantik modeli ko‘rsatilgan.

5.2 Parametrni amalga oshirish modellari

Dasturchilar tomonidan ishlab chiqilgan turli modellarning uchta asosiy



parametr uzatish usullari bor. Quyidagi qismda biz ularning nisbiy kuchli va ojiz tomonlarini muhokama qilamiz.

5.2.1. Qiymat tomonidan uzatish

Agar parametr qiymati bo‘yicha uzatiladigan bo‘lsa, haqiqiy parametr qiymati ishlatiladi, keyin lokal parametr sifatida bajaradi shu tariqa rejimda semantik usul amalga oshiriladi.

Ko‘pincha o‘rtacha qiymat odatda, nusxa tomonidan amalga oshiriladi, bu yondashuv bilan yanada samarali bo‘ladi. Yozish, himoya amalga oshirish har doim oddiy masala emas. Masalan navbatdagi parametr o‘tib bo‘lgan qism dastur boshqa qism dastur uchun samarali. Bu nusxasini uzatishni amalga oshirishni yana bir

sababidir. Biz 5.4 qismda ko‘ramiz, parametr qiymatlarini uzatish yo‘li bilan C ++ uchun qulay va samarali usulni beradi.

Qiymat tomonidan bajarilishining afzalligi tez qiymat uzatish va foydalanish vaqti tejaladi.

Qiymat tomonidan bajarilish usulini asosiy kamchiligi qo‘shimcha saqlash yoki rasmiy parametr uchun zarur bo‘lgan nusxalarni ishlatadi va foydalanuvchini chaqiradi. Bundan tashqari, haqiqiy parametrni saqlash hududga ko‘chiriladi mos rasmiy parametr uchun. Saqlash va nusxa ko‘chirish operatsiyalari kabi ko‘plab elementlar bilan bog‘liq qatorlar katta joy egallaydi.

5.3 Dastur parametri-orqali uzatish usullari

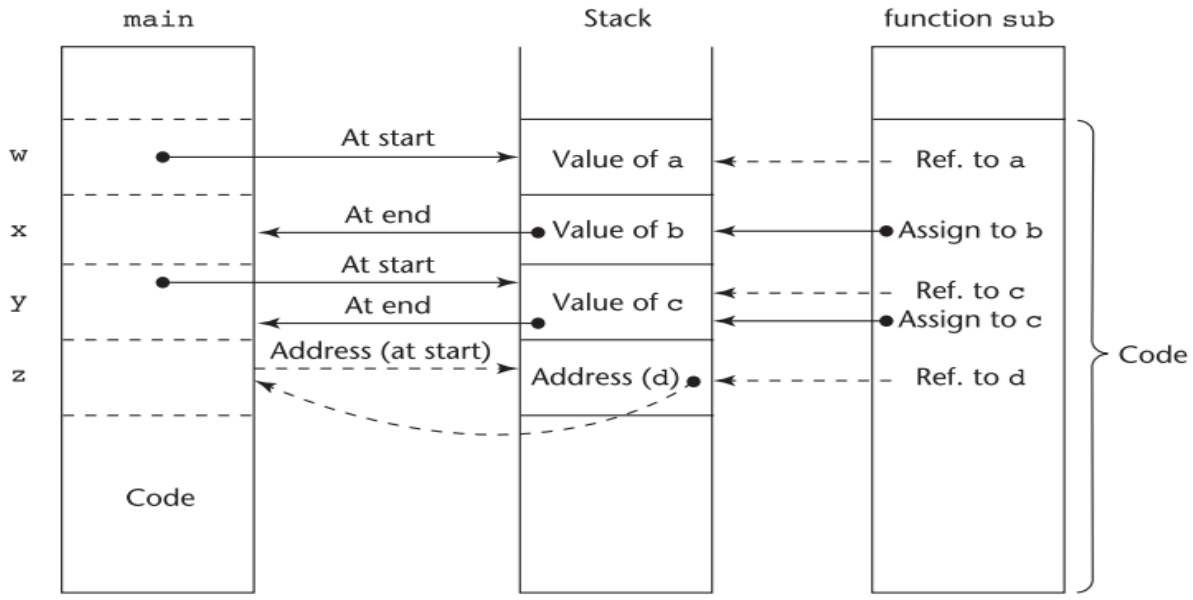
Ko‘plab mavjud tillarda, parametrlar bilan ishlash run-time stekdan foydalanish natijasida sodir bo‘ladi. Bu run-time stak run-time system tomonidan kiritiladi va ta’minlab beriladi. Bu stak dastur bajarilishini boshqaradi. Bu asosan qism dasturlarda jarayonni va parametrlardan foydalanishni nazorat qiladi. Quyida barcha parametrlarda stekdan foydalanilishiga ishonch hosil qilamiz.

Parametrlar qiymati bo‘yicha undan foydalanishda ularning qiymatlari stek sohasiga ko‘chiriladi. Keyin bu soha mavjud parametrga xotira sifatida xizmat qiladi.

Parametrlarni ularni qaytargan natijasi bo‘yicha undan foydalanishda qiymat bo‘yicha foydalanishga qaraganda teskari ish bajariladi. Qaytarilgan qiymat natijalari stakda joylashadi va chaqirilgan qism dasturning boshidan oxirigacha chaqirilgan dastur qismi takror foydalanish mumkin.

Natijani qiymat bo‘yicha uzatish (pass-by-value-result)da parametrlari qurilishiga ko‘ra qiymat bo‘yicha uzatish (pass-by-value) va natija bo‘yicha uzatish (pass-by-result) parametrlaridan olinadi. Bu parametrning stakda joylashuvi chaqirish paytida inisializatsiya qilinadi (ya’ni kiritiladi) va u chaqirilgan quyi dasturda lokal o‘zgaruvchiga o‘xshab foydalaniladi.

Tavsif bo‘yicha uzatish (pass-by-reference) parametrlaridan foydalanish eng osoni hisoblanadi. Shunga qaramasdan mavjud parametr tipi, faqatgina uning manzili stakda joylashadi. Literallar adresi esa stakda qo‘yiladi. Bunday vaziyatda kompilyatsiya qiluvchi (dasturchi) chaqirilgan qism dastur ishini nazorat qilib turish uchun dastur tuzishi kerak. Kod va uning bajarilishidagi natijani xotiradagi manzili stakda joylashadi. Dasturchi tomonidan chaqirilgan dasturning parametrlari (literal yoki ifoda) o‘zgarishidan himoya qila olishi kerak.



Chaqirilgan qism dasturlar parametrlariga stak xotirasidan kirish bilvosita adres orqali amalga oshiriladi. Yuqoridagi rasmdan ko‘rinadiki parametrlar orqali ish bajarishda run-time stakdan foydalaniladi. sub qism dasturda main dan sub(w, x, y, z) qism dasturi orqali chaqiriladi. Bu yerda w qiymat bo‘yicha uzatiladi, x natija bo‘yicha uzatiladi, y natija qiymati bo‘yicha uzatiladi, z tavsif bo‘yicha bilan uzatiladi.

5.3.1 Qism dasturga misol

2 ta a va b sonlar berilgan ularni o‘rnini almashtirish dasturini S dasturlash tilida ko‘rib chiqamiz.

```
void swap1 (a,b: integer)
temp: integer;
begin
temp:=a;
a:=b;
b:=temp;
end;
```

swap1 funksiyasini quyidagi ko‘rinishda chaqirish mumkin:

```
swap1(&c, &d);
```

swap1 funksiyasining amalini quyidagi psevdokod orqali tavsiflash mumkin:

```
a=&c                - birinchi parametr qiymatini jo‘natish
```

```
b=&d                - ikkinchi parametr qiymatini jo‘natish
temp=*a
*a=*b
*b=temp
```

O‘rnatish funksiyasini S tilida shunday modifikatsiyalaymizki, u ko‘rsatkichlar bilan parametrlar kabi ishlasin. Bu esa ssылka bo‘yicha parametrlarni yuborish effektiga erishishga imkon beradi.

```
void swap2 (int*a, int*b)
{
Int temp =*a;
*a=*b;
*b=temp;
}
```

swap2 funksiyasini quyidagi ko‘rinishda chaqirish mumkin:

```
Swap2(&c, &d);
```

swap2 funksiyasining amalini quyidagi psevdokod orqali tavsiflash mumkin:

```
a=&c - birinchi parametr manzilini jo‘natish
b=&d - ikkinchi parametr manzilini jo‘natish
temp=*a
*a=*b
*b=temp
```

Bu holda, to‘xtatish amali samarali bo‘ladi: c va d o‘zgaruvchilarning qiymatlari rostdan ham o‘zaro joy almashadi.

swap2 funksiyasini S++ tilida quyida ko‘rsatilganidek ssылkalarni parametrlar kabi qo‘llab yozish mumkin:

```
Void swap2 (int &a,int&b)
{int temp=a;
a=b;
b=temp;
}
```

Ushbu oddiy to‘xtatish amali Java tilida mumkin emas, chunki u yerda na ko‘rsatkichlar va na S++ tilidagi ssылkalarga muvofiq bo‘lgan ssылkalar mavjud. Java tilida ssылka skalyar kattalikka emas, balki faqat ob’ektgagina yo‘naltirilishi mumkin.

Pascal tilida swap2 funksiyasini quyidagicha yozish mumkin:

```
Procedure swap2 (var a, b:integer)
Temp : integer;
Begin
Temp:=a;
```

```
a:=b;
b:=temp;
end;
```

Aytaylik, swap2 amali Pascal tilida quyidagicha chaqirilsin:

```
swap2 (1, list[1]);
```

Bu holda, uning amallarini quyidagicha yozish mumkin:

```
a=&i - birinchi parametr manzilini jo‘natish
b=&list[i] - ikkinchi parametr manzilini jo‘natish
temp=*a
*a=*b
*b=temp
```

Nomi o‘zgartirilgan *a (i o‘zgaruvchisi qiymatiga teng) ssылkasining qiymati nomi o‘zgartirilgan *b (list[i] qiymatiga teng) ssылkasining qiymati o‘zgargunigacha o‘zgarib bo‘lganiga qaramay, bu to‘xtatishning to‘g‘riligiga hech qanday ta’sir ko‘rsatmaydi, chunki list[i] o‘zgaruvchisi qiymati chaqirish paytida hisoblanadi va bundan keyin i o‘zgaruvchisiga nima bo‘lmasin, o‘zgarmas bo‘lib qoladi.

Parametrlarni qiymat bo‘yicha yuborish semantikasi ssылka bo‘yicha yuborish semantikasi bilan ismlar aralashmasi yuz bermagan holdagina bir xildir. Eslatib o‘taylik, Ada tilida kiritishchiqarish rejimida qiymatlar va natijalar bo‘yicha yuborish skalyar parametrlarni yuborishda qo‘llaniladi. Parametrlarni qiymatlar va natijalar bo‘yicha yuborishni o‘rganib chiqish uchun, keyingi swap3 funksiyasini ko‘rib chiqamiz, bunda ushbu metod qo‘llaniladi. U Ada tiliga yaqin bo‘lgan sintaksis yordamida yozilgan.

```
procedure swap3 (a: in out integer, b : in out integer) is
temp : integer;
begin
temp := a;
a:= b;
b:= temp;
end swap3;
```

Aytaylik, swap3 protsedurasi quyidagicha chaqirilsin:

```
swap3(c,d);
```

swap3 funksiyasi amallarini ushbu chaqiruvda quyidagicha yozish mumkin:

addr_c = &c - birinchi parametr manzilini jo‘natish
addr_d = &d - ikkinchi parametr manzilini jo‘natish
a = *addr_c - birinchi parametr qiymatini jo‘natish
c = *addr_d - ikkinchi parametr qiymatini jo‘natish
temp = a
a = b
c = temp
*addr_c = a - birinchi parametr qiymatini qaytraish
*addr_d = b - ikkinchi parametr manzilini jo‘natish

Shunday qilib, ushbu to‘xtatish quyi dasturi to‘g‘ri ishlayapti. Endi quyidagi chaqiruvni ko‘rib chiqamiz:

swap3(i, list[i]);

Bu holda swap3 protsedurasi amallarini quyidagi ko‘rinishda yozish mumkin:

addr_i = &i birinchi parametr manzilini jo‘natish
addr_listi = &list[i] - ikkinchi parametr manzilini jo‘natish
a = *addr_i - birinchi parametr qiymatini jo‘natish
b = *addr_listi - ikkinchi parametr qiymatini jo‘natish
temp = a
a = b
b = temp
*addr_i = a - birinchi parametr qiymatini qaytarish
*addr_listi = b - ikkinchi parametr qiymatini qaytarish

Bu holda quyi dastur yana to‘g‘ri ishlayapti, chunki parametrlarning qaytariluvchi qiymatlari yoziluvchi katakcha manzillari quyi tizimdan qaytarish vaqtida emas, balki chaqirish vaqtida hisoblanadi. Agar faktli parametrlar manzillari quyi tizimdan qaytarilish vaqtida hisoblanganida edi, natijalar noto‘g‘ri bo‘lgan bo‘lar edi.

Ismlarni aralashtirib yuborishda nima yuz berishini ko‘rib chiqaylik. Bunda parametrlar qiymatlar va natijalar orqali jo‘natiladi. S tiliga yaqin bo‘lgan sintaksis yordamida yozilgan skelet dasturn ko‘rib chiqamiz:

```
int i =3; /* i- dunyoviy o‘zgaruvchi */
void fun (int a, int b) {
```

```
i=b;
}
void main() {
int list[10];
list[i]=5;
fun (i, list[i]);
}
```

fun funksiyasida ssылka bo‘ylab parametrlarni jo‘natish amalga oshiriladi, bunda i va a o‘zgaruvchilari alternativ nomlar bo‘ladi. Agar parametrlar qiymat va natijalar bo‘ylab yuborilsa, i va a o‘zgaruvchilari alternativ nomlar bo‘lmaydi. Bu holda fun funksiya amallari quyidagicha:

```
addr_i = &i - birinchi parametr manzilini jo‘natish
addr_listi = &list[i] - ikkinchi parametr manzilini jo‘natish
a = *addr_i          - birinchi parametr qiymatini jo‘natish
b = *addr_listi              - ikkinchi parametr qiymatini jo‘natish
i = b                - i ni 5 ga tenglash
*addr_i = a          - birinchi parametr qiymatini qaytarib olish
*addt_listi = b              - ikkinchi parametr qiymatini qaytarib olish
```

Bu holda, fun quyi dasturida i global o‘zgaruvchisini qabul qilish natijasida uning qiymati 2dan 3ga o‘zgaradi, orqaga esa 3 qiymatni saqlovchi birinchi rasmiy parametr nusxasi qaytariladi. Bu yerda ssылka bo‘ylab xabar jo‘natish amalga oshirilsa, nusxa qaytarish parametrlarni jo‘natish semantikasi bo‘limi bo‘lmasligini yodda tutish lozim. Shuning uchun i o‘zgaruvchisi 5 ga tengiligicha qoladi. Ikkinchi parametr manzili fun funksiyasi bajarilishidan oldin hisoblangani uchun, global i o‘zgaruvchisidagi o‘zgarishlar uning manziliga ta’sir ko‘rsatmaydi. Ushbu manzil list[i] o‘zgaruvchisining qiymati qaytarilganida chiqishda qo‘llaniladi.

6. Qism dastur parametrlari

Dasturlashda ko‘pincha agar qism dastur nomi boshqa qism dasturga parametr kabi yuborilsa, uni qayta ishlash afzalroq bo‘lgan holatlar ham bo‘ladi. Ba’zida, qism dastur qandaydir matematik funksiyani modellashtirishi lozim bo‘lganda bu hol yuzaga keladi. Masalan, sonli integralini hisoblovchi qism dastur, funksiya grafigi ostidagi shakl yuzini aniqlaydi va ushbu funksiyani turli nuqtalarda hisoblashi lozim. Bunday qism dastur har qanday berilgan funksiya uchun qo‘llanilishi lozim. Qism dasturni har bir funksiya uchun yozib o‘tirish shart

bo‘lmasligi kerak. Buning natijasida, dasturdagi matematik funksiya integralini hisoblovchi funksiya nomi integratsiyalovchi qism dasturga parametr kabi yuboriladi.

Bu g‘oya oddiyligi va tabiiyligiga qaramay, uning to‘liq ishlashi muammo tug‘dirishi mumkin. Agar faqat qism dastur kodini yuborish kerak bo‘lganida edi, alohida ko‘rsatkichni yuborishning o‘zi yetarli bo‘lar edi. Ammo, bir qator murakkab holatlar yuzaga keladi.

Birinchidan, parametr kabi yuboriluvchi qism dasturlarni chaqirishda parametrlar tiplarini tekshirish muammosi mavjud. Pascal tilining boshlang‘ich tavsifnomasida qism dasturlarni ularning parametrlari tiplarini qo‘shmay, ularni parametr kabi yuborish imkoniyati berilar edi. Agar mustaqil kompilyatsiya mavjud bo‘lsa, kompilyator parametrlar sonining to‘g‘riligini tekshirishga ham imkon bermaydi. Mustaqil kompilyatsiya mavjud bo‘lmasa, parametrlar muvofiqligini tekshirishning imkoni bor, ammo bu juda ko‘p qiyinchiliklar tug‘diradi va odatda, buning umuman iloji yo‘q. FORTRAN 77 tilida ham ushbu muammo mavjud, ammo FORTRAN 77 tilidagi tiplar muvofiqligini tekshirish hech qachon bajarilmaganligi uchun, bu unchalik ham muammo hisoblanmaydi.

Agar qism dastur nomi ALGOL 68 tilida yoki Pascal tilining oxirgi versiyalarida parametr kabi yuborilsa, rasmiy parametrlar tiplari qism dasturlardan qabul qilinuvchi rasmiy parametrlar ro‘yxatiga qo‘shiladi, shuning uchun, real chaqiriqda qism dasturga yuboriluvchi parametrlar tiplari muvofiqligini statistik holda bajarish mumkin. Masalan, quyidagi kodni Pascal tilida ko‘rib chiqaylik:

```
procedure integrate (function fun (x:real):real;
lowerbd, upperbd:real;
var result:real);
…
var funval; real;
begin
…
funval := fun (lowerbd);
…
end;
```

integrate protsedurasidagi fun funksiyasini chaqirishda faktli parametr statik tarzda integrate protsedurasining rasmiy parametrlari ro‘yxatidan fun funksiyasining rasmiy parametri tipi bilan muvofiqligini tekshirish mumkin.

S va C++ tillarida funksiyalar parametrlar kabi yuborila olmaydi, ammo funksiya ko‘rsatkichlari yuborilishi mumkin. Funksiya ko‘rsatkichi tipi uning protokoli hisoblanadi. Protokol barcha parametrlar tiplariga ega bo‘lgani uchun, bunday parametrlar tiplar muvofiqligiga tekshirilishi mumkin.

Modula-2 tilida protseduralar tiplari o‘zgaruvchilar kabi, ularni yuborish uchun qo‘llaniladi. Ushbu metod jo‘natilayotgan qism dasturlar parametrlari tiplari muvofiqligini tekshirishga imkon beradi, chunki parametrlar tiplari protsedura tipining bir bo‘lagi hisoblanadi. FORTRAN 90 tilida qism dasturlar parametrlari tiplari muvofiqligini tekshirishni tavsiflovchi mexanizm mavjud bo‘lib, ularning o‘zi parametrlar kabi yuboriladi. Ada tilida qism dasturlarni parametr kabi qo‘llab bo‘lmaydi. Buning o‘rniga, parametrlar kabi yuboriluvchi qism dasturlar ta’minlovchi funksional imkoniyatlar Ada tilida o‘rnatilgan funksiyalar yordamida egallaniladi va bu to‘liq tarzda 8-qismida ko‘rib chiqilgan.

Qism dasturlar nomlarini qo‘llashning yanada qiziq tarafi yuborilayotgan qism dastur ssыlkalarning to‘g‘ri muhiti haqidagi muammolar bilan bog‘liq. Buning uchta imkoniyati bor:

1. Chaqirish operatori muhiti, bu yuborilayotgan qism dasturni aktivlashtiradi (soyali bog‘lanish – shallow binding).
2. Yuborilayotgan qism dasturni aniqlash muhiti (chuqur bog‘lanish – deep binding).
3. Qism dasturni faktli parametr kabi yuboruvchi chaqirish operatori muhiti (alohida bog‘lanish − ad hoc binding).

Quyidagi keltirilgan dastur ushbu imkoniyatlarni amalga oshiradi. SUB 3 protsedurasi SUB 4 protsedurasini chaqira oladi deylik.

```
procedure SUB1;
var x : integer;
procedure SUB2;
begin
write (‘x=’, x);
end; {sub2}
procedure SUB3;
varx : integer;
begin x : integer;
x := 3;
SUB4(SUB2)
end; {SUB3}
procedure SUB4(SUBX);
varx : integer;
 begin
x := 4;
SUBX;
end; {SUB4}
begin {SUB1}
```

```
x := 1;
SUB3
end; {SUB1}
```

SUB 3 qism dasturini SUB 4 orqali chaqirishda nima kelib chiqishini ko‘rib chiqamiz. Soyali bog‘lanishda bunday bajarishning ssылkalar muhiti SUB 4 qism dasturi ssылkalar muhiti bilan mos keladi, shuning uchun, x o‘zgaruvchi ssылkasi SUB 2 muhitidagi mahalliy o‘zgaruvchi bilan muvofiq bo‘ladi.

7. O‘rnatilgan qism dasturlar

Ada tilida aralashgan iboralar qo‘llanilishi yo‘lga qo‘yilmagani uchun, funksiya konteksti funksiya tomonidan qaytariluvchi qiymatlar tipini aniqlashi mumkin. S++ va Java tilida aralashgan iboralar qo‘llanilishi yo‘lga qo‘yilgan, shuning uchun, qaytarilayotgan qiymat tipi hech qanday ahamiyatga ega emas.

Foydalanuvchilar ham qism dasturlarning bir qancha versiyalarini Ada, Java va C++ tillarida bir xil nom bilan yaratishlari mumkin. Ushbu qism dasturlar bir xil jarayonni bajarishi shart emasligiga qaramay, ular odatda shuni bajaradilar. Masalan, aniq bir dastur jamlashning ikki protsedurasiga ehtiyoj sezadi, biri – butun sonli massivlar uchun, yana biri – o‘zgaruvchan nuqtali sonlar massivi uchun zarur. Ushbu protseduralarning ikkovi ham SORT deb atalishi mumkin, chunki ularning parametrlari tiplari turli. Quyidagi Ada tilida keltirilgan dastur skeletida SORT nomli ikki protsedura mavjud:

```
procedure MAIN is
type FLOAT_VECTOR is array (INTEGER range <>) of FLOAT;
type INT_VEXTOR is array (INTEGER range <>) of INTEGER;
…
procedure SORT (FLOAT_LIST : in out FLOAT VECTOR;
                LOWER_BOUND : in INTEGER;
                UPPER_BOUND : in INTEGER) is
…
end SORT;
 procedure SORT (INT_LIST : in out INT_VECTOR;
                LOWER_BOUND : in INTEGER;
                UPPER_BOUND : in INTEGER) is
…
end SORT;
…
end MAIN;
```

O‘rnatilgan qism dasturlar turli xil chaqiruvlarga olib kelishi mumkin. Misol qilib, S++ tilidagi quyidagi kodni ko‘rib chiqamiz:

```
void fun (float b = 0.0);
void fun();
…
fun ();
```

Ushbu chaqiruv o‘zgaruvchan va kompilyatsiyada xatolik keltirib chiqaradi.

8. O‘rnatilgan qism dasturlar

Kodni qayta qo‘llash dasturiy ta’minotning ishlab chiqaruvchilanligini sezilarli tarzda oshiradi. Masalan, bu holda o‘z elementlari tiplari bilangina farq qiluvchi to‘rtta massivni to‘g‘rilash uchun dasturchiga sozlashning turli xil to‘rtta qism dasturlarni yozib o‘tirishga zaruriyat tug‘ilmaydi.

O‘rnatilgan (generic) yoki polimorf (polymorphic subprogram) qism dastur turli chaqiruvlarda turli tipli parametrlarni qabul qiladi. O‘rnatilgan qism dasturlar o‘zi bilan polimorfizmning turliligini taqdim etib, bu alohida polimorfizm deb ataladi. APL tili funksiyalari polimorfizmning yanada to‘liqroq ko‘rinishini ta’minlaydi. APL tilida dinamik bog‘lanish qo‘llangani uchun, parametrlar tiplarini kiritishga hojat yo‘q, ular bir-biri bilan shunchaki mos faktli parametrlar bilan bog‘lanadi.

Parametrli polimorfizm o‘rnatilgan parametrli qism dasturlar orqali ta’minlanadi. Ular qism dasturlar parametrlari tiplarini tavsiflash uchun qo‘llaniladi. Ada tilida ham, C++ tilida ham kompilyatsiya jarayonida parametrli polimorfizm turliligi qo‘llaniladi.

8.1. Ada tilida o‘rnatilgan qism dasturlar

Ada tilida parametrli polimorfizm konstruksiyalar orqali ta’minlanib, ular dastur birliklarining turli versiyalarini yaratishga imkon beradi. Foydalanuvchi dasturi so‘roviga ko‘ra qism dasturlarning turli versiyalari nusxalarini yaratadi yoki tuzatadi. Qism dasturning barcha versiyalari bir xil nomga ega bo‘lgani uchun turli chaqiruvlardagi bir qism dastur turli tipdagi ma’lumotlarni qayta ishlashi mumkin degan tasavvur uyg‘onadi. Turli xildagi dastur birliklari o‘rnatiluvchan bo‘lgani uchun, ularni ba’zida o‘rnatiluvchi komponentlar deb ataladi.

Ushbu mexanizmdan chaqiriluvchi qism dasturga turli holatlarda o‘rnatilgan qism dasturning turli nusxalarini chaqirish imkoniyatini berishi uchun ham

foydalanish mumkin. Bunday usul qism dasturlar funksionalligini ta’minlashda zarur bo‘lib, ular parametrlar kabi yuboriladi.

Quyida keltirilgan misol uchta o‘rnatilgan parametrga ega protsedurani tasavvur qilishga imkon beradi. Bu esa qism dasturga parametr sifatida o‘rnatilgan massivni qabul qilish imkoniyatini beradi. Protsedura almashinuv sortirovkasi algoritmini amalga oshiradi va har qanday massiv bilan ishlashi mumkin. Ushbu massiv elementlari ro‘yxatli tip indeksi diapazonini qo‘llagan holda raqamli tipga ega:

```
generic
type INDEX_TYPE is (<>);
type ELEMENT_TYPE is private;
type VECTOR is array (INTEGER_TYPE range <>) of ELEMENT_TYPE;
procedure GENERIC_SORT (LIST : in out VECTOR);
procedure GENERIC_SORT (LIST : in out VECTOR) is TEMP : ELEMENT_TYPE;
begin
for TOP in LIST’FIRTS..INDEX_TYPE’PRED(LIST’LAST) loop
for BOTTOM in INDEX_TYPE’SUCC(TOP)..LIST’LAST loop
if LIST (TOP) > LIST (BOTTOM) then
TEMP := LIST(TOP);
LIST := LIST (BOTTOM);
LIST (BOTTOM) := TEMP;
end if;
end loop; -- for BOTTOM …
```

Ushbu o‘rnatilgan protseduraning ba’zi bo‘laklari g‘alati ko‘rinishi mumkin, bu siz Ada tili bilan tanish bo‘lmasangizgina yuzaga keladi. Ammo, ushbu holatda sintaksisning barcha detallarini tushunish shart emas. Massiv tiplari va uning elementlari ushbu protseduraning ikkita o‘rnatilgan parametri orqali ishga tushiriladi, bunda indeks o‘zgarishining har qanday diapazoni va tipi qo‘llanilishi mumkin.

O‘rnatilgan sortirovka, protsedura shablonidan boshqa narsa emas. Kompilyator ushbu protsedura uchun hech qanday kodni generatsiyalamaydi. Va bu qandaydir aniq tip uchun protsedura nusxasi yaratilmagunigacha dastur ishlashiga halal bermaydi. Protsedura nusxasi operator yordamida yaratilib, u quyidagi ko‘rinishda bo‘lishi mumkin:

```
procedure INTEGER_SORT is new GENERIC_SORT (
                                    INDEX_TYPE => INTEGER;
                                    ELEMENT_TYPE => INTEGER;
                                    VECTOR => INT_ARRAY);
```

Ushbu operatorga javob tarzida kompilyator INTEGER_SORT nomi bilan GENERIC_SORT protsedurasi versiyasini yaratib, u INTEGER tipi indeksili INTEGER tipi elementlariga ega bo‘lgan INT_ARRAY tipi massivini sortirovkalaydi.

GENERIC_SORT protsedurasida > operatori sortirovkalanayotgan massiv elementlari uchun aniqlangan. GENERIC_SORT protsedurasining universalligini oshirish mumkin. Bunda unga o‘rnatilgan parametrli taqqoslash funksiyasi o‘rnatiladi.

Ada tilida qism dasturlarni boshqa qism dasturlarning parametrlari kabi yuborish yo‘lga qo‘yilmaydi. Ushbu imkoniyatga ega bo‘lish uchun, Ada tilida o‘rnatilgan rasmiy qism dasturlar qo‘llaniladi. Pascal dasturlash tilida qism dasturlar parametrlar kabi yuborilishi mumkin, shuning uchun aniq chaqiruvda yuborilayotgan qism dasturdan qism dastur natijasini chiqarish uchun foydalanish mumkin. Ada tilida xuddi shunday natijaga turli mavjud qism dasturlar bilan foydalanuvchiga o‘rnatilgan qism dasturning cheklangan sonda nusxa yaratish imkoniyatini berish orqali hal qilinadi. Masalan, 8.6. bo‘limda yozilgan integrate protsedurasini Ada tilida quyidagi ko‘rinishda yozish mumkin:

```
generic
with function FUN (X: FLOAT) return FLOAT;
procedure INTEGRATE (LOWERBD : in FLOAT;
                     UPPERBD : in FLOAT;
                     RESULT : out FLOAT) is
                     FUNVAL : FLOAT;
                     begin
                     …
                     FUNVAL:= FUN (LOWERBD);
                     …
                     end;
```

Foydalanuvchi tomonidan FUN1 funksiyasini integratsiyalash uchun ushbu protsedura nusxasini yaratishda quyidagi operatorni qo‘llash mumkin:

```
procedure INTEGRATE_FUN is new INTEGRATE (FUN=>FUN1);
```

Endi INTEGER_FUN1 protsedurasi FUN1 funksiyasini integratsiyalashga qaratilgan.

8.2. C++ tilida o‘rnatilgan qism dasturlar

C++ tilida o‘rnatilgan funksiyalar shablonli funksiyalar deb nomlanadi. Shablonli funksiyalar quyidagi ko‘rinishga ega bo‘ladi:

template <class parametrlar>

Sinf bo‘lgan parametrlarga ega bo‘lishi mumkin bo‘lgan aniq funksiyalar

Shablonli funksiya sinf bo‘lgan hech bo‘lmasa bitta parametrga ega bo‘lishi lozim. Har qanday bunday parametr quyidagi ko‘rinishda bo‘ladi:

class identifikator

Masalan, quyidagi shablonli funksiyani ko‘rib chiqamiz

```
template <class Type>
Type max (Type first, Type second)
{ return first>second ? first:second;
}
```

Bu yerda Type – parametr bo‘lib, funksiya ishlovchi ma’lumotlar tipini ko‘rsatadi. Bunday shablonli funksiya nusxalarini ma’lumotlarning har qanday tipi bilan ishlash uchun yaratish mumikn. Masalan, int tipi uchun protsedura nusxasi quyidagicha bo‘lishi mumkin:

```
int max (int first, int second) {
return first> second ? first: second;
```

Ushbu amallarni makroaniqlash (makros) ko‘rinishida tavsiflash mumkin, ammo bu holda u noto‘g‘ri ishlashi mumkin. Makros quyidagi ko‘rinishda bo‘lsin deylik:

```
define max (a,b) ((a) > (b) ? (a) : (b)
```

Ushbu makros o‘rnatiluvchan hisoblanadi, ya’ni u har qanday sonli tipdagi ma’lumotlar bilan ishlaydi. Ammo, u har doim ham to‘g‘ri ishlamaydi, asosan, nosozliklarga ega parametrlar orqali chaqirilgan hollarda, masalan:

```
max (x++, y)
```

Bu quyidagi natijaga olib keladi:

((x++)> (y) ? (x++) : (y))

x o‘zgaruvchisi qiymatlari u o‘zgaruvchisi qiymatidan katta bo‘lgan har safar, x o‘zgaruvchisi qiymati ikki marta oshadi.

C++ tilida shablonli funksiyalari nusxalari o‘z holicha yaratiladi. Masalan, yuqorida aniqlangan shablonli funksiyalar nusxalarini quyidagi kod fragmenti yordamida ikki marta yaratish mumkin: int tipli va char tipili parametrlar uchun, albatta:

```
int a, b, c;
char d, e, f;
…
c=max (a, b);
f=max (d, e);
```

Quyida C++ tilidagi o‘rnatilgan qism dastur versiyasi keltirilgan bo‘lib, u 8.8.1 bo‘limda ko‘rib chiqilgan. U oldingi versiyasidan salgina farq qiladi, chunki C++ tilidagi massivlar indeksi butun bo‘lishi lozim.

```
template <class Type>
void generic_sort (Type list[], int len {
int top, bottom;
Type temp;
for (top = 0; top <len-2;top++)
for (bottom=top+1; bottom<len-1; bottom)
if (list[top]>list[bottom]) {
temp = list[top];
list[top]=list[bottom];
list[bottom]=temp;
} //* for siklining oxiri (bottom = …
} //** o‘rnatilgan sortirovka qilish funksiyasining oxiri
```

Quyida ushbu shablonli funksiyasining nusxasini yaratishga misol keltirilgan:

```
float flt_lit[100];
…
generic_sort(flt_list, 100);
```

Ada tilidagi o‘rnatilgan qism dasturlar va C++ tilidagi shablonli funksiyalar qism dasturlarga turlicha qaraydi, bunda rasmiy parametrlar chaqirilganda faktli

parametrlar tiplari bilan dinamik holda bog‘lanadi. Bu holda, kodning birgina nusxasi kerak bo‘ladi.

Smalltalk, Java, Ada 95 va C++ tillarida chaqiruvlar metodning kerakli versiyasi bilan dinamik bog‘lanadi, bunda u faktli parametrlar tiplariga muvofiq bo‘ladi. Ushbu mavzu 11 bobda o‘rganib chiqilgan.

9. Alohida va mustaqil kompilyatsiya

Dasturning o‘zini kompilyatsiyalamagan holda dastur bo‘laklarini kompilyatsiyalash imkoniyati dasturiy ta’minotning yirik tizimlariga xos. Natijada, bunday ilovalar uchun ishlab chiqilgan tillar bunday kompilyatsiya turini kiritishi lozim. Bunday imkoniyatga ega bo‘lib, dasturchi tizimni ishlab chiqarishda yoki ekspluatatsiyalashda o‘zgarishlarga uchragan modullarnigina qayta kompilyatsiyalashi mumkin.

Mustaqil kompilyatsiyaning asosiy xususiyati shundaki, alohida kompilyatsiyalanuvchi modullar orasidagi interfeyslar tiplar muvofiqligini tekshirish zaruriyatini tug‘dirmaydi. FORTRAN 77 tilidagi qism dasturlar interfeysi o‘zi bilan parametrlar ro‘yxatini taqdim etadi. Qism dastur alohida kompilyatsiyalanganida, uning parametrlari tiplari kompilyatsiyalanayotgan kod yoki kutubxonada saqlanmaydi. Natijada, ushbu qism dasturni chaqiruvchi boshqa dasturni kompilyatsiyalashda chaqiruvdagi faktli parametrlar tiplari rasmiy parametrlar tiplari bilan muvofiqligi tekshirilmaydi.

FORTRAN 77 tili uchun bu yangilik emas. Qandaydir qism dasturni chaqiruvchi dastur va ushbu qism dastur bir xil faylda kompilyatsiyalansa ham, ular mustaqil kompilyatsiyalanadi. Shunday qilib, FORTRAN 77 tilidagi dasturiy modullar orasidagi interfeys tiplar muvofiqligini hech qachon talab qilmaydi.

Ba’zi tillar na alohida va na mustaqil kompilyatsiyaga yo‘l qo‘yadi. Bu kompilyatsiyalashning yagona usuli butun bir dasturni kompilyatsiyalash deganidir. Bunday tillarga misol qilib FORTRAN II tili va Pascal tilining boshlang‘ich versiyalarini keltirish mumkin. Na alohida va na mustaqil kompilyatsiyaga ruxsat beruvchi tilning imkoniyatlarini cheklaydi. FORTRAN va Pascal tillarining oxirgi versiyalarida bu muammo hal qilindi.

10. Funksiyalarni ishlab chiqish savollari

Funksiyalarga ishlab chiqishning quyidagi savollari xos.

- Nosozliklarga yo‘l qo‘yiladimi?
- Qanday tipli qiymatlar funksiyalar tomonidan qaytarilishi mumkin?

10.1. Funksiyalar nosozliklari

Ifodalarda chaqiriluvchi funksiyalarga xos bo‘lgan nosozliklar natijasida 4 bobda yozilganidek, funksiyadagi parametrlar kiritish rejimida yuborilishi lozim. Ba’zi dasturlash tillari ushbu talab bajarilishini talab qiladi; masalan, Ada tilidagi funksiyalar faqat kiritish rejimida yuboriluvchi rasmiy parametrlarga ega bo‘lishi mumkin. Bu parametrlar yuborilishida yoki parametrlar hamda dunyoviy o‘zgaruvchilar nomlari bir xilligi vujudga kelganda nosozliklar kelib chiqishidan saqlaydi. Ammo, boshqa tillarda, funksiyalar yoki qiymat bo‘yicha, yoki ssылka bo‘yicha yuboriluvchi parametrlarga ega bo‘lishi mumkin, buning natijasida, nosozliklar hamda nomlar bir xilligi yuzaga keladi.

10.2. Qaytariluvchi qiymatlar tiplari

Dasturlashning boshqaruvchi tillarining ko‘pchiligida funksiyalar tomonidan qaytarilishi mumkin bo‘lgan o‘zgaruvchilar tiplari cheklangan. FORTRAN 77, Pascal va Modula-2 tillarida funksiyalar faqat strukturalashmagan tiplarni qaytarishi mumkin. S tilida funksiyalar massiv va funksiyalardan tashqari har qanday tipdagi o‘zgaruvchilarni qaytarishi mumkin. C++ tili S tiliga o‘xshash, ammo unda funksiyalar foydalanuvchi tomonidan aniqlangan o‘zgaruvchilar yoki sinflar ob’ektini ham qaytarishi mumkin. Ada tili – zamonaviy boshqaruvchi tillar orasida funksiyalar har qanday tipli o‘zgaruvchilarni qaytara oladigan yagona til hisoblanadi.

11. Mahalliy bo‘lmagan muhitlarga kirish

Odatda, qism dasturlar o‘rtasidagi aloqa faqat parametrlar orqali o‘rnatilishiga qaramay, ko‘pchilik tillarda o‘zgaruvchilar bilan bog‘lanishning yana bir turi mavjd bo‘lib, u tashqi muhit orqali amalga oshiriladi.

Mahalliy bo‘lmagan o‘zgaruvchili qism dasturlar – bu qism dasturdako‘rinadigan o‘zgaruvchilar bo‘lib, lekin mahalliy e’lon qilinmagan bo‘ladi. Dunyoviy o‘zgaruvchilar – bu dasturning barcha modullarida ko‘rinuvchan bo‘lgan o‘zgaruvilar hisoblanadi. 4 bobda nomahalliy muhitlarga kirishning ikkita metodlari o‘rganib chiqilgan: statik to‘plam va dinamik to‘plam. Ushbu ishlab chiqarishdagi muammolar quyida ko‘rib chiqilgan.

Ma’lumotlarning statik to‘plamiagi asosiy muammo quyidagicha: dastur strukturasi mahalliy va dunyoviy o‘zgaruvchiarning kiritiluvchanligi bilan aniqlanadi. Bu muammolar to‘liq ravishda 4 bobda ko‘rib chiqilgan.

Ma’lumotlarning dinamik to‘plami ikki turdagi muammolarni keltirib chiqaradi. Birinchidan, qism dastur boshlanishidan to tugalangunigacha ketadigan vaqt davrida mahalliy o‘zgaruvchilar bajarilayotgan har qanday qism dastur uchun ko‘rinuvchan bo‘ladi, bunda o‘z ichidagi ma’lumotlar bo‘yicha ushbu dasturlar qanchalik mosligiga ahamiyat berilmaydi. Bunday kirishlardan mahalliy

o‘zgaruvchilarni himoyalash mumkin emas. Qism dasturlar chaqirilayotgan modul muhitiga bog‘lif bo‘lmagan holda bajarildi. Buning natijasida, ma’lumotlarning dinamik to‘plami ma’lumotlarning statistik to‘plamiga qaraganda ishonchsizroq dasturlar yaratadi.

Dinamik to‘plam bilan bog‘liq bo‘lgan ikkinchi muammo – bu nomahalliy o‘zgaruvchilarga o‘rnatilgan ssьlkalar tiplari muvofiqligini statistik tahlil qilish imkoniyatiniing yo‘qligidir, chunki bunday o‘zgarvchining taqdimotini statistik holda topishning hech ham iloji yo‘q.

Nomaalliy o‘zgarvchilarga kirishni amalga oshirish 9 bobda to‘liq ko‘rib chiqiladi.

11.1. FORTRAN dasturlash tilining COMMON bloklari

FORTRAN dasturlash tilida mahalliy o‘zgaruvchilar saqlanuvchi bloklarga kirish COMMONoperatori orqali amalga oshiriladi. Dasturda bunday bloklardan bir qancha bo‘lishi mumkin, va ulardan bittasidan tashqari qolgan hammasi nomga ega bo‘lishi lozim. (bitta nomlanmagan COMMON bloki ko‘pincha bo‘sh COMMON bloki deb nomlanadi va nomlangan COMMON-bloklardan farq qiluvchi xossalarga ega). COMMON blokiga kirishga ega bo‘lish yoki yaratish talab qilinuvchi har qanday qism dastur COMMON operatoriga ega va ushbu operator bu blokni nomlaydi hamda blokdagi ma’lumotlarga kirishdagi o‘zgaruvchilar ro‘yxatini beradi.

Dasturda cheksiz qism dasturlar yaratilishi mumkin bo‘lib, ular bir xil blokka yo‘naltirilgan COMMON operatoriga ega bo‘lishi mumkin. Ushbu blok ko‘rsatilgan har bir qism dastur o‘zining o‘zgaruvchilari ro‘yxatini tuzishi mumkin va bu o‘zgaruvchilar soni hamdatiplariga bog‘liq bo‘ladi.

Masalan, bir dastur ramkasida bir xil blokni turli qism dasturlarda turlicha e’lon qilish mumkin. Birinchi qism dasturda quyidagi e’lon mavjud bo‘lsin:

```
REAL A (100)
INTEGER B (250)
COMMON / BLOCK1 / A, B
```

Boshqa qism dasturda e’lon boshqacha bo‘lishi mumkin:

```
REAL C (50), D (100)
INTEGER E (200)
COMMON / BLOCK 1 / C, D, E
```

BLOCK1 identifikatori to‘g‘ri chiziqlar bilan ajratilgan bo‘lib, blok nomini taqdim etadi. BLOCK1 blokidagi o‘zgaruvchilarga ikki xil qarash 8.3 rasmda

ko‘rsatilgan. Bu yerda, butun sonli va haqiqiy o‘zgaruvchilar bir xil hajmdagi xotirani egallaydi deb taxmin qilinadi.

Agar BLOCK1 bloki birgalikda qo‘llaniluvchi xotira bo‘lsagina, bu hol tushunarli bo‘lishi mumkin. Birgalikda qo‘llaniluvchi xotira – bu o‘zgaruvchilar ro‘yxati o‘zgarishiga ruxsat beradi. COMMON operatoridagi o‘zgaruvchilar ro‘yxatidagi oddiy bir o‘zgarish turli tipdagi o‘zgaruvchilar birgalikda xotiraning bir xil qismini egallashi aniqlash mumkin bo‘lgan xatolikka olib keladi.

FORTRAN 90 dasturlash tili tavsifi COMMON operatori “nomuvofiq xossa” ekanligini ko‘rsatadi. Bu ushbu operator FORTRAN tilining keyingi versiyasigagina qo‘shilishi mumkinligini bildiradi. Buning natijasida esa, ushbu tildan chiqarib tashlanadi. EQUIVALENCE operatori FORTRAN 90 tilining nomuvofiq xossalaridan biri hisoblanadi.

11.2. Tashqi e’lonlar va modullar

Modula-2 va Ada dasturlash tillari statik obzordan dastur modullari bilan ma’lumotlarni birgalikda qo‘llash vositasi kabi foydalanadi. Ushbu tillarning ikkalasida ham ma’lumotlarni birgalikda ishlatishning alternativ metodi ham ko‘rib chiqilgan. Dastur modullarida tashqi modullarni ko‘rsatishga imkon beriladi, bularga kirish talab qilinadi. Ushbu metod yordamida har bir modulda boshqa modullarni aniq ko‘rsatish mumkin bo‘lib, ularga albatta kirish huquqi lozim. Modula-2 tilida ushbu huquq faqat ko‘rsatilgan protseduralar, o‘zgaruvchilar va berilgan tashqi moduldagi ma’lumotlar tiplarigagina ochilishi mumkin. Ada dasturlash tilida dasturchi tashqi modulning faqat nomini kiritishi mumkin, keyin esa barcha tiplari, o‘zgaruvchilari va protseduralariga kirish huquqini beradi. Ushbu metod Modula-2 tilida ham qo‘llanilishi mumkin. Kirish talab qilinadigan barcha vositalarni ko‘rsatish, ko‘rinib turganidek, juda qiyin, chunki kerakli tiplar, o‘zgaruvchilar va protseduralar ro‘yxati juda uzun bo‘lishi mumkin. Ammo, ushbu metod modulni nomining o‘zinigina ko‘rsatishga qaraganda xavfsizroq.

FORTRAN 90 dasturlash tili ham nomahalliy ma’lumotlarni birgalikda tanlov asosida qo‘llashni o‘z ichiga jamlaydi, ushbu ma’lumotlar tiplar muvofiqligini tekshirishni amalga oshiradi.

Ushbu tilga xos xususiyatlar ma’lumotlar abstraksiyasi bilan birga 10 bobda to‘liq o‘rganib chiqilgan.

C++ hamda Java kabi ob’ektga yo‘naltirilgan dasturlash tillarida, ma’lumotlar to‘plamini inkapsulyatsiyalash uchun sinflardan foydalanish mumkin bo‘lib, ushbu sinflar boshqa sinflar bilan o‘zaro uzviy bog‘liq holda qo‘llaniladi. Sinflar 11 bobda to‘liq o‘rganib chiqilgan.

S tilida kiritilgan qism dasturlar mavjud emas, shuning uchun, qism dastur ob’ektlarining faqat birgina bosqichi mavjud. Dunyoviy o‘zgarishlarni yaratish

mumkin, bunda oldindan aniqlangan funksiyalarda ular e’lon qilinadi. Funksiyada tashqi o‘zgaruvchi kabi e’lon qilingan o‘zgaruvchiga kirish external operatori orqali amalga oshiriladi. Boshlang‘ich fayldagi dunyoviy o‘zgaruvchilar e’lon qilinganidan so‘ng aniqlanuvchi barcha funksiyalar tashqi funksiya kabi e’lon qilinmasdan o‘zgaruvchilarga kira oladi. Ushbu metod juda qulay, ammo kirishga bo‘lgan huquqni keragidan ortiq miqdorda beradi.

12 Foydalanuvchi orqali aniqlanuvchi o‘rnatilgan operatorlar

Ada va C++ dasturlash tillarida foydalanuvchi operatorlarni o‘rnatishi mumkin. Bunga misol qilib, Ada tilidagi funksiyani ko‘rib chiqamiz, bunda u ikkita vektor skalyar ko‘paytmasini hisoblash uchun (*) ko‘paytirish operatorini o‘rnatadi. Ikkita vektorning skalyar ko‘paytmasi deganda, ushbu vektorlarning mos elementlarini juftlab ko‘paytirish tushuniladi. Aytaylik, VECTOR_TYPE o‘zgaruvchisi INTEGER tipidagi elementlarni saqlovchi massiv kabi aniqlangan bo‘lsin.

```
function “*” (A, B: in VECTOR_TYPE) return INTEGER is
SUM: INTEGER:=0;
begin
for INDEX in A’range loop
SUM := SUM+A (INDEX) * B (INDEX);
end loop; - for Index…
return SUM;
end “*”;
```

Ushbu funksiya aniqlanishida ko‘rsatilganidek, VECTOR_TYPE tipli operandlar orasida ko‘paytirish (*) belgisi qo‘yilishi bilanoq skalyar ko‘paytma hisoblanadi. Yulduzcha bundan keyin ham hohlagancha o‘rnatilishi mumkin, chunki funksiya aniqanishi oynali protokolga ega.

Yuqorida ko‘rsatilgan skalyar ko‘paytmani hisoblash funksiyasi C++ tilida ham yozilishi mumkin. Bunday funksiya prototipi quyidagi ko‘rinishda bo‘ladi:

```
int operator * (const vector &a, const vector &b, int len);
```

Albatta, savol yuzaga keladi: operatorni qancha marta o‘rnatish mumkin va uni juda ko‘p o‘rnatib yuborish yomon emasmi? Javob mana bunday: operatorni chekli miqdorda o‘rnatish mumkin, bu − dasturchining o‘ziga bog‘liq. Operatorning juda ko‘p o‘rnatilib yuborilishi sababli dasturning ishonchliligi yo‘qoladi. Ko‘p hollarda, dasturda boshqa turdagi operandlarga qo‘llaniluvchi operatorlar mavjud bo‘lsa, u o‘qiluvchan hisoblanadi. Hatto skalyar ko‘paytmada ham oddiy olish operatori tushunarli emas:

C:=A*B;

A, B va C o‘zgaruvchilar skalyar kattaliklar deb xato tushunchaga ega bo‘lish mumkin.

Bundan tashqari, dasturiy ta’minot tizimlari ishlab chiqaruvchilarning turli guruhlari tomonidan yaratiluvchi modullardan yaratiladi. Agar turli guruhlar bir xil operatorni turli usullar bilan o‘rnatgan bo‘lsalar, modullar yagona tizimga birlashtirilishidan oldin ushbu nomutanosibliklarni yo‘qotish lozim.

13 O‘zaro bog‘liq dasturlar

O‘zaro bog‘liq dasturlar (coroutines) – bu qism dasturlarning alohida xilma-xilligidir. Odatda chaqiruvchi va chaqiriluvchi qism dasturlar orasidagi “boshliq-ishchi” munosabatidan farqli ravishda, chaqiruvchi va chaqiriluvchi o‘zaro bog‘liq dasturlar o‘zaro tengdir. O‘zaro bog‘liq dasturlarni boshqarish mexanizmi simmetrik modullarni boshqarish modeli deb ham ataladi.

Simmetrik modullarni boshqarish konsepsiyalarining asosiy kelib chiqish tarixini aniqlash qiyin. O‘zaro bog‘liq dasturlar eng oldingi maqolalarda sitaktik tahlilga qo‘llanilgan (Conway, 1963). O‘zaro bog‘liq dasturlarni qo‘llab-quvvatlash vositalariga ega bo‘lgan yuqori bosqichdagi birinchi dasturlash tili SIMULA 67 dasturlash tilidir. Eslatib o‘taylik, SIMULA ning boshlang‘ich maqsadi tizimlarni modellashtirish bo‘lib, ushbu tizimlar mustaqil jara1nlarni modellashtirishni talab qilgan. SIMULA 67 tilida o‘zaro bog‘liq dasturlarning yaratilishiga shu sababchi bo‘lgan. O‘zaro bog‘liq dasturlarni qo‘llovchi boshqa dasturlash tillari – bu BLISS (Wulf et al., 1971), INTERLISP (Teitelman, 1975) va Modula-2 (Wirth, 1985) dir.

O‘zaro bog‘liq dasturlar bir nechta kiruvchi nuqtalarga ega bo‘lib, o‘zaro bog‘liq dasturlar tomonidan boshqariladi. Ular chaqiruvlar orasida o‘z ahvolini qo‘llab-quvvatlash vositalariga ham egadir. Bu o‘zaro bog‘liq dasturlar o‘zlarining tarixlariga bog‘liq bo‘lishlari lozimligini va natijada, statistik mahalliy o‘zgaruvchilarga ega bo‘lishlari kerakligini bildiradi. O‘zaro bog‘liq dasturlarni qayta bajarish ularning boshlang‘ichi bilan mos tushmaydigan nuqtalardan boshlanadi. Buning natijasida, o‘zaro bog‘liq dasturlarni aktivlashtirishning ushbu jarayoni chaqiriq deb emas, balki yangilanish (resume) deb ataladi.

O‘zaro bog‘liq dasturlar qism dasturlarning bir xossasiga ega: bir vaqtning o‘zida faqatgina bir dona o‘zaro bog‘liq dasturlar bajarilishi mumkin. Ammo, o‘zaro bog‘liq dasturlar bajarilgunigacha operatsiyalarni bajarish ko‘pincha qo‘llaniladi va boshqaruvni boshqa o‘zaro bog‘liq dasturlarga beradi. O‘zaro bog‘liq dasturlar qayta aktivlashtirilganida operator bajarishini yangilashdan boshlaydi.

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

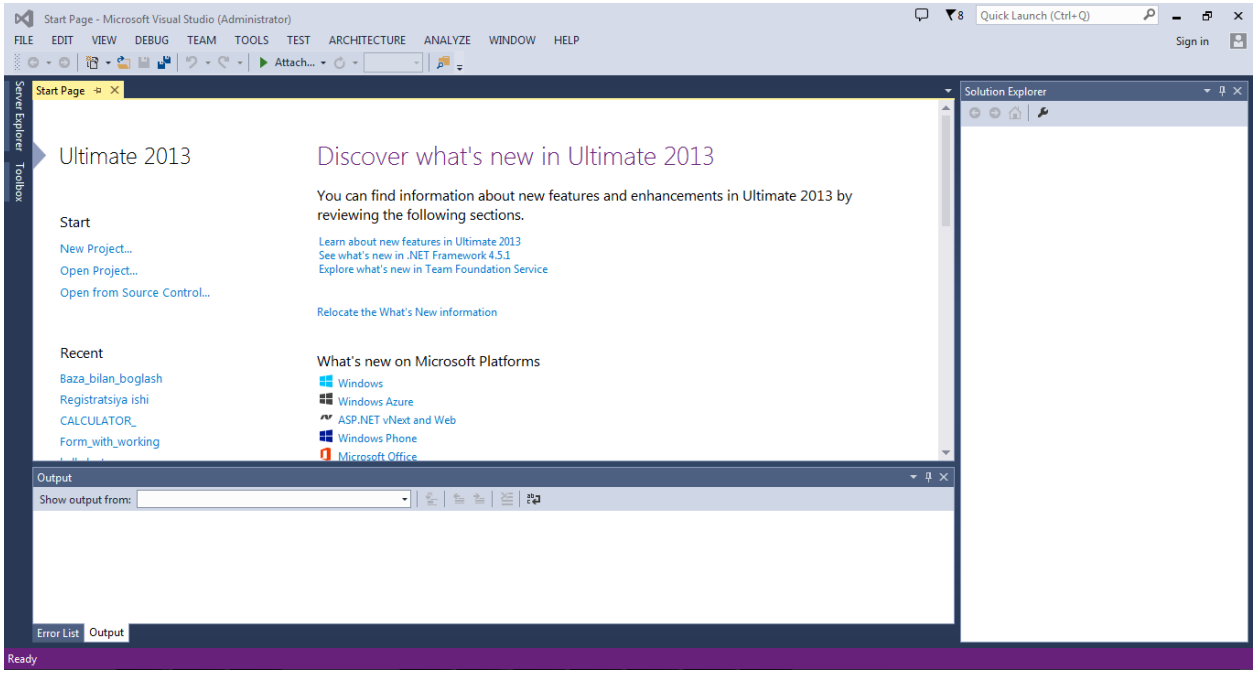
Glossariy

- Qism dastur – dasturning bir qismi bo‘lib, qachonki unga murojat qilinsagina o‘sha dastur ishga tushadi
- Void – qism dasturni qiymat qaytarmasligini anglatuvchi tipi hisoblanadi
- Lokal o‘zgaruvchi - qism dastur ichida e’lon qilgan yoki blok chegarasida ko‘rinish sohasiga ega
- Global o‘zgaruvchi - har qanday qism dasturlardan tashqarida e’lon qilinadi va programma bajarilishining oxirigacha amal qiladi. Bunday o‘zgaruvchilarga programmadan ihtiyoriy funksiyalardan murojat qilish mumkin
- New - massiv yaratishda bu operatordan foydalaniladi
- Sikl operatori – takrorlash sharti deb nomlanuvchi mantiqiy ifodani rost (true) qiymatida programmaning ma’lum bir qismlaridagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajarishni amalga oshiradi (itaratsiya)
- **Break –** operatori sikldan chiqib ketishda foydalaniladi
- **BigInteger –** katta sonlar bilan ishlashga mo‘ljallangan klass
- **continue** – bu operatori xuddi break operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin programmani qurilmadan chiqib ketmasdan takrorlashning keyingi qadamiga, for, foreach – sikl operatorlari

Amaliy mashg‘ulot.

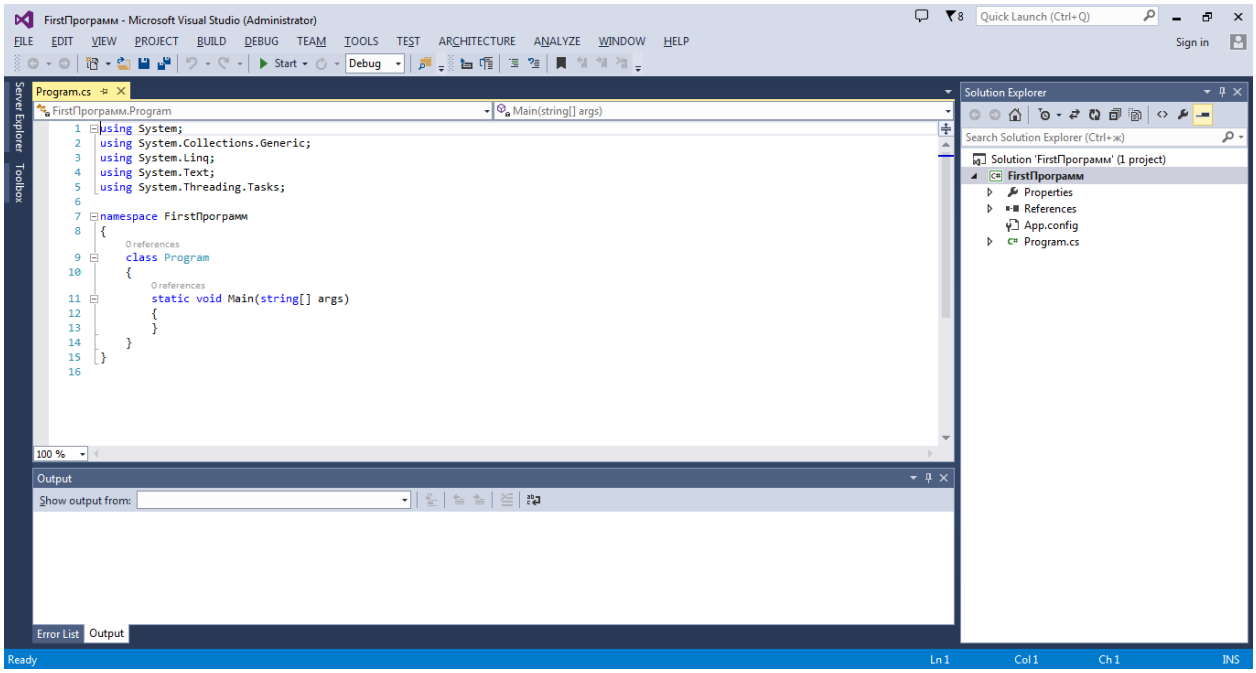
1-misol. 2 ta sonni maksimal qiymatini topib konsolga chiqarib beruvchi Printmax deb nomlanuvchi metod yarating.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **1.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



1.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **FirstProgramm** kabi kiritib, **OK** tugmasini bosamiz. Natijada **1.2 rasm**da keltirilgan quyidagi oyna shakllantiriladi.



1.2-rasm. Dasturiy kod oynasi

Endi asosiy funksiya blokidan tashqari PrintMax deb nomlanuvchi metod yaratamiz uni kodi quyidagicha

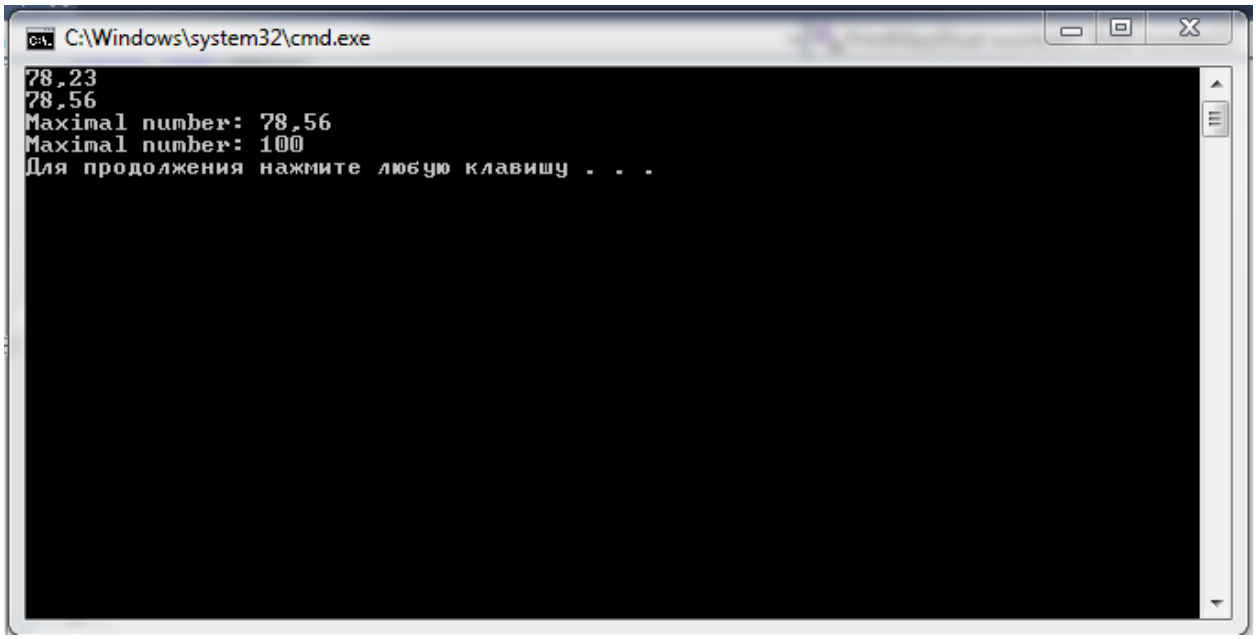
```
 static void PrintMax(float number1, float number2)
  {
   float max = number1;

   if (number2 > max)
   {
     max = number2;
   }
   Console.WriteLine("Maximal number: " + max);
     }
```

PrintMax metodini asosiy funksiyadan chaqiramiz

```
using System;

class Program
{
  static void Main()
  {
   float var1, var2;
        var1 = float.Parse(Console.ReadLine());
        var2 = float.Parse(Console.ReadLine());
        PrintMax(var1, var2);          // konsoldan qiymat o'qib metodga jo'natish
        PrintMax(100.0f, -23.5f);

      }
      static void PrintMax(float number1, float number2)
```

```
    {
      float max = number1;

      if (number2 > max)
      {
        max = number2;
      }
      Console.WriteLine("Maximal number: " + max);
    }

}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. Konsol orqali 2 ta son kiritamiz va ularni maksimal qiymatini chiqarib beradi. Bundan tashqari ekrandan kiritilgan 100.0f, -23.5f sonlarni ham maksimal qiymatini konsolga chop etadi va 1.3-rasmda keltirilgan natijaga erishamiz.

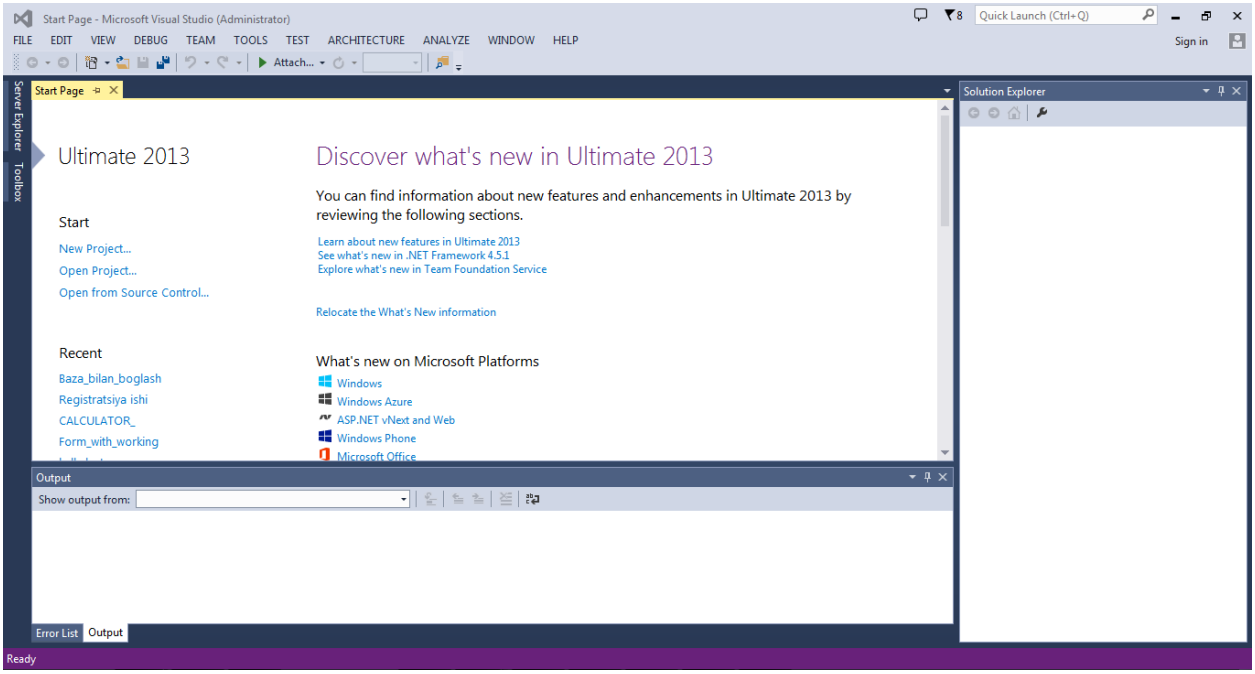


1.3-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

2-misol. Massivni berilgan o‘lchamiga qarab tasodifiy sonlarni kiritish uchun InputArray metodini yarating. Xosil bo‘lgan massivni 3 ga ham 5 ga ham bo‘linadigan elementlarini yig‘indisini topuvchi ArraySum va massivni konsolga chop etuvchi Printarray deb nomlanuvchi metodlar yarating.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **2.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



2.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **SecondProgramm** kabi kiritib, **OK** tugmasini bosamiz.

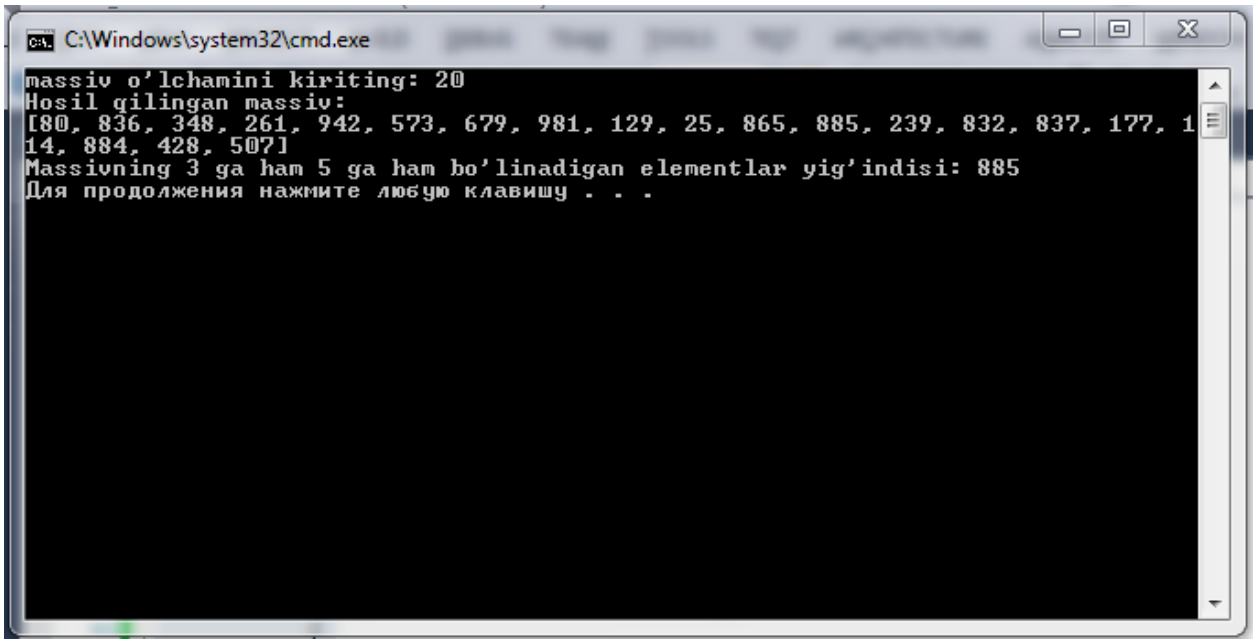
Endi asosiy funksiya blokidan tashqari **InputArray**, **PrintArray** va **ArraySum** deb nomlanuvchi metod yaratamiz uni kodi quyidagicha

```
static int[] InputArray(int n)
  {
    Random rd = new Random();
    int [] a=new int[n];
    for (int i = 0; i < n; i++)
    {
      a[i] = rd.Next(1, 1000);
    }
    return a;
```

```
  }

  static void PrintArray(int[] arrParam)
  {
    Console.Write("[");
    int length = arrParam.Length;
    if (length > 0)
    {
      Console.Write(arrParam[0].ToString());
      for (int i = 1; i < length; i++)
      {
        Console.Write(", {0}", arrParam[i]);
      }
    }
    Console.WriteLine("]");
  }

  static void ArraySum(int [] arrParam)
  {
    int summ=0;

    foreach(var arr in arrParam)
    {
      if (arr % 3 == 0 && arr % 5 == 0)
      {
        summ += arr;
      }
    }
    Console.WriteLine("Massivning 3 ga ham 5 ga ham bo'linadigan elementlar yig'indisi: {0}",
summ);
       }
PrintMax metodini asosiy funksiyadan chaqiramiz
using System;

class Program
{
  static void Main()
  {
    Console.Write("massiv o'lchamini kiriting: ");
    int n = int.Parse(Console.ReadLine());
    int[] a = new int[n];
    a = InputArray(n);

    Console.WriteLine("Hosil qilingan massiv:");
    PrintArray(a);

    ArraySum(a);
```

```
  }

  static int[] InputArray(int n)
  {
    Random rd = new Random();
    int [] a=new int[n];
    for (int i = 0; i < n; i++)
    {
      a[i] = rd.Next(1, 1000);
    }
    return a;
  }

  static void PrintArray(int[] arrParam)
  {
    Console.Write("[");
    int length = arrParam.Length;
    if (length > 0)
    {
      Console.Write(arrParam[0].ToString());
      for (int i = 1; i < length; i++)
      {
        Console.Write(", {0}", arrParam[i]);
      }
    }
    Console.WriteLine("]");
  }

  static void ArraySum(int [] arrParam)
  {
    int summ=0;

    foreach(var arr in arrParam)
    {
      if (arr % 3 == 0 && arr % 5 == 0)
      {
        summ += arr;
      }
    }
    Console.WriteLine("Massivning 3 ga ham 5 ga ham bo'linadigan elementlar yig'indisi: {0}",
summ);
  }
}
```

Endi ushbu dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. Konsol oynasi hosil bo‘ladi. Massiv o‘lchamini konsoldan kiritamiz va 1.3-rasmda keltirilgan natijaga erishamiz.



2.2-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliyot topshiriqlari

1. Eng yaxshi forvardni aniqlang va 5 tadan kam o‘yin o‘ynagan futbolistlar haqida ma’lumotni ekranga chiqaring
2. Barcha fan baholari bo‘yicha o‘rtacha balni aniqlang va o‘rtacha bali 4 dan yuqori bo‘lgan talabalar haqida ma’lumotni ekranga chiqaring.
3. Kamida bir yil oldin sotilgan maxsulotlar sonini aniqlang va ular haqida ma’lumotni ekranga chiqaring.
4. Soni 5 tadan ko‘p bo‘lgan maxsulotlar qanchaligini aniqlang va bular haqida ma’lumotni ekranga chiqaring
5. Mazkur yilda ishlab chiqarilgan barcha maxsulotlarni umumiy narxini aniqlang va bu maxsulotlar haqida ma’lumotni ekranga chiqaring.
6. Umumiy narxi eng baland bo‘lgan maxsulot nomini ekranga chiqaring
7. Fizika fani baholari bo‘yicha o‘rtacha balni, informatika bo‘yicha bahosi 5 bo‘lgan talabalar sonini aniqlang va ular haqida ma’lumotni ekranga chiqaring
8. “Ivanov” tomonidan sotilgan tovarlar sonini aniqlang, ular haqida ma’lumotni ekranga chiqaring va eng yuqori narxdagi tovarni aniqlang.

b
osib
yoki
sto
p
ope
rato
rida
n
foy
dala
nib,
ush
bu
tug
un
nuq
tad
a
o‘zg
aru
vchi
larn
ing
qiy
mat
lari
ni
teks
hiri
b
ko‘r
ish
imk
oni
yati
mav
jud.

9. Narxi o‘rtacha narxdan yuqori bo‘lgan maxsulotlar haqida ma’lumotni ekranga chiqaring
10. Betlar soni 150 tadan ko‘p bo‘lgan kitoblar haqida ma’lumotni ekranga chiqaring.
11. Tiraj 10000 nusxadan oshmaydigan kitoblar haqida ma’lumotni ekranga chiqaring
12. Oliy ma’lumotga ega bo‘lmagan, 30 yoshdan yuqori bo‘lgan hodimlar haqida ma’lumotni ekranga chiqaring
13. Hokkeychilarni o‘rtacha yoshini aniqlang va yoshi 25 dan katta bo‘lganlari haqida ma’lumotni ekranga chiqaring.
14. Tiraj 10 000 dan yuqori bo‘lgan plastinkalar haqida ma’lumotni ekranga chiqaring.
15. AMD firmasi tomonidan ishlab chiqarilgan narxi minimal kompyuterni aniqlang va u haqida barcha ma’lumotni ekranga chiqaring
16. Hokkeychilarni o‘rtacha yoshini aniqlang va yoshi 25 dan kichik bo‘lganlari haqida ma’lumotni ekranga chiqaring.
17. Mazkur yilda ishab chiqarilgan barcha maxsulatlarning o‘rtacha narxini aniqlang va haqida ma’lumotni ekranga chiqaring
18. Maxsulotlarni o‘rtacha narxini va minimal narxdagi maxsulotni aniqlang.
19. Eng kichik hodimni aniqlang va haqida ma’lumotni ekranga chiqaring
20. Matematikani “95” ga topshirgan talabalar familiyasini chop eting va ularni soni nechtaligini aniqlang
21. Omordagi soni eng ko‘p bo‘lgan maxsulotni aniqlang va u haqida barcha ma’lumotni ekranga chiqaring.
22. Operativ hotira hajmi 10 Gbaytdan katta bo‘lgan kompyuterlar soni qanchaligini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
23. Yoshi 19 dan katta bo‘lgan talabalar sonini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
24. Yoshi 60 dan katta bo‘lgan hodimlar sonini aniqlang ular haqida barcha ma’lumotni ekranga chiqaring.
25. Omordagi soni eng qimmat maxsulotni aniqlang va u haqida barcha ma’lumotni ekranga chiqaring.
26. Barcha kompyuterlarning o‘rtacha narxini hisoblang va kompyuterlarni nom va o‘rtacha narxini ekranga chiqaring.
27. Injener – hodimlarni sonini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
28. Talabalarni fizika bo‘yicha o‘rtacha ballarini xisoblang va talabalarni familiyasi, tugilgan sanasi va informatika bo‘yicha baholarini chop eting.
29. Uzog‘i bilan ikki yil oldin chiqarilgan maxsulotlarni sonini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
30. DVD ROM mavjud bo‘lgan kompyuterlarni aniqlang va ular haqida

Testlar

1. Dasturlash tili qanday abstraksiyalarni o‘z ichiga oladi?
a) jarayon abstraksiyasi va ma’lumot abstraksiyasiyalarini
b) jarayon abstraksiyasi
c) ma’lumot abstraksiyasi
d) yagona bstraksiyasi

2. Jarayon abstraksiyasi nima?
a) Qism dasturlarning shakli bo‘lib, xamma dasturlash tillarida asosiy tushuncha xisoblanadi.
b) Qism dastur.
c) Katta dastur.
d) Qism dastur o‘zagi

3. Dasturlashtirilgan 1-kompyuterning nomi qanday?
a) Uning Analitik
b) Babbage Uning Analitik Enginer
c) Babbage
d) Analitik Enginer

4. Qism dasturlarni qanday fundamental turlari bor?
a) jarayon
b) protsedura va funksiyalar
c) jarayon,protsedura va funksiyalar
d) qism dastur, protsedura va funksiyalar

5. Qism dasturlarning qisqacha qanday parametrlari o‘z ichiga oladi?
a) raqamli va rasmiy parametrlar
b) raqamli parametrlar
c) rasmiy parametrlar
d) to‘g‘ri javob berilmagan

6. Qism dasturlar odatda nimalar aks ettiradi?
a) hisoblashlarni
b) parametrlarni
c) hisoblashlarni va parametrlarni
d) sonlarni

7. Kalit so‘z parametrlariga qushimcha qanday beriladi?
a) Ada, Fortan 95+, python
b) Ada
c) Fortan 95+
d) Ada va Fortran

8. Qism dasturlarning qanday alohida kategoriyalari bor?
a) tartibi va vazifalari
b) tartibi

c) vazifalari
d) xossali

9. Lokal o‘zgaruvchilar qanday bo‘lishi mumkin?
a) statik yoki dinamik
b) statik
c) dinamik
d) lokal o‘zgaruvchi ko‘rsatkich bo‘ladi

10. Statik lokal o‘zgarvchilarni dinamik lokal o‘zgaruvchilardan afzalligi nimada
a) joylashtirish va kaytarishga ko‘prok vakt sarflaydi
b) joylashtirish va kaytarishga kamroq vakt sarflaydi
c) afzalligi yo‘q.
d) xotradagi sig‘imi kam

11. Ada tili zamonaviy boshqaruvchi tillar orasida qanday til hisoblanadi?
a) Ada tili zamonaviy tillar orasida funksiyalar har qanday tipli uzgaruvchilarni qaytara oladigan yagona til xisoblanadi.
b) Ada tili zamonaviy tillar orasida funksiyalar faqat butun tipli o‘zgaruvchilarni qaytara oladigan yagona til xisoblanadi.
c) Ada tili zamonaviy tillar orasida funksiyalar o‘zgaruvchilarni qaytara oladigan yagona til xisoblanadi.
d) To‘g‘ri javob berilmagan

12. **FORTRAN dasturlash tilida lokal o‘zgaruvchilar saqlanuvchi bloklarga kirish qaysi** operator orqali amalga oshiriladi.
a) COMMON.COM
b) COM
c) COMMON
d) CRL

13. SIMULA 67 tilining boshlang‘ich maqsadi nima?
a) Dasturlarni modellashtirish
b) Tizimlarni modellashtirish
c) Funksiyalarni modellashtirish
d) Protseduraviy modellashtirish

14. Qaysi tilda funksiyalar faqat kiritish rejimida yuboriluvchi rasmiy parametrga ega.
a) Ada tilida
b) Modula2 tilida
c) Modula1 tilida
d) Modula1 va Modula2 tilida

15. Qaysi tillarda chaqiruvlar metodning kerakli versiyasi bilan dinamik bog‘lanadi.
a) Modula1,Modula 2, Ada
b) Smalltak, Java, Ada95, S++
c) S# , SIMULA 67, Ada

d) S++, Fortran, Ada

16. Quydagi shablonli funksiyada Type qanday vazifani bajaradi.

```
Template <class Type>
Type max (Type first, Type second)
{
        return first <  second > frsit: second;
}
```

a) konstruktor
b) parametr
c) metod
d) xossa

17. Ada tilida qaysi tizmlarni parametri kabi qo‘llab bo‘lmaydi.

A) bo‘lmaydi
B) bo‘lmaydi
S) tizimga bog‘liq

18. S# dasturlash tilini ma’lumotlar ba’zasi bilan bog‘lab bo‘ladimi.

a) bo‘ladi
b) bo‘lmaydi
c) mumkun emas
d) faqat oddiy dasturlarda bog‘lanadi

19. **S# tilida kataloglar bilan ishlash uchun nomlar fazosida qaysi kutubxonadan** foydalaniladi.

a) using system IO
b) using system Netfromvork
c) using system form Application
d) using System model

20. S# tilida satrlar bilan ishlash uchun qaysi kalit suzdan foydalaniladi.

a) int
b) float
c) string
d) bool

9-mavzu. Ma’lumotlarning abstrakt tiplari va inkapsulyatsiya tuzilishi

Reja:

1. **Abstraksiya tushunchasi**
2. **Inkapsulyatsiya**
3. **Ma’lumotlr abstraksiyasiga kirish**
4. **Tiplarni rivojlantirish**
5. **Ma’lumotlar abstraksiyasining turli tillardagi ko‘rinishlari**
6. **Parametrlashgan ma’lumotlar abstrakt tipi**

Dastlab dasturlash va dasturlash tillaridagi abstraksiyaning umumiy tushunchalarini ko‘rib chiqamiz. Keyin dasturlash tillaridagi inkapsulyatsiyani muxokama qilamiz. So‘ngra aniq misollar bilan mustahkamlangan ma’lumotlar abstraksiyasining qoidasi beriladi. Undan so‘ng SIMULA67 tilidagi ma’lumotlar abstraksiyasining qisman qo‘llanilishining qisqacha tavsifi keltiriladi. Ma’lumotlar abstraksiyasi dasturlanishini tillar bilan butunlay qo‘llab-quvvatlash ikkita aniq til: Ada va C++ atamalari bilan muhokama qilinadi. Ma’lumotlar abstraksiyasi misollarini amalga oshirilishi shu tillardagi har bir ma’lumotlar bilan ko‘rsatilgan. Bu ma’lumotlar abstraksiyani qo‘llab quvvatlaydi, til vositalarini qo‘llashdagi farqlar va hususiyatlarni ajratishga yordam beradi. Ada tili alternativi sifatida Modula-2, Java tili ma’lumotlar abstraksiyasini qo‘llab-quvvatlash esa C++ alternativi sifatida ko‘rib chiqiladi. Xulosa qilib aytganda Ada va C++ tillarida abstrakt tiplar parametrlarini yaratish imkoniyatlari ko‘rsatilgan.

Shuni qayd etamizki, bu kitobda “*ma’lumotlar abstraksiyasi*” va “*ma’lumotlar abstraksiyasi turlari*” jumlalari bir ma’noni anglatadi.

Abstraksiya tushunchasi

Abstraksiya – bu faqatgina mana shu kontekstga hos bo‘lgan hususiyatlarga ega bo‘lgan ayrim ob’ektlar haqidagi tasavvur yoki fikrdir. Abstraksiya ob’ektlarni ichidan qarab chiqmasdan ya’ni ulardan mavhumlangan ob’ektlar nusxalarini guruxlarga birlashtirish imkoniyatini beradi. Bunda gurux ichida faqatgina uni bir – biridan ajratib turadigan alohida elementlar hususiyatlarigina o‘rganish kerak bo‘ladi. Bu gurux elementlarini ancha soddalashtiradi. Ob’ektlarni mukammal o‘rganish zaruriyati bo‘lganda ular haqidagi mavhum bo‘lgan tasavvurlarni ham qarab chiqish zarur. Abstraksiya – bu dasturlashni dasturlarni murakkabligiga qarshi samarali vositadir. Chunki u dasturchiga ob’ektlarni muhim xususiyatlariga diqqatni qaratishni majbur qiladi, muhim xususiyatlarini inkor qiladi. Zamonaviy dasturlash tillarida abstraksiyani 2 asosiy turi mavjud: jarayon abstraksiyasi va ma’lumotlar abstraksiyasi.

Jarayon abstraksiyasi tushunchasi – dasturlash tillari ishlab chiqarish soxasidagi ilk tushunchalardan biridir. Hatto PlankalkUl tili ham jarayon abstraksiyasini qo‘llab quvvatlaydi. Barcha qism dasturlar jarayon abstraksiyasi hisoblanadi, chunki ular qism dastur o‘rnatgan uslubni aniqlaydi (hech bo‘lmaganda chaqiruvchi dasturda) . Masalan agar dasturda ayrim tiplar ma’lumotlarni son massivini tartibga solish kerak bo‘lsa, odatda tiplash vazifasini bajaruvchi qism dastur ishlatiladi. Dasturni tiplashni amalga oshiradigan nuqtasida quyidagi operator qo‘llaniladi:

```
Sort int (list, list_len)
```

Bu chaqirishni real tiplash jarayonining abstraksiyasi bo‘lib,, uning algoritmi aniqlanmagan. Chaqirish chaqirayotgan qism dastur amalga oshiradigan algoritmiga bog‘liq emas. **Sort_int** qism dasturning yagona muxim xususiyatlari tartib solinadigan massivni nomi uning elementlarining tipi va shu faktki **Sort_int** qism dastur massivni saralashga olib keladi. **Sort_int** aynan qanday algoritmni bajarishi foydalanuvchi uchun muxim emas.

Jarayon abstraksiyasi – dasturlashdagi asosiy tushunchadir. Qism dastur bajaradigan algoritmni ko‘p detallardan abstraksiyalash imkoniyatlari katta dasturlarni yaratish , o‘qish va tushintirish imkoniyatini beradi. Eslatamizki hech bo‘lmaganda bir necha ming kod qatoriga ega bo‘lgan dastur xozirgi paytda katta dastur hisoblanadi. Barcha qism dastur jumladan (oldingi mavzuda ko‘rilgan) parallel qism dasturlar hamda boshqa xolatlarni qayta ishlash jaryonlarini o‘z ichiga olganlar jarayonlar abstraksiyasi hisoblanadi. Ma’lumotlar abstraksiya evolyusiyasi jarayonlar evolyusiyasi birgalikda olib boriladi. Chunki har qanday abstraksiyaning muxim va ajralmas qismi jarayonlar abstraksiyasi operatsiyasi yordamida aniqlanadi.

Inkapsulyatsiya

Inkapsulyatsiya ma’lumotlar tiplari abstraksiya tiplaridan oldin keladi va ularni qo‘llab-quvvatlaydi. Dastur o‘lchami bir necha ming qatorga yetganda 2 ta amaliy muammo paydo bo‘ldi. Dasturchilar nuqtai nazarida dasturga qism dasturlarning yagona to‘plami sifatida qarash dastur tashkil qilish, uni boshqarishning adekvat darajasini ta’minlaydi. Dasturni mantiqiy bog‘liq qism dasturlar va ma’lumotlarga ega bo‘lgan guruhlarini o‘z ichiga olgan sintaksik konteynerlarga bo‘lib, bu muammoni yechish mumkin. Bu sintaktik konteynerlar modullar deb ataladi, ularni qayta ishlash jarayoni modulyarizatsiya deyiladi. Katta dasturlar bilan bog‘liq bo‘lgan ikkinchi amaliy muammo takroriy kompilyatsiya muammosi. Kichchik dastur uchun xar bir modifikatsiyadan kiyin butun dasturni takroran kompilyatsiya qilish xech narsa emas. Biroq dasturning o‘lchami bir necha ming qatorgacha yetsa takroriy kompilyatsiyaning bahosi ancha katta bo‘ladi. Demak dastur o‘zgarmas qismlarining takroriy kompilyatsiyasiga yo‘l qo‘ymaslik usullarini topish kerak. Buni qilish uchun shunday ma’lumotlar va qism programmalar to‘plamidan iborat dastur tuzish kerakki ulardan har biri aloxida dasturni boshqa qismini takroriy kompilyatsiya qilmasdan alohida qismlarini olib qilish mumkin. Bunday to‘plam kompilyatsiya birligi deyiladi.

Inkapsulyatsiya – bu qism dasturlarning va ular qayta ishlaydigan ma’lumotlarni yagona qilib birlashtirish usulidir. Aloxida kompilyatsiya qilinadigan yoki boshqa dasturlardan mustaqil bo‘lgan inkapsulyatsiya abstraksiya tizimining va to‘g‘ri keluvchi xisob-kitoblar yig‘indisini mantiqiy tashkil qilishning asosidir. Shundan kelib chiqadiki yuqorida ko‘rsatilgan ikkala muammoni ham inkapsulyatsiya yechadi.

Inkapsulyatsiyalarni ko‘pincha kutubxonalarga joylashtirishadi va ularni boshqa dasturlarda qayta qo‘llash uchun ochiq qoldiradilar.

Dasturchilar o‘lchami bir necha ming bo‘lgan dasturlarni 40 yillardan buyon yozib keladilar, demakki inkapsulyatsiya yaratish texnikasi allaqachonlar rivojlanib ulgurgan.

Ko‘pgina algoritmik dasturlash tillarida ular qo‘llaydigan qism dasturlardan ancha yuqori darajada bo‘lgan qism dasturlar ko‘rinishida tashkil qilish mumkin. 4- mavzuda muhokama qilinganidek tasvir kontekstidan foydalanadigan dasturni tashkil qilish uslubi idealdan ancha uzoq. Bundan tashqari dasturlashning ayrim tillarda qism dasturlar kompilyatsiyasining birliklari hisoblanmaydi. Demakki ularni inkapsulyatsiya uchun mos konstruksiya deb atalmaymiz.

FORTRAN 77 tilida qism dasturlarni fayllarga yeg‘ish, bir biridan mustaqil holda kompilyatsiya qilish va kutubxonalarga joylashtirish mumkin. COMMON - bloklar tavsiflari to‘plamlarni analogik tarzda qayta ishlash mumkin. Bu dasturlarning tashkil qilishning samarali

usuli. Lekin bunday inkapsulyatsiyalarning ishlatishda interfeys tekshiruvi yo‘q. Shuning uchun bunday yondashuv o‘z tabiyatiga ko‘ra xavfsiz emas.

C tilida ma’lumotlar tavsifi va o‘zaro bog‘liq funksiyalarni to‘plamlarini boshqa fayllardan mustaqil kompilatsiya qilingan boshqa faylga joylashtirish mumkin. C tili kompilyatorlari hozirgi paytda kerakli darajada ayrim funksiyalarning interfeysi to‘g‘riligini tekshirishiga qaramay ular hali ham boshqa fayllardagi ma’lumotlar aniqligini tekshirishni amalga oshirmaydi(“ Kerakli darajada aniq ” jumlasi ostida biz shunday funksiyalarni nazarda tutamizki ular ANCI S standartiga mos kelmaydigan funksiyalar sarlavhasidan foydalanmaydilar). Demak C fayllari ham xavfsiz inkapsulyatsiyani ta’minlay olmaydi. Ko‘pgina zamonaviy tillar , jumladan FORTRAN 90 va Ada qism dasturlar to‘plamlarini turlarini va ma’lumotlarini modullarga yig‘ish imkoniyatini beradi. Bu modullarni alohida kompilyatsiya qilish mumkin. Bunda shuni nazarda tutish kerakki ularni interfeysi haqidagi ma’lumotlar kompilyatorda interfeys tipini boshqa modulda ishlatilganda tekshirish uchun saqlanadi. Ko‘rib chiqilayotgan tillar shuningdek bu modular asosini boshqarish mexanizmiga ega. Bu esa modulga tashqi modullarga ko‘rinarli bo‘lgan ayrim tiplarni o‘zida aks ettirish imkoniyatini beradi. Bunda bu tiplarni namoyish etish boshqa mohiyatlar uchun faqat modul ichida mumkin. Bu modullar a’lo darajadagi inkapsulyatsiyani ta’minlaydi . Ular faqatgina dasturning aniq va mantiqiy tashkil qilishi qo‘llab-quvvatlab qolmasdan balki bu dasturiy tuzilmani dasturchilar uchun tushunarli qiladi.

SIMULA 67, Ada va S++ tillaridagi inkapsulyatsiyani ta’minlash vositalarini o‘ziga hos hususiyatlari ma’lumotlar abstract tiplari bilan birga 5 qismida muhokama qilinadi.

3 Ma’lumotlr abstraksiysiga kirish

Ma’lumotlar abstrakt tipi – bu shunday inkapsulyatsiyaki uning tarkibiga bitta aniq tip ma’lumotlari va shu tip ma’lumotlari bilan operatsiyalar bajaruvchi qism dasturlar kiradi. Kirishni boshqarish yordamida tipni tasvirlashning muhim bo‘lmagan qismlarini shunday tip foydalanadigan tashqi moduldan berkitish mumkin. Ma’lumotlarning abstrakt tipi foydalanadigan dasturli modular shu tipni o‘zgarishlarini e’lon qilishi mumkin, garchand tipning aniq tasavuri ulardan berkitilgan bo‘lsa,da, ma’lumotlar abstrakt tipi nusxasi ob’ekt deb ataladi.

Ma’lumotlar abstrakt tipi va jarayon abstraksiyasi yaratilishining bitta umumiy sababi bor. Bu – murakkablikka qarshi vosita, katta va murakkab dasturlarni boshqara oladigan usuli. Yaratishning boshqa mavzulari va ma’lumotlar abstrakt tipilari afzalliklari shu mavzuning davomida muxokama qilinadi huddi jarayonlar abstraksiyasidek jarayonlar abstraksiyasi ham dasturlashning butkul turli xil usullariga ijozat beradi.

So‘ngi yillarda dasturlash ta’minoti ishlab chiqarishning ommaviyligi oshib boryotgan yangi usuli – ob’ektga yo‘naltirilgan dasturlash. Keyingi mavzuda tasvirlangan ob’ekt yo‘naltirilgan dasturlash, bu dastur ishlab chiqarishda ma’lumotlar abstraksiyasi qo‘llanishi natijasidir, ma’lumotlar abstraksiyasi esa uning muxim tarkibiy qismlaridan biridir.

3.1. haqiqiy (haqiqiy (suzuvchi nuqtali)) son ma’lumotlar abstrakt tipi sifatida

Ma’lumotlar abstrakt tipi tushunchasi chunonchi kiritilgan tiplar atamalarida – yangi kashfiyot emas. Barcha ma’lumotlarni kiritilgan tiplar hattoki FORTRAN 1 tilidagilari ham abstraksiya hisoblanadi, garchand ularni ham holatlarga shunday atasa ham. Misol uchun haqiqiy (suzuvchi nuqtali) sonlarni qarab chiqamiz. Dasturlashning ko‘pgina turlari bunday sonlarni hech bo‘lmasa bitta tipi haqidagi tasavvurlarni o‘z ichiga oladiki , bu tip o‘zgaruvchilarining yaratish va ular bilan arifmetik amallar bajarish imkoniyatini beradi. Yuqori darajadagi tiplar tillarida haqiqiy sonlarni taqdim etish uchun ma’lumotlar abstraksiyasida kalit tushuncha: ma’lumotlar berkitish.

Haqiqiy sonlarni saqlashga mo‘ljallangan hotira yacheykasidagi ma’lumotlar haqidagi real tasavvur foydalauvchidan berkitilgan. Ular bilan faqatgina tilda ko‘zda tutilgan operatsiyalarni yoki jarayonlarni bajarish mumkin. Foydalanuvchi bu tip ma’lumotlari bilan operatsiyalar yarata olmaydi. Kiritilgan operatsiyalar yordamida yaratilganlaridan tashqari. Bundan tashqari haqiqiy sonlar aniq tasavvurlarining qismlari bilan bevosita minipulyatsiya qilish mumkin emas. Chunki bu tasavvurlar foydalanuvchidan berkitilgan. Shunday qilib tilning aniq realizatsiyasi o‘rtasida dasturlarning o‘tuvchanligi ta’minlanadi., garchand bu realizatsiyalar haqiqiy sonlarning har-xil tasavvurlarida ishlatilsa ham.

3.2. Foydalanuvchi tomonidan aniqlangan ma’lumotlarning abstrakt tiplari

Foydalanuvchi tomonidan aniqlangan ma’lumotlar abstraksiyasi tushunchasi nisbatan yaqin vaqt ichida paydo bo‘ldi. Foydalanuvchilar tomonidan ma’lumotlar abstraksiyasi tiplari huddi haqiqiy sonlarning hususiyatlari kabi bo‘lishi kerak:

1) Tipni aniqlashning modullari dasturlariga bu tip o‘zgaruvchanligi e’lon qilishga yo‘l qo‘yadigan hotirada bu o‘zgaruvchilar haqida aniq tasavvur hosil qiladigan aniq qoida

2) Berilgan tip ob’ektlari bilan minipulyatsiyalari uchun operatsiyalari to‘plami.

Keyinchalik foydalanuvchi aniqlagan tiplar kontekstidagi ma’lumotlar abstrakt tipining shakliy qoidasi keltiriladi. Ma’lumotlar abstrakt tipi – bu quyidagi ikki shartni qoniqtiradigan ma’lumotlar tipidir:

- Tip haqidagi tasavvur va shu tipdagi ob’ektlar ustidagi operatsiyalar bitta sintaktik birikmada mavjud. Bundan tashqari bu tip o‘zgaruvchilarini boshqa modullarda ham yaratish mumkin.
- Berilgan tip ob’ektlari haqidagi tasavvur dasturiy modullardan bekitilgan, demakki bunday ob’ektlar ustida faqatgina tip aniqlanishida to‘g‘ridan – to‘g‘ri ko‘zda tutilgan operatsiyalarni amalga oshirish mumkin.

Ayrim ma’lumotlar abstrakt tipidan foydalanuvchi dasturiy modullar shu tipning modulari deb ataladi.

Tip va operatsiyalarni alohida sintaktik birikmaga joylashning asosiy afzalliklari huddi inkapsulyatsiyadagidek, ya’ni, birinchidan bu dasturlarni alohida kompilatsiya qilish mumkin bo‘lgan logik birikmalarni tashkil qilish imkonini beradi, ikkinchidan shu tip ob’ektlaridagi tasavvurlar yoki ular bilan bo‘lgan operatsiyalarni dasturni alohida qismida modifikatsiya qilish imkoniyati paydo bo‘ladi. Tip tasavvuri detallarini ochishning bir qancha afzalliklari bor. Bulardan eng muhimi shuki, mijozlar ob’ektlar tasavvuri detallarini ko‘ra olmaydilar va bundan kelib chiqadigi ularni ko‘di bu tasavvurga bog‘liq emas. Shunday qilib ob’ektlar tasavvurlarini xoxlgan paytda o‘zgartirish mumkin. Bunda mijozlar kodini o‘zgartirishni talab qilishni xojati yo‘q. Abstrakt interfeysni uning ayrim (hammasi emas) xususiyatlarini namoyish etadi.

Ma’lumotlar ochishning boshqa aniq va muhim ustinligi uning yuqori ishonchliligidir. Mijozlar ob’ektning asosiy tasavvurlarini bilvosita ataylab yoki tasodifan o‘zgartira olmaydilar. Shuning uchun bunday ob’ektlarning yaxlitligi ortadi. Ob’ektlarni faqatgina ular uchun mo‘ljallangan operatsiyalar yordamida o‘zgartirish mumkin. Ma’lumotlar abstrakt tiplari tasavurlarini ochish detallarini muhimligini qayta baxolash qiyin.

3.3 Misol

Aytaylik stek uchun berilganlar abstrakt tipini yaratish zarur, buning uchun quyidagi abstrakt operatsiyalarga ega:

create(stack) stek tipidagi ob’ektni yaratish va nomlash mumkin.
destroy(stack) stek egallaydigan xotirani bo‘shatadi.

empty(stack) predikat(bul) funksiyasi , “xaqiqat” ma’nosini qaytaradi va aksincha agar stek bo‘sh bo‘lsa, “yolg‘on” qiymat qaytaradi.

push(stack,element) ko‘rsatilgan elementni ko‘rsatilgan stekka joylashtiradi.

top(stack) ko‘rsatilgan stekdan yuqori elementni o‘chirib tashlaydi.

top(stack) ko‘rsatilgan stekdan yuqorida turgan elementni nus’hasini qaytaradi.

Qayd etamizki ma’lumotlar abstrakt tipini ayrim qo‘llashlar ob’ektlarini yaratish va buzish operatsiyalarini ko‘zda tutmaydi . Masalan ma’lumotlar abstrakt tipining o‘zgaruvchilarining sodda qoidasi ma’lumotlar zarur strukturasini noaniq tuzishi mumkin va noaniq nomlashi mumkin. Stek tipidagi mijoz quyidagi birin ketinligidagi operatorlariga ega bo‘lishi mumkin:

```
....
create(STK1);
 push(STK1, COLOR1);
 push(STKl, COLOR2);
 if(not empty(STK1))
 then TEMP := top(STKl);
....
```

Aytaylik stek abstraksiyasining dastlabki qo‘llanilishi uni massiv elementlarining birin ketinligi ko‘rinishida taqdim etadi. Keyinchalik xotira boshqarishdagi muammalar tufayli massivga bog‘liq stek o‘zaro bog‘liq ro‘yxat ko‘rinishida taqdim etila boshlanadi. Ma’lumotlar abstraksiyasidan foydalanilganligi tufayli bunday o‘zgaruvchilarning stek tipini aniqlovchi kodga kiritish mumkin , lekin bunda , bironta mijoz kodiga o‘zgartirish mumkin emas. Shuningdek yuqorida keltirilgan operatorlar ketinligini o‘zgartirishsiz qoldirish mumkin. Albatta har qanday operatsiya pratakolidagi o‘zgarishlar mijozlar kodlaridagi o‘zgarishlarni talab qilishi mumkin.

Agar stek ma’lumotlar abstrakt tipi sifatida tadbiq etilmagan bo‘lsa, bunday o‘zgarishlar mijozlar kodlarini o‘zgarishlariga olib kelgan bo‘ladi. Aytaylik stek bilan operatsiyalar. Ada tilida massiv operatsiyalari sifatida amalga oshirilgan stek taqdimotlarining massivdan bog‘langan ro‘yxatga o‘zgarishi mijozlar kodlari modifikatsiyasini talab qiladi . Endi ular ko‘rsatkichlarni berish kerak , lekin stek operatsiyalarini amalga oshiruvchi jarayon parametrlarining massivlari nomlari sifatida emas. Bu holatda ayrim operatsiyalar pratakollari o‘zgarishi kerak, oqibatda mijozlar kodlariga o‘zgarish kirishi mumkin. Xulosa qilib aytamizki ma’lumotlar abstraksiyasining maqsadi dasturlarga ma’lumotlar tiplarini aniqlash imkoniyatini berishdir.

Tiplarni rivojlantirish

Dasturlash tilida ma’lumotlar abstarakt tiplarini aniqlash vositalari sintaksik birikmalar yaratishni ta’minlaydi, ular tip va abstrakt operatsiyalarni bajaruvchi qism dasturlar qoidalarini inkapsulyatsiya qilish mumkin. Tip nomini va qism programmalar sarlovhalarini abstrakt mijozlariga ko‘rinarli qilish imkoniyati mavjud bo‘lishi kerak. Bu mijozlarga abstrakt tip o‘zgaruvchilarini e’lon qilish va ular ma’nolari bilan manipulatsiya qilish imkoniyatini beradi . Tipning nomi tashqaridan ko‘rinarli bo‘lishiga qaramay uning aniqlanishi yashirin bo‘lishi kerak. Tip aniqlovchi operatsiyalarga nisbatan abstrakt tiplar ob’ektlari bilan bariladigan umumiy kirish operatorlari juda kam. Shunchaki ma’lumotlar abstrakt tipi keng doirasiga kirishi mumkin bo‘lgan katta miqdordagi operatsiyalar mavjud emas. Bunday operatsiyalarga o‘zlashtirish ,shuningdek tenglik va tengsizlik operatsiyalari kiradi. Agar til foydalanuvchiga o‘zlashtirish operatsiyasini yuklash imkoniyatini bermasa, u holda u o‘rnatilgan bo‘lishi kerak. Tenglik va tengsizlikni tekshirish

oldindan aniqlangan ayrim hollarda aniqlanmagan. Masalan ko‘rsatkichlar tenglikni bildiradi. Lekin foydalanuvchi bunday tenglik manzillari ko‘rsatkichlarda saqlangan strukturalar tenglikni bildirishni xoxlashi mumkin.

Ayrim operatsiyalar ma’lum abstrakt tiplari ko‘pchilgi uchun kerak. Lekin ular universal bo‘lmaganligi tufayli tipni ishlab chiqaruvchi ularni o‘zi aniqlashi kerak. Bunday opeatsiyalarga iteratsiyalar, konstruktorlar va destruktorlar kiradi. Iteratsiyalar oldingi mavzularda muhokama qilingan. Konstruktorlar yangi yaratilgan ob’ektlar qisimlarini nomlash uchun foydaliniladi. Destruktorlar abstract tiplar ob’ektlari bilan band bo‘lgan dinamik xotirani qisimlarini bo‘shatish uchun ishlatiladi. Oldin aytib o‘tilganidek ma’lumotlarning alohida tipining va ular bilan bog‘liq operatsiyalarni inkapsulyatsiya amalga oshiridi. Concurrent Pascal (Brinch Hansen, 1975), Smalltalk (Goldberg and Robson, 1983), C++ va Java tillari ma’lumotlari abstrakt tipini bilvosita qo‘llab-quvvatlaydi. Bunga alternativ umumiy konstruksiyalarni qo‘llab quvvatlash bo‘lib, ular predmetlarning har qanday miqdorini aniqlatdi, ularning har biri alohida xolida moduldan tashqarida ko‘rinadi. Bu yondashish Modula-2 va Ada dasturlash tillarida amalga oshirilgan. Biz bu loyihalarni inkapsulyatsiya deb atadik. Inskopsulatsiya ma’lumotlar abstract tipi emas balki ularni umumlashtiruvchisidir va shu hususida inkapsulyatsiya dan ma’lumotlar abstrakt tipini aniqlashda foydalanish mumkin.

Tiplar ishlanmasi asosiy savollarini inkapsulyatsiya yordamida belgilaymiz. Birinchidan abstract bo‘lishi mumkin bo‘lgan ko‘pgina tiplarni cheklash kerakmi? Bunaqa chegaralash bir qancha ustunliklarga ega. Xususan faqat ko‘rsatkichlar abstract bo‘lsa, dasturlarni ishlashda takroriy kompilyatsiya bilan bog‘liq bo‘lgan katta muammolardan qochish mumkin. Boshqa tamondan ayrim tadqiqotchilar buni juda kuchli cheklanish deb biladilar, chunki u ko‘pgina kamchiliklarga ega. Tiplar ishlanmalarining ikkinchi savoli ma’lumotlar abstrakt tipini parametrlash mumkinmi? Masalan agar dasturlash tili ma’lumotlar tiplarini parametrlashni qo‘llab-quvvatlasa ma’lumotlar abstrakt tiplarini har qanday skalyar tipdagi elementlaran iborat bo‘lgan ma’lumotlar abstrakt tiplarining navbatlarini ishlab chiqish mumkin. Ma’lumotlarning parametrlashgan abstrakt tiplari keying mavzuda muhokama qilinadi. Xulosa qilib aytamizki yana shunday savollar mavjudki ma’lumotlarga kirishni boshqarish qanday amalga oshiriladi va bu boshqaruv qanday aniqlanadi.

5 Ma’lumotlar abstraksiyasining turli tillardagi misollar

Bu mavzuda ma’lumotlar abstraksiyasini qo‘llab quvvatlash CIMULA 67, Ada, C++ va Java dasturlash tillarida ko‘rib chiqiladi.

5.1. SIMULA 67 tilidagi sinflar

Tildagi birinchi ma’lumotlar abstraksiyasini to‘g‘ridan-to‘g‘ri qo‘llash vositalari garchand bizning aniqlshimizga butunlay mos kelmasAda CIMULA 67 tilidagi sinflar qurilmalarida paydo bo‘lgan.

5.1.1. Inkapsulyatsiya

SIMULA 67 tilida sinfning tarifi tip tasviridir. Nusxalar (sinflar Ob’ektlari) dinamik xotirada foydalanuvchi dasturlar talabi bilan yaratiladi va ularga kirish faqat ko‘rsatkichlar orqali amalga oshiriladi. Shunday qilib sinf ob’ektlari dinamik hisoblanadi.

SIMULA 67 tilidagi sinf ta’rifining umumiy sintaktik formasi quyidagi ko‘rinishga ega:

```
class class_name;
begin
-- sinf o‘zgaruvchilarini e’loni --
-- sinf qism dasturlarining ta’riflari --
-- sinf kodi mavzui --
 end class_name;
```

Kodga ega bo‘lgan sinf aniqlanish sohasi faqat bir marta ob’ekt yaratilgan paytda bir marta bajariladi. U sinf loyihachisi vazifasini bajaradi va shu vazifada sinfda aniqlangan o‘zgaruvchilarning nomlanishida foydalaniladi. SIMULA 67 tilining ma’lumotlar abstraksiyasi ishlanmasiagi hissasi inkapsulyatsiya imkoniyatlariga yeg‘iladi.

Shunisi qiziqarliki sinflar hususiyatlarining ahamiyti SIMULA 67 ishlanmasi yakunlanganidan bir necha yil keyin ham tan olinmagan. Ma’lumotlar abstraksiyasini muhimligi 70-yillar oxirlarigacha anglab yetilmagan.

5.1.2. Ma’lumotlarni ochish

SIMULA 67 tili sinfida e’lon qilingan o‘zgaruvchilar shu sinf ob’ektini yaratuvchi mijozlardan berkitilmagan. Bu o‘zgaruvchilarga kirishni sinf qism dasturlari bajarayotgan operatsiyalari orqali yoki bevosita ularning nomlaridan olish mumkin. Bu ma’lumotlar abstrakt tipini aniqlashdagi informatsiyani ochish talablariga zid. Chunki sinf mohiyatiga kirishning ko‘p uslublari mavjud. Natijada SIMULA 67 til sinflari haqiqiy ma’lumotlar abstrakt tipiga qaraganda kamroq ishonchlidir. Bundan tashqari sinf mijozlari sinfdagi ma’lum o‘zgaruvchilarga bog‘liq bo‘lganligi tufayli bu o‘zgaruvchilar aniqlanishidagi o‘zgarishlar mijozlar kodlaridagi o‘zgarishlarni keltirib chiqaradi va bu bunday dasturlarni qo‘llab-quvatlashni murakkablashtiradi.

5.1.3. Baxolash

SIMULA 67 tilidagi sinf qurilmasi ma’lumotlar taqdimoti detallarini sinf mijozlari o‘zgartirishining asrash inkapsulyatsiyani ta’minlaydi, ma’lumotlar ochilishini emas. SIMULA 67 tili o‘zining sinf qurilmasini ishlashda inqilobiy edi. Biroq dasturlashning bu turi keng tarqalmaganligi tufayli bu yerda biz uni faqat tarixiy qiziqish nuqtai nazaridan eslab o‘tdik. Endi esa ma’lumotlar abstraksiyasini to‘liq qo‘llab-quvvatlashni ta’minlaydigan ikkita zamonaviy Ada va C++ dasturlash tillarini ko‘rib chiqishga qaytamiz.

5.2. Ma’lumotlar abstrakt tiplari Ada tilida

Ada tili inkapsulyatsiya sining ma’lumotlar abstrakt tiplarini modellashtirishda ishlatiladigan vositalarni ta’minlaydi. Shuningdek ularni taqdim qilish imkoniyatini yashiradi.

5.2.1. Inkapsulyatsiya

Ada tilidagi inkapsulyatsiya qurilmalari paketlar deb ataladi. Paketlar ikki qisman iborat bo‘lib, ularning har biri ham paket deb ataladi. Ular inkapsulyatsiya interfeysini ta’minlaydigan paket tasnifi va tasnifda sanalgan mohiyatlar tadbiqini ta’minlaydigan paket tanasi deb ataladi. Barcha paketlar ham tanaga ega emas (faqat tiplar va konstantalarni inkapsulyatsiya qiladigan paketlar tanaga ega emas va ularga muhtoj ham emas)

Paket tasnifi va u bilan bog‘liq paket tanasi bir xil nomga ega. Paket sarlavhasidagi qayd etilgan body so‘zi u paket tanasi ekanligini bildiradi. Paketning tasnifi va tanasi faqatgina paket tasnifi birinchi bo‘lib, kompilyatsiya qilingan sharaoitdagina aloxida kompilatsiya qilinishi mumkin.

5.2.2. Ma’lumotlarni ochish

Paket tasnifini tasvirlash uchun va ularni eksport qilish uchun qo‘llaniladigan tiplar uchun hech qanday chegirmalar yo‘q. Foydalanuvchi, yoki mijozlarga butunlay ko‘rinarli bo‘lgan mohiyatni yaratish mumkin yoki uning interfeysi haqidagi ma’lumotlarni ta’minlashi mumkin. Buning uchun paket tasnifida ikkita mavzu qo‘llaniladi. Ulardan birinchisida mijozlarga ko‘rinarli mohiyat aks etadi, ikkinchisida o‘z mazmunini yashiradi. Masalan agar tip eksport qilinishi kerak bo‘lsa, lekin uning taqdimoti yashirilgan bo‘lsa, tasnifning ko‘rinadigan qismida faqatgina tipning ismi va uning taqdimoti yashirincha ekanligi haqidagi fakt qisqartirilgan e’lon shaklida joylashtiriladi. Tipning taqdimoti yopiq (private) deb ataladigan tasnif qismida joylashtiriladi va zaxiralangan private so‘zidan boshlanadi. Yopiq mavzu doimo tasnifning oxirida joylashtiriladi.

Aytaylik NODE_TYPE nomli tip paket bilan eksport qilinayapti, lekin uning taqdimoti yashiringan. NODE_TYPE tipi paket tavsifining ko‘rinadigan qismida uning taqdimot detallarisiz quyida ko‘rsatilganidek e’lon qilinadi:

```
type  NODE_TYPE is private;
```

E’lonlarning yopiq mavzusida NODE_TYPE tipi takrorlanadi lekin bu safar tipning to‘liq aniqlanishi bilan:

```
package LINKED_LIST_TYPE is type NODE_TYPE is private;
private
type NODE_TYPE;
type PTR is access NODE_TYPE;
type NODE_TYPE is
record
INFO : INTEGER;
LINK : PTR;
end record;
end LINKED_LIST_TYPE;
```

Agar paketdagi mohiyatlarning hech biri yopiq bo‘lmasi tasnifini yopiq qismi kerak emas. Paketlar tasnifining taqdimoti sabablari kompilyatsiya masalalari bilan bog‘liq. Mijoz faqat paket tasnifini ko‘rishi mumkin (paket tanasini emas) , kompilyator esa mijoz kompilyatsiyasida eksport qilinadigan tiplar ob’ektlarini joylashtirishi kerak. Bundan tashqari mijoz faqatgina ma’lumotlar abstract tiplari paket tasnifi mavjudligidagina kompilyatsiya qilinishi mumkin. O‘z navbatida kompilyator tasnif paketidagi ob’ekt o‘lchamini aniqlash imkoniyatiga ega bo‘lishi kerak. Demak ma’lumotlar tipining taqdimoti mijoz kodi uchun emas balki, kompilyator uchun ochiq bo‘lishi kerak. Bu aynan shunday holatki paket tasnifiniyopiq mavzui bilan aniqlanadi. Yopiq deb e’lon qilingan tiplar yopiq tiplar deb ataladi (private types). Ma’lumotlarni yopiq tiplari o‘zlashtirish va qiyoslash qurilma operatsiyalariga ega. Qolgan barcha operatsiyalar tip aniqlaydigan paket tasnifida e’lon qilinishi kerak. Yopiq tiplarning o‘rindoshlari bo‘lib, cheklangan yopiq tiplar (limited private types) hizmat qiladi, ular yopiq mavzuda tasvirlanadi. Ular orasidagi yagona sintaktik farq

shundaki , cheklangan yopiq tiplar paket tasnifining ko‘rinarli qismidagi rezervlangan *Limeted private* so‘zlari bilan e’lon qilinadi. Cheklangan yopiq tip ob’ektlari Kiritilgan ob’ektlarga ega emas. Bunday tiplar odatdagi o‘zlashtirish va taqqoslash operatsiyalar foydasiz bo‘lgan holatda kerak bo‘ladi. Masalan o‘zlashtirish o‘zlashtirish va taqqoslash steklari uchun kamdan kam ishlatiladi. Agar o‘zlashtirish va qiyoslash operatsiyalarini bajarish kerak bo‘lsa, lekin ularning kiritilgan versiyalari bunga mos bo‘lmasa bu operatsiyalarni paket tasnifida ko‘rib chiqishi kerak. O‘zlashtirish operatsiyasini oddiy protsedura shakliga ega bo‘lishi kerak, ayni paytda tenglik va tengsizlikni tekshirish operatorlari ular operatorlarini yangi tipga nisbatan qaytayuklanish yo‘li bilan amalga oshirilishi mumkin.

5.2.3. Na’muna

Quyida ma’lumotlar abstrakt tipi uchun maxsus paketining stekni tasvirlovchi tasnifi keltirilgan

```
package STACKPACK is
-- Ko‘rinarli mohiyat yoki ochiq interfeys –
type STACKTYPE is limited private;
MAX_SIZE : constant := ICO;
function EMPTY(STK : in STACKTYPE) return BOOLEAN;
procedure PUSH(STK : in out STACKTYPE;
ELEMENT : in INTEGER);
procedure POP(STK : in out STACKTYPE);
function TOP(STK: in STACKTYPE) return INTEGER;
-- Mijozlardan berkitilgan qism private --
type LISTJL'YPE is array (1. .MAX_SIZE) of INTEGER;
type STACKTYPE is record
LIST : LIST TYPE;
TOPSUB : INTEGER range O..MAX_SIZE := 0;
end record; end STACKPACK;
```

Qayd etamizki yaratish va buzish operatsiyalari paket tasnifida kritilmagan chunki ularga zarurat yo‘q.

```
Paket tanasi STACKPACK:
with TEXT_IO; use TEXT_IO;
package body STACKPACK is
function EMPTY(STK: in STACKTYPE) return BOOLEAN is begin
return STK.TOPSUB = 0;
end EMPTY;
procedure PUSH(STK : in out STACKTYPE;
ELEMENT : in INTEGER) is
begin
if STK.TOPSUB >= MAX_SIZE then
PUT_LINE("XATO – Stek to’lgan”);
else
STK.TOPSUB := STK.TOPSUB + 1;
STK.LIST(TOPSUB) := ELEMENT;
```

```
end if;
 end PUSH;
procedure POP(STK : in out STACKTYPE) is begin
if STK.TOPSUB = 0
```

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

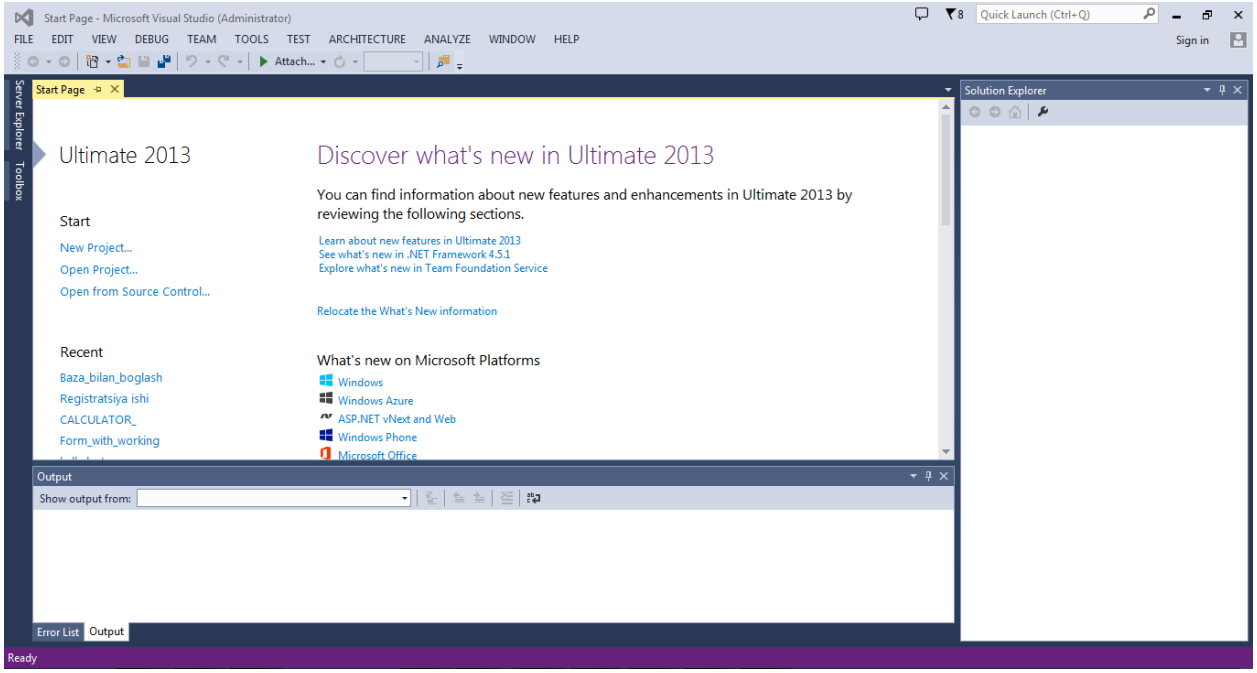
Glossariy

- **Metodlar** – ma'lum bir klass bilan bog'langan funksiyalar hisoblanadi. Bunday funksiyalar sifatida klass nusxasi metodlari yoki klass nusxasi hosil qilinishini talab qilmaydigan statik metodlar (masalan, Console.WriteLine) tushuniladi.
- **Xususiyatlar** – mijoz tomonidan murojaat qilish imkoni mumkin bo'lgan funksiyalar bo'lib, klassning ochiq maydonlariga o'xshash. C# tilida xususiyatlar bilan ishlovchi maxsus read va write sintaksislari mavjud. Xususiyatlar maxsus sintaksisga ega bo'lib, oddiy funksiyadan farq qiladi.
- **Konstruktorlar** – klass nusxasi hosil qilinganda avtomatik chaqriladigan funksiya hisoblanadi. Ushbu funksiyalarning nomi klass nomi bilan ustma-ust tushishi va hech qanday qiymat qaytarmasligi lozim. Konstruktorlar klass nusxasi osil qilinganda maydonlarga boshlang'ich qiymat berishda foydalaniladi.
- **Destruktorlar** – konstruktorlarga o'xshash bo'lib, klass nusxasi xotiradan o'chirilganda avtomatik chaqiriladi. Ular ham klass nomi kabi nomlanib, oldiniga tild belgisi (-) qo'yiladi. Dasturning keraksiz ma'lumotlardan tozalashni CLR bajarishini inobatga olib, qachon destruktor chaqirilishini aytish qiyin. C# tilida destruktorlar kamroq qo'llaniladi
- **indeksatorlar** – ob'ektlarni massiv va kolleksiya kabi indekslash uchun qo'llaniladi.
- Qism dastur – dasturning bir qismi bo'lib, qachonki unga murojat qilinsagina o'sha dastur ishga tushadi
- Void – qism dasturni qiymat qaytarmasligini anglatuvchi tipi hisoblanadi
- Lokal o'zgaruvchi - qism dastur ichida e'lon qilgan yoki blok chegarasida ko'rinish sohasiga ega
- Global o'zgaruvchi - har qanday qism dasturlardan tashqarida e'lon qilinadi va programma bajarilishining oxirigacha amal qiladi. Bunday o'zgaruvchilarga programmadan ihtiyoriy funksiyalardan murojat qilish mumkin
- New - massiv yaratishda bu operatordan foydalaniladi

Amaliy mashg'ulot.

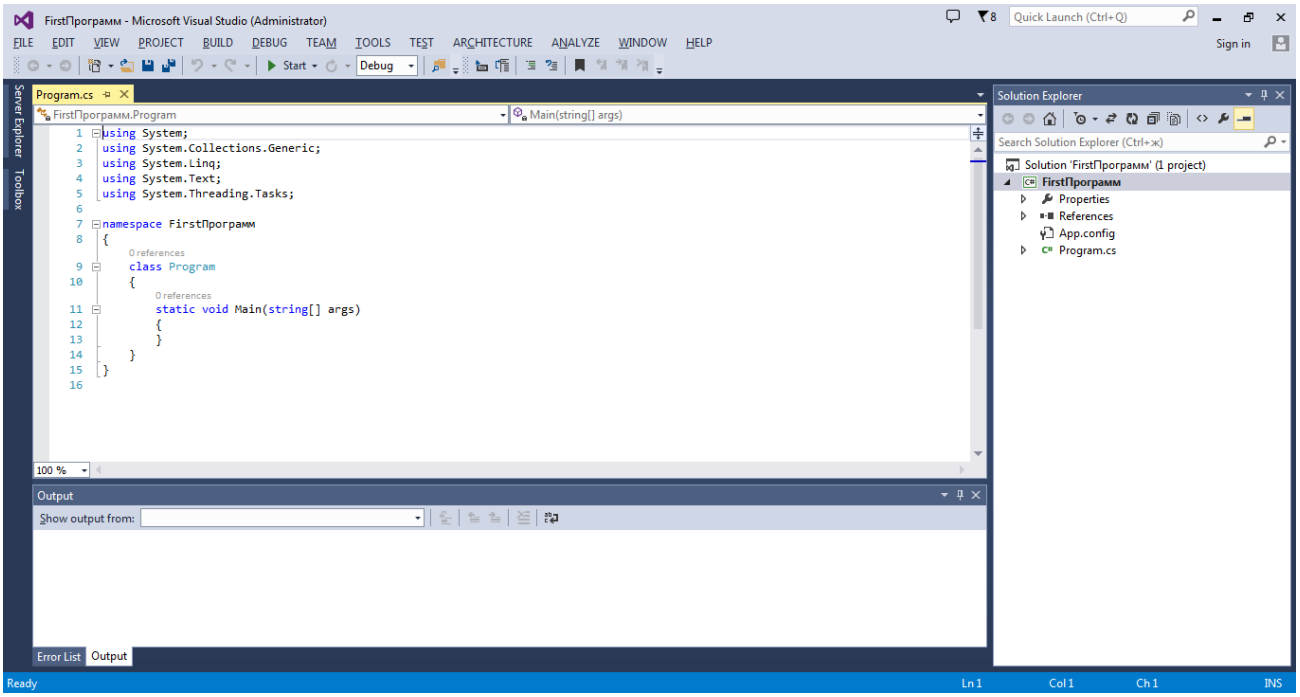
1-misol. Qayta yuklash operatorlari orqali kompleks sonlar ustida amallar bajarish dasturini yarating.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **1.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



1.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **FirstProgramm** kabi kiritib, **OK** tugmasini bosamiz. Natijada **1.2 rasm**da keltirilgan quyidagi oyna shakllantiriladi.



1.2-rasm. Dasturiy kod oynasi

Endi peregruska deb nomlangan klass yaratamiz va u klassni kodi quyidagicha:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace complex
{
   class peregruska
   {
      int a;
      int b;
      public peregruska (int a,int b)
      {
         this.a=a;
         this.b=b;
      }
      //public void show()
      //{
      //    Console.WriteLine(a+b+"i");
      //    //Console.WriteLine(b);

      //}
      public override string ToString()
      {
```

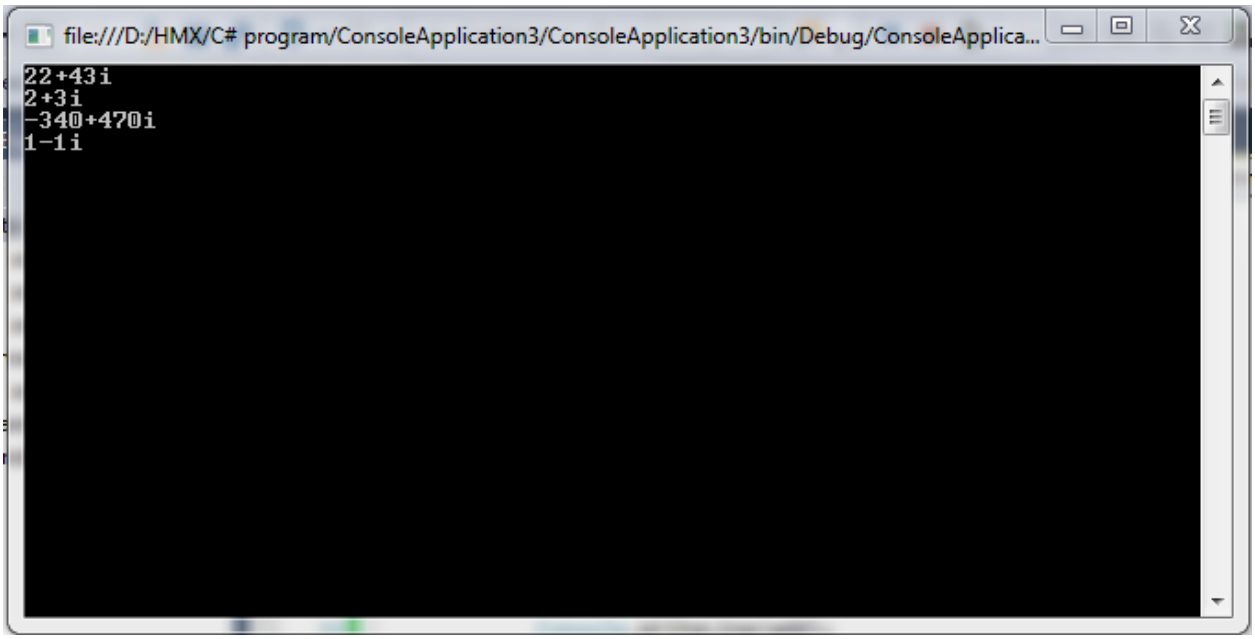
```
        if(b>0)
            return a+"+"+b+"i";
        else
            return a + "" + b + "i";
    }
    public static peregruska operator+(peregruska a1, peregruska a2)
    {
        peregruska c = new peregruska(a1.a, a2.b);
        c.a=a1.a+a2.a;
        c.b=a1.b+a2.b;
        return c;
    }
    public static peregruska operator-(peregruska a1, peregruska a2)
    {
        peregruska c = new peregruska(a1.a, a2.b);
        c.a=a1.a-a2.a;
        c.b=a1.b-a2.b;
        return c;

    }
    public static peregruska operator*(peregruska a1, peregruska a2)
    {
        peregruska c = new peregruska(a1.a, a2.b);
        c.a=a1.a*a2.a-a1.b*a2.b;
        c.b=a1.a*a2.b+a1.b*a2.a;
        return c;
    }
    public static peregruska operator/(peregruska a1, peregruska a2)
    {
        peregruska c = new peregruska(a1.a, a2.b);
        c.a=(a1.a*a2.a-a1.b*a2.b)/(a1.a*a1.a-a2.b*a2.b);
        c.b=(a1.a*a2.b+a1.b*a2.a)/(a1.a*a1.a-a2.b*a2.b);
        return c;
    }

  }
}
```

Endi ushbu klassni Main funksiya orqali chaqiramiz. Quyida chaqirish kodi berilgan.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace complex
```

```
{
   class Program
   {
      static void Main(string[] args)
      {
         peregruska p1 = new peregruska(12,23);
         peregruska p2 = new peregruska(10, 20);

         peregruska add = p1 + p2;
         peregruska m = p1 - p2;
         peregruska n = p1 * p2;
         peregruska k = p1 / p2;

         //p.show();
         Console.WriteLine(add);
         Console.WriteLine(m);
         Console.WriteLine(n);
         Console.WriteLine(k);
         Console.ReadKey();
      }
   }
}
```

Dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. 1.3-rasmda keltirilgan natijaga erishamiz.



1.3-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin.

Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliyot topshiriqlari

1. Komplex sonlar ustida arifmetik amallar bajaradigan KOMPLEX sinfi yaratilib, unda ‘+’, ’-‘, ’*’ va ’/’ amallari qayta yuklansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
2. Berilgan n o’lchamli vector ustida vektorlarni qo’shish, ayirish, skalyar ko’paytirish, hamda vectorni songa ko’pytirish amallarini qayta yuklash bajarilgan VECTOR sinfi aniqlansin. Sinf ob’ektlari ustida ko’rsatilg+an amallar bajarilsin.
3. Berilgan natural n va m o’lchamdagi haqiqiy elementli matrisa uchun MATRISA sinfi yaratilsin va unda matrisani matisaga qo’shish, ayirish, ko’paytirish, hamda matrisani songa ko’paytirish amallari qayta yuklansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
4. Vector yordamida to’plamni hosil qilish amalinini bajaruvchi TUPLAM sinfi yaratilsin. To’plam ustida asosiy amallarni – to’plamga yangi element qo’shish va o’chirish, tuplamlar keshishmasini, birlashmasini, hamda ayirmasi amallari qayta yuklansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
5. Vaqtning sequnt, minut, soat qiymatlari ustida bajariladigan qoshish, ayirish va taqqoslash amallarini qayta yaklaydigan VAQT sinfi aniqlansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
6. Sananing kun, oy, yil qiymatlari ustida bajariladigan qoshish, ayirish va taqqoslash amallarini qayta yaklaydigan SANA sinfi aniqlansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
7. Rasional sonlar ustida, yani surat va mahraj juftligi bilan berilgan sonlar ustida qo’shish, ayirish, kopaytirish taqqoslash amallarini qayta yaklaydigan RATSIONAL sinfi aniqlansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
8. Dekart koordinatasida, tekislikda berilgan nuqta koordinatasini Qutb koordinatasiga va aksincha, Qutbdan koordinatasidan Dekart koordinatasiga otkazuvchi amallarni o’z ichiga olgan QUTB va DEKART sinflari aniqlansin. Sinflar ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
9. AKSLANTIRISH_01 sinfi aniqlansin. Unda haqaqiy sonlar massivini [0,1] segmentga akslantirish operator – funksiya ko’rinishida aniqlansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
10. SATR sinfi aniqlansin va unda nol terminalli satrlar ustida satrga satr qo’shish, satrdagi bir satr ostini ikkinchi satr bilan almashtirish amallari operator funksiya ko’rinishida aniqlansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.
11. STEK sinfi aniqlansin. Unda stek ustidagi barcha amallar operator funksiya sifatida aniqlansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin.

12. INTERVAL sinfi aniqlansin. Unda ‘+’, ’-‘, ’*’ va ’/’ amallari qayta yuklansin. Sinf ob’ektlari ustida ko’rsatilgan amallar bajarilsin

13. Ikki xil ko’rinishdagi haqiqiy turdagi matrisalar berilgan: to’g’riburchakli va bosh diagonalga nisbatan simmetrik bo’lgan kvadrat matrisalar. Matrisalarni xotirada saqlashda qiymati nol bo’lgan elementlar saqlanmasligi kerak. Kvadrat matrisa uchun yana qo’shimcha shart - faqat bosh diagonal va undan yuqorida joylashgan elementlar xotirada saqlanishi kerak. Matrisalar sinflar shajarasi ko’rinishida tavsiflansin. Bunda to’g’riburchakli matrisa uchun TB_MATRISA sinfi va uning vorisi sifatida kvadrat matrisa sinfi KV_MATRISA aniqlansin. Matrisalar ustidagi qo’shish va ko’paytirish amallar qayta yuklanuvchi operator ko’rinishida amalgam oshirilsin.

16. Katta sonlar ustida arifmetik amallarni bajarish. O’nlik sanoq sistemasidagi ikkita a va b butun sonlar satr ko’rinishida berilgan. a+b, a-b, a*b va a/b amallar qayta yuklanuvchi operator ko’rinishida aniqlangan UZUN_SON sinfi aniqlansin.

17. Uzunligi oldindan no’ma`lum bo’lgan binar a va b sonlar ustida arifmetik amallar bajarilsin. Sonlar satr ko’rinishida berilgan. a+b, a-b, a>>n (a razryadlarini o’ngga n pozisyaga surish), a<<n (a razryadlarini o’ngga n pozisyaga surish) va a⊕b (istisnoli qo’shish) amallari qayta yuklanuvchi operator ko’rinishida aniqlangan BINAR_SON sinfi aniqlansin.

18. 0<=a<=11,0<=b<=11 butun sonlar ustida qoshish amali gadvalda berilgan huddi shunday ayirish amalini ham aniqlash mumkin,bunday amallarni matematikada modulyar arifmetika deyishadi. Huddi shu amallar aniqlangan SOAT ARIFMETIKASI SINFI aniqlansin.

1. Eng yaxshi forvardni aniqlang va 5 tadan kam o‘yin o‘ynagan futbolistlar haqida ma’lumotni ekranga chiqaring
2. Barcha fan baholari bo‘yicha o‘rtacha balni aniqlang va o‘rtacha bali 4 dan yuqori bo‘lgan talabalar haqida ma’lumotni ekranga chiqaring.
3. Kamida bir yil oldin sotilgan maxsulotlar sonini aniqlang va ular haqida ma’lumotni ekranga chiqaring.
4. Soni 5 tadan ko‘p bo‘lgan maxsulotlar qanchaligini aniqlang va bular haqida ma’lumotni ekranga chiqaring
5. Mazkur yilda ishlab chiqarilgan barcha maxsulotlarni umumiy narxini aniqlang va bu maxsulotlar haqida ma’lumotni ekranga chiqaring.
6. Umumiy narxi eng baland bo‘lgan maxsulot nomini ekranga chiqaring
7. Fizika fani baholari bo‘yicha o‘rtacha balni, informatika bo‘yicha bahosi 5 bo‘lgan talabalar sonini aniqlang va ular haqida ma’lumotni ekranga chiqaring
8. “Ivanov” tomonidan sotilgan tovarlar sonini aniqlang, ular haqida ma’lumotni ekranga chiqaring va eng yuqori narxdagi tovarni aniqlang.
9. Narxi o‘rtacha narxdan yuqori bo‘lgan maxsulotlar haqida ma’lumotni ekranga chiqaring
10. Betlar soni 150 tadan ko‘p bo‘lgan kitoblar haqida ma’lumotni ekranga chiqaring.
11. Tiraj 10000 nusxadan oshmaydigan kitoblar haqida ma’lumotni ekranga chiqaring
12. Oliy ma’lumotga ega bo‘lmagan, 30 yoshdan yuqori bo‘lgan hodimlar haqida

ma’lumotni ekranga chiqaring

13. Hokkeychilarni o‘rtacha yoshini aniqlang va yoshi 25 dan katta bo‘lganlari haqida ma’lumotni ekranga chiqaring.
14. Tiraj 10 000 dan yuqori bo‘lgan plastinkalar haqida ma’lumotni ekranga chiqaring.
15. AMD firmasi tomonidan ishlab chiqarilgan narxi minimal kompyuterni aniqlang va u haqida barcha ma’lumotni ekranga chiqaring
16. Hokkeychilarni o‘rtacha yoshini aniqlang va yoshi 25 dan kichik bo‘lganlari haqida ma’lumotni ekranga chiqaring.
17. Mazkur yilda ishab chiqarilgan barcha maxsulatlarning o‘rtacha narxini aniqlang va haqida ma’lumotni ekranga chiqaring
18. Maxsulotlarni o‘rtacha narxini va minimal narxdagi maxsulotni aniqlang.
19. Eng kichik hodimni aniqlang va haqida ma’lumotni ekranga chiqaring
20. Matematikani “95” ga topshirgan talabalar familiyasini chop eting va ularni soni nechtaligini aniqlang
21. Omordagi soni eng ko‘p bo‘lgan maxsulotni aniqlang va u haqida barcha ma’lumotni ekranga chiqaring.
22. Operativ hotira hajmi 10 Gbaytdan katta bo‘lgan kompyuterlar soni qanchaligini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
23. Yoshi 19 dan katta bo‘lgan talabalar sonini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
24. Yoshi 60 dan katta bo‘lgan hodimlar sonini aniqlang ular haqida barcha ma’lumotni ekranga chiqaring.
25. Omordagi soni eng qimmat maxsulotni aniqlang va u haqida barcha ma’lumotni ekranga chiqaring.
26. Barcha kompyuterlarning o‘rtacha narxini hisoblang va kompyuterlarni nom va o‘rtacha narxini ekranga chiqaring.
27. Injener – hodimlarni sonini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
28. Talabalarni fizika bo‘yicha o‘rtacha ballarini xisoblang va talabalarni familiyasi, tugilgan sanasi va informatika bo‘yicha baholarini chop eting.
29. Uzog‘i bilan ikki yil oldin chiqarilgan maxsulotlarni sonini aniqlang va ular haqida barcha ma’lumotni ekranga chiqaring.
30. DVD ROM mavjud bo‘lgan kompyuterlarni aniqlang va ular haqida

Testlar

1. Dasturlash tili qanday abstraksiyalarni o‘z ichiga oladi?

e) jarayon abstraksiyasi va ma’lumot abstraksiyasiyalarini
f) jarayon abstraksiyasi
g) ma’lumot abstraksiyasi
h) yagona bstraksiyasi

2. Jarayon abstraksiyasi nima?

a) Qism dasturlarning shakli bo‘lib, xamma dasturlash tillarida asosiy tushuncha xisoblanadi.
b) Qism dastur.

c) Katta dastur.
d) Qism dastur o‘zagi

3. Dasturlashtirilgan 1-kompyuterning nomi qanday?

e) Uning Analitik
f) Babbage Uning Analitik Enginer
g) Babbage
h) Analitik Enginer

4. Qism dasturlarni qanday fundamental turlari bor?

e) jarayon
f) protsedura va funksiyalar
g) jarayon,protsedura va funksiyalar
h) qism dastur, protsedura va funksiyalar

5. Qism dasturlarning qisqacha qanday parametrlari o‘z ichiga oladi?

e) raqamli va rasmiy parametrlar
f) raqamli parametrlar
g) rasmiy parametrlar
h) to‘g‘ri javob berilmagan

6. Qism dasturlar odatda nimalar aks ettiradi?

e) hisoblashlarni
f) parametrlarni
g) hisoblashlarni va parametrlarni
h) sonlarni

7. Kalit so‘z parametrlariga qushimcha qanday beriladi?

e) Ada, Fortan 95+, python
f) Ada
g) Fortan 95+
h) Ada va Fortran

8. Qism dasturlarning qanday alohida kategoriyalari bor?

e) tartibi va vazifalari
f) tartibi
g) vazifalari
h) xossali

9. Lokal o‘zgaruvchilar qanday bo‘lishi mumkin?

e) statik yoki dinamik
f) statik
g) dinamik
h) lokal o‘zgaruvchi ko‘rsatkich bo‘ladi

10. Statik lokal o‘zgarvchilarni dinamik lokal o‘zgaruvchilardan afzalligi nimada

e) joylashtirish va kaytarishga ko‘prok vakt sarflaydi
f) joylashtirish va kaytarishga kamroq vakt sarflaydi
g) afzalligi yo‘q.
h) xotradagi sig‘imi kam

11. Ada tili zamonaviy boshqaruvchi tillar orasida qanday til hisoblanadi?

e) Ada tili zamonaviy tillar orasida funksiyalar har qanday tipli uzgaruvchilarni qaytara oladigan yagona til xisoblanadi.

f) Ada tili zamonaviy tillar orasida funksiyalar faqat butun tipli o‘zgaruvchilarni qaytara oladigan yagona til xisoblanadi.

g) Ada tili zamonaviy tillar orasida funksiyalar o‘zgaruvchilarni qaytara oladigan yagona til xisoblanadi.

h) To‘g‘ri javob berilmagan

12. FORTRAN dasturlash tilida lokal o‘zgaruvchilar saqlanuvchi bloklarga kirish qaysi operator orqali amalga oshiriladi.

e) COMMON.COM

f) COM

g) COMMON

h) CRL

13. SIMULA 67 tilining boshlang‘ich maqsadi nima?

e) Dasturlarni modellashtirish

f) Tizimlarni modellashtirish

g) Funksiyalarni modellashtirish

h) Protseduraviy modellashtirish

14. **Qaysi tilda funksiyalar faqat kiritish rejimida yuboriluvchi rasmiy parametrga** ega.

e) Ada tilida

f) Modula2 tilida

g) Modula1 tilida

h) Modula1 va Modula2 tilida

15. Qaysi tillarda chaqiruvlar metodning kerakli versiyasi bilan dinamik bog‘lanadi.

e) Modula1,Modula 2, Ada

f) Smalltak, Java, Ada95, S++

g) S# , SIMULA 67, Ada

h) S++, Fortran, Ada

16. Quydagi shablonli funksiyada Type qanday vazifani bajaradi.

```
Template <class Type>
Type max (Type first, Type second)
{
return first <  second > frsit: second;
}
```

e) konstruktor

f) parametr

g) metod

h) xossa

17. Ada tilida qaysi tizmlarni parametri kabi qo‘llab bo‘lmaydi.

A) bo‘lmaydi

B) bo‘lmaydi

S) tizimga bog‘liq

18. S# dasturlash tilini ma’lumotlar ba’zasi bilan bog‘lab bo‘ladimi.

e) bo‘ladi

f) bo‘lmaydi
g) mumkun emas
h) faqat oddiy dasturlarda bog‘lanadi

19. **S# tilida kataloglar bilan ishlash uchun nomlar fazosida qaysi kutubxonadan** foydalaniladi.

e) using system IO
f) using system Netfromvork
g) using system form Application
h) using System model

20. S# tilida satrlar bilan ishlash uchun qaysi kalit suzdan foydalaniladi.

e) int
f) float
g) string
h) bool

10-mavzu. Ob’ektga yo‘naltirilgan dasturlash

Reja:

1. Ob’ektlar va klasslar
2. Klasslar, xossalar va holatlar
3. Objects – Instances of Classes
4. Ob’ektlarni yaratish va foydalanish
5. Konstuktorlar tushunchasi.

Bu mavzuda Ob’ektga Yo‘naltirilgan Dasturlashning asosiy tushunchalari bilan tanishiladi – klasslar va ob’ektlar – shuningdek .NET Frameworkining standart biblotekasida klasslarni qanday ishlatishni tushuntiriladi. Keng qo‘llaniladigan ayrim tizim klasslari haqida gapirib o‘tiladi hamda ularning instanslarini (ob’ektlar) qanday yaratish va ishlatishni ko‘rib chiqiladi. Ob’ekt maydonlariga qanday qilib kirishni, konstruktorlarni qanday chaqirishni konstruktorlar va klasslardagi statik maydonlar bilan qanday ishlashni muhokama qilinadi. Oxirida "namespaces" atamasi bilan tanishiladi – ular bizga qanday yordam bera oladi, ularni qanday kiritish va ishlatish mumkin.

Ob’ektlar va klasslar

Keyingi o‘n yil ichida dasturlash va informatika aql bovar qilmas o‘sish va o‘zgarishlarga uchradi. Bu dasturlarning tuzilishi yo‘lini o‘zgartirib yubordi. Ob’ektga yo‘naltirilgan dasturlash (OYD) mana shunday tub o‘zgarishlarni o‘zida ask ettiradi. OOP ning asosiy tamoyillari va tushunchalariga qisqacha ma’lumot beriladi. Birinchi navbatda klasslar va ob’ektlar nima ekanligini tushuntiramiz. Bu ikki atama OOP ning asosiy tushunchalari. Zamonaviy dasturchilar hayotini ulardan ayricha tasavvur qilish qiyin.

Ob’ektga yo‘naltirilgan dasturlash

Ob’ektga Yo‘naltirilgan Dasturlash (OYD) – bu dasturlash paradigmasi bo‘lib, ular ob’ektlarni va ularning o‘zaro aloqasini kompyuter dasturlarini qurish uchun ishlatadi. Shu orqali subektiv sohaning sodda modelini tushunish osonroq bo‘ladi, bu esa dasturchiga haqiqiy hayotdagi muammolarni mantiqan yechish imkonini beradi.

Hozircha OOP ning afzalliklari va maqsadlari borasidagi tafsilotlarni o‘rganmaymiz, shuningdek, klasslar va ob’ektlar ierarxiyasini qurish tamoyillarini ham ko‘rib chiqmaymiz. Biz OOP ning dasturlash texnikasi ko‘pincha inkansulyatsiya, abstraksiya, polimorfizm va vorislik ekanligini ta’kidlab o‘tamiz. Bu texnikalar ushbu bo‘lim maqsadlariga kirmaydi va biz ular haqida keyinroq “Ob’ektga Yo‘naltirilgan Dasturlash tamoyillari” bo‘limida gapiramiz. Hozir esa ob’ektlarga OOP ning asosiy tushunchasi sifatida diqqat qaratamiz.

Ob’ekt nima

Biz ob’ekt tushunchasi bilan OOP kontekstida tanishamiz. Dasturiy ta’minot haqiqiy dunyo ob’ektlari yoki abstrakt tushunchalarni (ular ham ob’ekt sifatida qaraladi) modellashtirishni maqsad qilgan.

Haqiqiy dunyo ob’ektlariga insonlar, mashinalar, narsa-buyumlar, xaridlar, va boshqalarni misol qilish mumkin. Abstrakt tushunchalar ham ob’ekt sohasida bo‘ladi va ularni kompyuter dasturida ishlatish uchun modellashtirishimiz kerak. Abstrakt ob’ektlarga ma’lumot tuzilishi to‘plami, navbat, ro‘yxat va shajaralarni misol qilish mumkin. Ular bu bo‘limda sub’ekt bo‘lmaydi, lekin biz ularni keyingi bo‘limlarda batafsil ko‘rib chiqamiz.

Haqiqiy hayotdan olingan ob’ektlarda (shuningdek abstrakt ob’ektlarda ham) biz ularning ikki xil xarakterli guruhini farqlashimiz mumkin:

- *Holatlar* – bular ob’ekt xarakteristikasini umumiy yoki muayyan momentlarda tasvirlaydigan usulda aniqlashtiradi.
- *Behavior (qoidalar, fe’l atvor)* – bular ob’ekt tomonidan qilinadigan muayyan o‘ziga xos harakatlardir.

Keling, haqiqiy hayotdan ob’ekt misol qilib ko‘ramiz – “it”. Itning holatlariga uning “ismi”, “junining rangi”, “zoti” misol bo‘lishi mumkin, uning *Behavior* (fe’l-atvori)ga esa – “vovullashi”, “o‘tirishi” va “yurishi” kiradi.

OOP da ob’ektlar ma’lumot va ularni qayta ishlash vositalarini birlashtiradi. Ular haqiqiy hayotda ob’ektlarga mos tushadi hamda ma’lumotlar va harakatlarni o‘z ichiga oladi:

- ***Ma’lumot a’zolari* – ob’ektlar o‘zgaruvchanlarida joylashtiriladi, ular holatlarni tasvirlaydi.**
- ***Metodlar* – biz bu haqida batafsil aytib o‘tganmiz. Ular ob’ektlarni qurish vositasidir.**

Klass

Klass ob’ektning abstrakt xarakteristikalarini bildiradi. U ob’ektlarning tuzilishini ta’minlab beradi, yoki ob’ektning tub mohiyatini tasvirlash uchun namuna bo‘ladi. Klasslar OOP ning qurilish bloklaridir va ular bevosita ob’ektlarga aloqador bo‘ladi. Shuningdek, har bir ob’ekt aniq bir klassning ko‘rinishi hisoblanadi.

Biz bunga uning ko‘rinishi bo‘lgan klass va ob’ektni misol qilib keltirishimiz mumkin. Bizda class Dog va ob’ekt Lassie bor, u class Dogning ko‘rinishi hisoblanadi (biz uni Dog turining ob’ekti deb aytamiz). class Dog hamma itlar xarakteristikasini tasvirlasydi, Lassie esa aniq bir it.

Klasslar Odddiy vaziyatlarda ham ularning jihatlari ma’noli bo‘lib, ular odamlarga tushunarli bo‘lgan, odamlar muammo sohasiga yaqin bo‘gani bilan ham dasturchilarning o‘zlari bilan yaqindan tanishmaslar. Misol tariqasida olaylik, class Dog (klassi) jihatlari “RAM” (yoki eng kamida bo‘masligi kerak) bo‘lolmaydi , chunki kontekstdagi bu klassning shunday jihatlarida ma’no yo‘q.

Classes, Attributes (Belgilar) va Behavior (Hatti-harakatlar)

Class predmetning jihatlarini aniqlaydi (belgilarga murojaat qilinganda) va uning xatti-harakatlarini (predmet orqali say-harakatlarni amalga oshirish mumkin) ham. Class belgilari tana qismi (body)da o‘zining turlarini aniqlaydi (bular o‘zgaruvchilar a’zolar-member deb nomlanadi). Predmetning hatti-harakati Classlardagi metodlarning(methods) izohi orqali modellashtirilgan bo‘ladi.

Classning haqiqiy dunyodagi izohidagi namuna orqali ilgarilab ketayotgan tushintirishlarni misol qilib ko‘rsatiladi. Keling **Dog** misolidagi namunaga qaytamiz. Biz Dog klassini modellari haqiqiy predmeti ‘’**Dog**’’ ekanligini izohlab beramiz. Bu klass hamma **Dog** lar uchun atalgan umumiy jihatlarni o‘z ichiga oladi (shuningdek naslini va tuk ranglarini ham) **Dog** hatti-harakatlari uchun namunalarini ham (shuningdek, baqirishni-**Bark(),** o‘tirishni- **Sit()**, sayr qilishni-**Walk()**) . Shu o‘rinda nasl (**breed**) va junini rangi (**furColor**) ning belgilarini va Sayr qilish(), o‘tirish() va Baqirish() metodlari orqali amalga oshira oladigan hatti-harakatlarni belgiladi.

Ob’ektlar - klass Instances (na’munalari)

Aytib o‘tilgandan to hozirgacha biz har bir predmet hozirgi bitta klassning va shu klassning nusxasiga ko‘ra yaratilgan misolni bilamiz. Izohlangan klassning predmetini yaratish (ixtiro qilish) davr deyiladi. Predmet e’lon qilinib o‘tgan vaqt predmet namuna deyiladi.

Har bir predmet maxsus klassning namunasidir. Bu namuna klass belgilari bilan bog‘langan qiymat to‘plami vaziyati orqali o‘ziga xos xususiyatni ko‘rsatib beradi.

2ta narsani o‘z ichiga olgan predmetning shunga o‘xshash hatti-harakati kontekstda: predmetning klassdagi aniqlangan hatti-harakati va hozirgi vaziyati bo‘ladi. Namuna uchun maxsus vaziyat hisoblanadi(predmet), lekin hamma predmet uchun atalgan umumiysi bu hatti-harakat hisoblanadi, hatti harakat yana klass namunasi hamdir.

S# klasslari

Anchadan beri biz OOP ning bir nechta oddiy jihatlarini hisobga olib kelamiz. Zamonaviy dasturlash tillaridan eng muhim bo‘lagi ob’ektga yo‘naltiriltirish hisoblanadi. Ularning har bittasida predmet va klasslar bilan ishlash uchun maxsus xususiyatlar bor. Bu kitobda biz diqqatimizni faqat 1 ta tilga qaratmoqchimiz, ya’ni S# ga. S# dagi OOP ilmini bilish o‘quvchiga foydasi yaxshi bo‘ladi, OYDni amalda bajarishda hech qanday qiyinchilik bo‘miydi. Chunki mana shu OOP dasturlashda boshlang‘ich tushuncha bo‘ladi, zamonaviy dasturlash tillarining barcha virtualligidan foydalangan holda.

S# klasslari

S# dagi klass, “class’’ kalit so‘zi orqali aniqlangan, ayrim kodlangan (code) bir xil predmetlar guruhi (block) dagi uslublar *(Methods)* va a’zo (member) ma’lumotlarining joylashuvi uningdek lassning shaxsini (nomini) aniqlash orqali quyidagilarga ajratilgan.

S# dagi klass lar quyidagi elementalarni o‘z ichiga oladi:

- **Maydon (fields)** a’zo -aniq turdan olingan o‘zgaruvchi.
- **Hususiyat -** bular elementlarning maxsus turlari bo‘lib, klass maydonida uni yozish va olib tashlaganda qo‘shimcha ma’lumot boshqaruvining mahoratini berish orqali maydonning (Maydon) funksionalligi kengayadi.
- **Metodlar** ular ma’lumotning mukammalligini ta’minlashadi.

Classlarga misollar

Biz S# dagi klass misol qilib olmoqchimiz, u ro‘yxatga olingan elementlarni o‘z ichiga oladi. cat (mushuk) klass modellari haqiqiy dunyodagi ‘’ cat’’ predmeti va nomi, rangi xususiyatlariga ega. Class bir nechta maydon, properties va methodsni izpohlaydi, bu haqda keyinroq gaplashamiz. Hozir siz klass ga izoh ko‘rishingiz mumkin (klass izohlarini detalma-detal izohlamoqchimasmiz, biz diqqatni «Class ni izohlash (Defining Classes)» ga qaratmoqchimiz:

```
public class Cat
{
  // Field name
  private string name;
  // Field color
  private string color;
  public string Name
  {
    // Getter of the property "Name"
    get
    {
      return this.name;
    }
    // Setter of the property "Name"
    set
    {
      this.name = value;
```

```
    }
  }
  public string Color
  {
    // Getter of the property "Color"
    get
    {
      return this.color;
    }
    // Setter of the property "Color"
    set
    {
      this.color = value;
    }
  }
  // Default constructor
  public Cat()
  {
    this.name = "Unnamed";
    this.color = "gray";
  }
  // Constructor with parameters
  public Cat(string name, string color)
  {
    this.name = name;
    this.color = color;
  }
  // Method SayMiau
  public void SayMiau()
  {
    Console.WriteLine("Cat {0} said: Miauuuuuu!", name);
  }
}
```

Misol uchun Cat klassi rangni, nom xususiyatlarni aniqlaydi shuningdek nom va rangdagi yashirin maydondagi qiymatlarini saqlaydi. Bundan tashqari 2 ta konstruktor Cat klassining namunasini yaratish uchun aniqlangan hisoblanadi, mos ravishda parametrlarsiz va SayMiau() degan klass metodi bo‘ladi.

Class misolidan keyin aniqlangan quyidagi yo‘lda uni hozir ishlatishimiz mumkin:

```
static void Main()
{
  Cat firstCat = new Cat();
  firstCat.Name = "Tony";
  firstCat.SayMiau();
  Cat secondCat = new Cat("Pepy", "red");
  secondCat.SayMiau();
  Console.WriteLine("Cat {0} is {1}.",
    secondCat.Name, secondCat.Color);
}
```

Agar biz misolni bajarsak, quyidagilarni ishlab chiqa olamiz:

```
Cat Tony said: Miauuuuuu!
Cat Pepy said: Miauuuuuu!
Cat Pepy is Red.
```

Class lardan foydalanish va izohlash uchun oddiy misolni ko‘rdik, va “Ob’ektlardan foydalanish va yaratish’’ bo‘limida bu misollar qanday ishlashini, tqanday qilib tushinish uchun ruxsat bermoqchiligini va ularni metodlarini chaqirishni, qanday qilib ularni xususiyatlaridan foydalanishni va ob’ektlarni (predmetlarni)qanday qilib yaratishni batafsil tushintirib bermoqchimiz.

Classlar sistemasi

System.Console klassining **Console.WriteLine(…)** metodida nomlanishi S# dagi klass sistemasining qo‘llanilishi namunasidir. S# bilan tuzish qolipi uchun standart biblotekalarda aniqlangan klass larni biz klasslar sistemasi deb chaqiramiz (yoki boshqacha dasturlash programmsi deb). Ularni bizning barcha .NET ilovalarimizdan foydalana olishadi (ular S# da maxsus yozib qo‘yilgan bo‘ladi). Shuningdek String, Environment va Math klasslarini misol uchun olamiz.

Bu haqida ham, keyinroq aytib o‘tib ketamiz “Dasturlashga kirish” mavzuidan .NET Framework SDK bilan dasturlash tillari to‘plami (VB.NET va S# ga o‘xshagan) kelishini allaqachon bilib oldik, shuningdek, mobil ilova va GUI, Web ga asoslanganiga o‘xshaganini yaratish, ma’lumot almashish, ma’lumot bazasiga kirish, tarmoqlash, parallel bajarish, klass to‘plamlari, matn almashish, osongina kiritish va chiqarishga o‘xshagan dasturlashda eng oddiy mashqlarni bajarish uchun to‘plovchi va standart klass biblotekasi millionlab klass sistemlarini yetkazib beradi.

Bu Class lardagi mantiqni bajarish muhim hisoblanadi, shunigdek, ularni ichida **encapsulated (yashiringan).** Dasturchi uchun qanday qilib ularni bilamasligi, ular uni qanday bajarishlari muhim hisoblanadi va bu sabab uchun klasslarning asosiy bo‘lagiga ijtimoy sabablari mavjudmas (public (hammaga ma’lum)). Class sistemalari bilan bajarish hammma dasturchilar uchun ham tez-tez ko‘rishga vaqt topilavermaydi. Shunday qilib, yangi mavhum tushunchalar qavatlari OOP da boshlang‘ich vazifalardan biri bo‘lib kashf qilingan.

Biz klasslar sistemasiga keyinroq alohida e’tibor qaratamiz. Hozir esa dasturlarda ob’ektlardan foydalanish va yaratish bilan yaqindan tanishib olamiz.

Ob’ektlarni yaratish va foydalanish

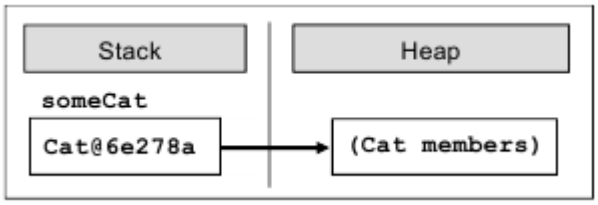
Biz hozirdan boshlab dasturlarimizda ob’ektlardan foydalanish va yaratishga diqqatni qaratmoqchimiz. NET Frameworkdagi eng ko‘p klasslar sistemasi va allaqachon aniqlangan klasslar bilan ishlamoqchimiz. “Classlarni Aniqlash” mavzuida o‘z shaxsiy klasslarimizni aniqlashning maxsus yo‘llari haqida keyinroq to‘xtalamiz.

Ob’ektlarni chiqarish va yaratish

Dasturni bajarish davomida dastlabki aniqlangan klasslardan ob’ektlarni yaratish yangi operator orqali ijro etiladi. Ob’ektning kashf qilinishi qaytaddan bo‘lishi ob’ektning klassi bilan turini muofiqlashtirishdan odatda o‘zgarishini aniqlaydi. Bu vazifada ob’ekt nusxalanmaganligini eslatib o‘tmoqchimiz va endigina kashf qilingan ob’ektga faqat ishora o‘zgaruvchida yozilgan bo‘ladi (uni adressi xotirada). Mana bu yerda uni qanday qilib ishlashi oddiy misolda ko‘rsatilgan:

```
Cat someCat = new Cat();
```

Cat turning someCat o‘zgaruvchilarini biz Cat klassining endigina yaratilgan namuna sifatida belgilaganmiz. To‘planganda qolib ketgan someCat o‘zgaruvchilari va uning qiymati (Cat klassining namunasi) boshqariligan to‘plamda qolib ketadi.



Parametrlarni olish bilan ob’ektlarni yaratish

Hozir yuqorida parametrsiz klass yaratishning misoli yendi parametrlar bilan klassni yaratishning misolini ko‘ramiz

```
Cat someCat = new Cat("Johnny", "brown");
```

S# sinfini turi bilan tanishish

.NET platformasiga tegishli bo‘lsa, u xolda yanada fundamental dastur konstruksiyasi sifatida sinf turi xisoblanadi. Formal sinf bu foydalanuvchi tomonidan aniqlanadigan tur. U berilganlarni maydonidan va holatidan tashkil topgan. Berilganlar maydonining sinfini barchasi birga sinf nusxalarini “Holatini” tasvirlaydi. Ob’ektga yo‘naltirilgan dasturlash tillari S# ga o‘xshash berilganlarni guruxlash xususiyatidan tashkil topgan va shu bilan sinfni aniqlashdagi funksionallik bilan bog‘liq, bu esa real xayot asosida dasturiy ta’minotni madellashtirish imkonini beradi.

Boshlanishida yangi S# Console Application yaratamiz uni SimpleClassExample. Next ataymiz. Keyin loyihaga yangi sinflar faylini (Project → Add Class) qo‘shamiz. Keyin natijalanuvchi dialog oynasida Class belgisini tanlash zarur. Bu esa 5.1 rasmda ko‘rsatilgan va Add qo‘shish tugmasiga bosamiz.

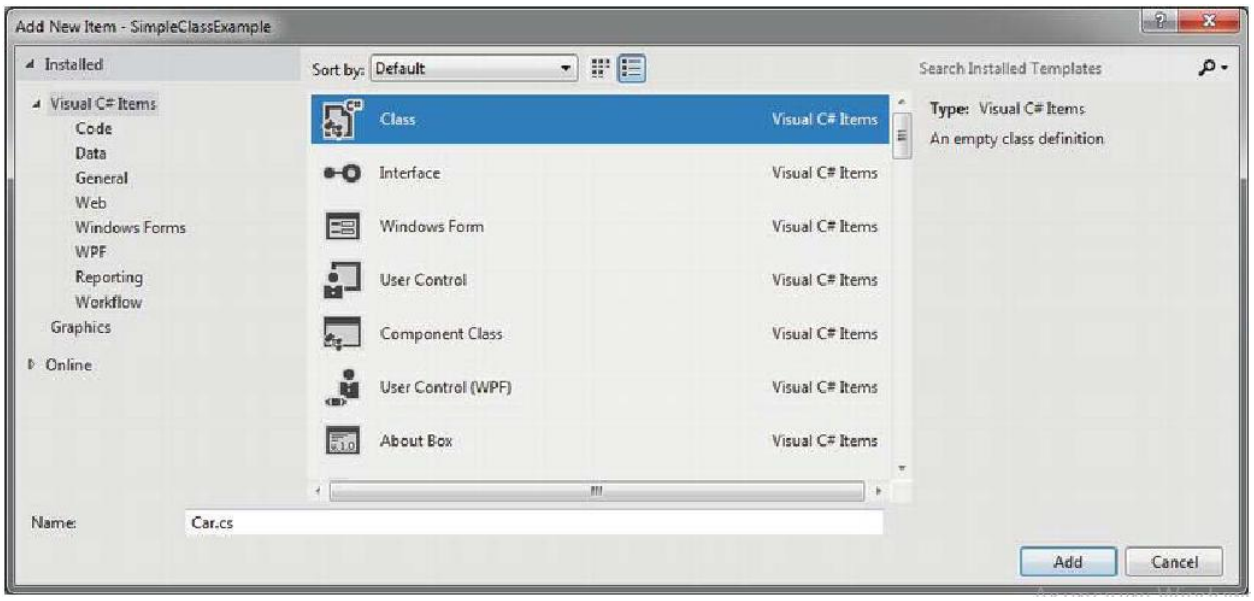
S# da sinfni aniqlash class yordamchi kalit so‘z yordamida amalga oshiriladi. Oddiy mavjud sinflarni e’lon qilish keyingi ko‘rinishda bo‘ladi.

```
class Car
{
}
```

Sinfni turini aniqlab bo‘lgach o‘zgaruvchan xadlar to‘plamini aniqlash lozim. Bu aniqlanganlar uni xatoligini tasdiqlash uchun ishglatiladi.

Masalan siz Car ob’ektlari int turidagi berilganlar maydoniga ega bo‘lishi kerak deb yechishingiz mumkin. Ular xozirgi tezlikni tasvirlaydi va string tipidagi berilganlar maydoni do‘stona avtomobil ismini tasvirlash uchun ishlatiladi. Boshlang‘ich proektiv xolatini xisobga olgan xolda Car sinfi keyingi ko‘rinishda bo‘ladi.

```
class Car
{
   // The 'state' of the Car.
   public string petName;
   public int currSpeed;
}
```



5.1 rasm. S# sinfiga yangi tipni qo‘shish.

Bu o‘zgaruvchan xadlar public mavjudlik modifikatorini ishlatish bilan e’lon qilingan. Ochiq (public) sinfni xadlari mavjuddir. Sizlarga ma’lum bo‘lishi kerakki “ob’ekt” termini berilgan sinf tipini nusxasini tasvirlash uchun xizmat qiladi, u new kalit so‘z yordamida yaratilgan.

O‘zgaruvchan xadlar to‘plamini aniqlagandan so‘ng, loyhalashtirishdagi keyingi qadam sifatida xadlarini yaratish xisoblanadi, ular uning xolatini modellashtiradi. Bunga misol sifatida Car sinfi SpeedUp() ismi bo‘yicha bitta usulni aniqlaydi va yana bittasi ismi bo‘yicha PrintState() sinf kodini keyingi ko‘rinishda modifikatsiyalaydi

```
class Car
{
   // The 'state' of the Car.
   public string petName;
   public int currSpeed;

   // The functionality of the Car.
   public void PrintState()
   {
      Console.WriteLine("{0} is going {1} MPH.", petName, currSpeed);
   }
   public void SpeedUp(int delta)
   {
      currSpeed += delta;
   }
}
```

PrintState() metodi bu qaysidir o‘rinda tashxisli funksiya, u oddiygina Car ob’ektini hozirgi xolatini chiqaradi va u guruxli qatorga chiqadi. SpeedUp() metodi Carni tezligini oshiradi uni kattalikka oshirgan xolda endi Main() metodidagi kodni yangilang u quyida ko‘rsatilgan:

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
  // Allocate and configure a Car object.
  Car myCar = new Car();
  myCar.petName = "Henry";
```

```
  myCar.currSpeed = 10;
  // Speed up the car a few times and print out the
  // new state.
  for (int i = 0; i <= 10; i++)
  {
    myCar.SpeedUp(5);
    myCar.PrintState();
  }
  Console.ReadLine();
}
```

Dasturni ishga tushirib, siz ko‘rishingiz mumkin Car o‘zgaruvchi (myCar) o‘zining xozirgi xolatini butun xayoti davomida ushlab qoladi, u keyingi xulosada ko‘rsatilgan.

```
***** Fun with Class Types *****
Henry is going 15 MPH.
Henry is going 20 MPH.
Henry is going 25 MPH.
Henry is going 30 MPH.
Henry is going 35 MPH.
Henry is going 40 MPH.
Henry is going 45 MPH.
Henry is going 50 MPH.
Henry is going 55 MPH.
Henry is going 60 MPH.
Henry is going 65 MPH.
```

new kalit so‘zi yordamida ob’ektlarni joylashtirish

Kodni oldingi misolda ko‘rsatilgandek, ob’ektlar xotiraga new kalit so‘zini ishlatish bilan joylashtirishlari kerak. Agar new kalit so‘zi ko‘rsatilmasa va sinfni o‘zgaruvchisidan foydalanishga u xolda kompilyatsiya xatoligi kelib chiqadi.

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
  // Error! Forgot to use 'new' to create object!
  Car myCar;
  myCar.petName = "Fred";
}
```

Masalan, Main() keyingi metodi kompilyatsiyalanmaydi. New kaliti so‘zini qo‘llash orqali ob’ektni korrekt yaratish uchun, xotirasida Car ob’ektini joylashtirish va aniqlash mumkin kodni bitta qatorida.

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
  Car myCar = new Car();
  myCar.petName = "Fred";
}
```

Sinfni xotirasini xotirasida alternative sifatida joylashtirish kodni xar hil qatorlarida quyidagicha amalga oshirish mumkin.

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
```

```
  Car myCar;
  myCar = new Car();
  myCar.petName = "Fred";
}
```

Bu yerda kodni birinchi operatori xali yaratilmagan Car tilidagi ob'ektga jo'natmani e'lon qiladi. Jo'natmani oshkor o'zlashtirishi bo'lgandan so'ng xotiradagi ob'ektga ko'rsatadi. Xoxlagan xolatda bu xaqida biz oddiy sinf tipiga ega bo'ldik, u bir nechta berilganlar elementlarini aniqlaydi va ba'zi bir bazali operatsiyalarni xozirgi Car sinfini funksionallarini kengaytirish uchun konstruktorlar ahamiyatini o'rganish lozim.

Konstuktorlar tushunchasi

Dasturchi ob'ektning maydonlarini qiymatini o'zlashtirishni o'ylaydi u bilan ishlashdan oldin hozirgi vaqtda Car tilini petName va currSpeed maydonlarini o'zlashtirishni talab etadi. Xozirgi misol uchun bu unchali muammoli emas chunki berilganlarni aniq elementlarini atiga 2 ta xolos tushunarliki hech kimga bunday sinfni barcha 20 ta elementi uchun 20 ta operatorlarni initsializatsiya qilish yoqmaydi.

Baxtimizga, C# Carda konstruktorlarni mexanizmi qo'llab quvvatlanadi, ular ob'ektni xolatini o'rnatishga ruxsat berishadi. Konstruktor bu - sinfni alohida usuli bo'lib u ob'ektni yaratishda oshkor bo'lmagan xolda chaqiriladi va u new kalit so'zini ishlatish bilan amalga oshiriladi. Biroq "normal" usuldan farqi shundaki unda oshkor konstruktor xech qachon qaytariladigan qiymatga ega bo'lmaydi va doim sinf ismiga identetik nomlanadi.

Standart konstruktorni roli

Har bir S# sinfi standart konstruktor bilan ta'minlanadi. U zarur bo'lgan vaqtda qayta aniqlanishi mumkin ta'rif bo'yicha standart konstruktor xech qachon argumentlarni qabul qilmaydi. Yangi ob'ekt joylashtirilgandan so'ng xotirada standart konstruktor bo'yicha berilganlar maydonlarini kafolatlaydi. Ular mos standart qiymatlarda bo'ladi.

Agar siz bunday standart o'zlashtirish bilan qanoatlansangiz o'zingizga kerakli molikdagi standart konstruktorni yozishingiz mumkin. O'zgartiish maqsadida S# Car sinfi quyida ko'rsatilgan:

```
class Car
{
  // The 'state' of the Car.
  public string petName;
  public int currSpeed;

  // A custom default constructor.
  public Car()
  {
    petName = "Chuck";
    currSpeed = 10;
  }
...
}
```

Berilgan xolatda biz Car ob'ektlarni o'z xayotini Chuck ismidan boshlashga majburlaymiz va soatiga 10 ml tezlik bilan. Bunda Car ob'ektlarini standart qiymatlar bilan keyingi ko'rinishda yaratish mumkin:

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
```

```
  // Invoking the default constructor.
  Car chuck = new Car();

  // Prints "Chuck is going 10 MPH."
  chuck.PrintState();
...
}
```

Maxsus konstruktorlarni aniqlash

Standart konstruktorga o‘xshash sinflarda qo‘shimcha konstruktorlar aniqlanadi. Bunda ob’ektni foydalanuvchi oddiy va kelishilgan ob’ekt xolatini initsiallovchi usul bilan uni yaratilish vaqtida ta’minlanadi. Sar sinfini keyingi o‘zgarishiga e’tibor qarating. U endi butun 3 ta konstruktorni qo‘llab quvvatlaydi.

```
class Car
{
  // The 'state' of the Car.
  public string petName;
  public int currSpeed;

  // A custom default constructor.
  public Car()
  {
    petName = "Chuck";
    currSpeed = 10;
  }

  // Here, currSpeed will receive the
  // default value of an int (zero).
  public Car(string pn)
  {
    petName = pn;
  }

  // Let caller set the full state of the Car.
  public Car(string pn, int cs)
  {
    petName = pn;
    currSpeed = cs;
  }
...
}
```

1 ta konstruktor ikkinchisidan argumentlar miqdori va tipi bilan aniqlanadi. [2]ning 4-mavzuda ko‘rsatilganki, usullarni bir xil ism bilan lekin xar hil miqdorda va argumentlarni tiplari aniqlaydi. **Qayta yuklanishi** deb ataladi. Shunday qilib Car sinfi qayta yuklangan konstruktorga esa istalgan xolatda Car ob’ektlarni endi istalgan uning ochiq konstruktorlarida foydalangan xolda yaratish mumkin.

Masalan:

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
```

```
  // Make a Car called Chuck going 10 MPH.
  Car chuck = new Car();
  chuck.PrintState();
  // Make a Car called Mary going 0 MPH.
  Car mary = new Car("Mary");
  mary.PrintState();
  // Make a Car called Daisy going 75 MPH.
  Car daisy = new Car("Daisy", 75);
  daisy.PrintState();
...
}
```

Konstruktor haqida

Barcha sinflar bekorga (pulsiz) standart konstruktor bilan ta’minlanadi. Shunday qilib agar xozirgi loyihaga yangi sinfni Motorcycle nomi bilan qo‘shsak u keyingi ko‘rinishda aniqlangan:

```
class Motorcycle
{
   public void PopAWheely()
   {
      Console.WriteLine("Yeeeeeee Haaaaaeewww!");
   }
}
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Class Types *****\n");
  Motorcycle mc = new Motorcycle();
  mc.PopAWheely();
...
}
```

Biroq, agar maxsus konstruktor ixtiyoriy miqdordagi parametrlar bilan aniqlangani bilinsa, standart konstruktor jimgina sinfda o‘chib ketadi va kirib bo‘lmas bo‘lib qoladi. Buni bunday qabul qiling agar siz maxsus konstruktorni aniqlamagan bo‘lsangiz, u holda S# kompilyator standart konstruktor sinfi bilan ta’minlanadi.

Agar siz ajoyib konstruktorni aniqlasangiz kompilatorsiz boshqaruvni o‘z qo‘lingizga oldi deb xisoblaysiz.

```
class Motorcycle
{
  public int driverIntensity;

  public void PopAWheely()
  {
    for (int i = 0; i <= driverIntensity; i++)
    {
      Console.WriteLine("Yeeeeeee Haaaaaeewww!");
    }
  }
  // Put back the default constructor, which will
  // set all data members to default vaules.
  public Motorcycle() {}

  // Our custom constructor.
```

```
  public Motorcycle(int intensity)
  {
    driverIntensity = intensity;
  }
}
```

This kalit so‘zining roli

S# tilida This kalit so‘zi mavjud u hozirgi sinfni elementlariga kiritishni ta’minlab beradi. This so‘zini mavjud qo‘llanuvchilarida biri shundaki kontekstni bir qiymatliligini yechish uchun u ham bo‘lishi mumkin. Unda quruvchi parametr sinfi berilganlar maydonidagi ismdek quruvchi parametr xam nomlanganda bo‘ladi. Eng yaxshi xolati nomlash xaqidagi kelishuvda qolganlikdir. U bunday bir qiymatlilikka olib kelmaydi. Shunga qaramasdan bunday this kalit so‘zini ishlatishni ko‘rsatish uchun Motorcycle klassiga yangi string tipidagi maydonni qo‘shamiz u tashuvchining ismini tasvirlaydi. Bundan so‘ng **SetDriverName()** nomli metodni qo‘shamiz u quyidagi ko‘rinishda amalga oshirilgan.

```
class Motorcycle
{
  public int driverIntensity;

  // New members to represent the name of the driver.
  public string name;
  public void SetDriverName(string name)
  {
    name = name;
  }
...
}
```

Visual Studio o‘zgaruvchi o‘zi o‘ziga o‘zlashtirish xaqidagi ogoxlantiruvchi xabarni ko‘rsatadi. buni ko‘rsatish uchun Main()ga SetDriverName() chaqiruvini qo‘shamiz va name maydondan qiymatini chaqiramiz. Aniqlanadiki name maydoni qiymatida bo‘sh qator qoldi.

```
// Make a Motorcycle with a rider named Tiny?
Motorcycle c = new Motorcycle(5);
c.SetDriverName("Tiny");
c.PopAWheely();
Console.WriteLine("Rider name is {0}", c.name); // Prints an empty name value!
```

Muammo shundaki, SetDriverName()ni amalga oshirishda uning o‘zining qiymatini quruvchi parametrga o‘zlashtirishni bajaradi, chunki kompliyator nomi bu yerda o‘zgaruvchiga jo‘natiladi deb o‘ylaydi.

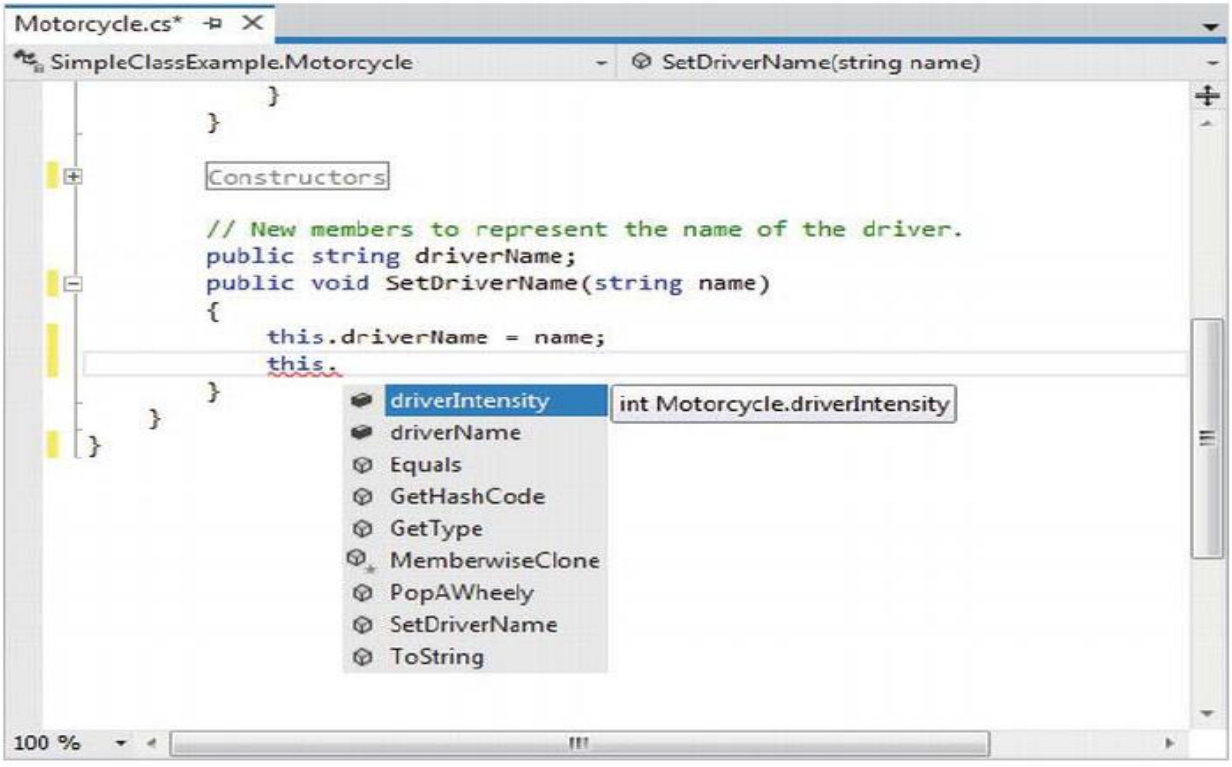
```
public void SetDriverName(string name)
{
  this.name = name;
}
```

Xisobga olingki agar bir qiymatlilik bo‘lmasa u holda siz this kalit so‘zini ishlatishingiz shart emas. Masalan: agar string tipidagi Name berilganlar hadini driverNamega qayta nomlash kerak bo‘lsa, u holda thisni qo‘llanishi majburiy bo‘lmaydi, chunki kontekstni bir qiymatliligi yo‘qoladi.

```
class Motorcycle
{
  public int driverIntensity;
  public string driverName;
```

```
  public void SetDriverName(string name)
  {
    // These two statements are functionally the same.
    driverName = name;
    this.driverName = name;
  }
...
}
```

Bir qiymatli bo‘lmagan holatta thisni ishlatishdagi olingan katta bo‘lmagan yutuq bu kalit so‘zlarni ishlatishda foydali bo‘lishi mumkin, chunki IDE SharpDevelop va Visual Studio muhitlari IntelliSense vositasini qo‘shadi. Qachonki this kiritilsa. Bu agar siz sinfni nomini esdan chiqargan bo‘lsangiz bu juda yaxshi yordam beradi. 5.2 rasmga qarang.



5.2. Rasm. Thisni ishlatish bilan kanstruktorlarni chaqiruv zanjirlarini ko‘rinishi.

This kalit so‘zininig boshqa qo‘llanuvi sinfi loyihalashtirishda tashkil topgan u konstruktorlarni ishlashi yoki konstuktorlangan zanjirlarini qurulishi nomi ostida metodni qo‘llaydi. Bu loyihalashtirish shabloni klass bo‘lmaganda foydalidir, u bir nechta kanstruktorlarni aniqlaydi. Quruvchi argumentlarni kanstruktorlari har xil ish qoidalariga bo‘ysunishni tekshirishadi shunda, konstruktorlar to‘plamini ichidagi aniqlikka tekshiruvini o‘tkazish yuzaga keladi. Motorcycle sinfini keyingi o‘zgartirilgan qayta yuklanuvchini ko‘tib chiqamiz.

```
class Motorcycle
{
  public int driverIntensity;
  public string driverName;
  public Motorcycle() { }
  // Redundent constructor logic!
  public Motorcycle(int intensity)
  {
    if (intensity > 10)
```

```
    {
      intensity = 10;
    }
    driverIntensity = intensity;
  }
  public Motorcycle(int intensity, string name)
  {
    if (intensity > 10)
    {
      intensity = 10;
    }
    driverIntensity = intensity;
    driverName = name;
  }
...
}
```

Bu yerda har bir konstruktorda tekshiruv ishlab chiqariladi, unda quvvat darajasi 10-dan oshmaydi buni hammasi to‘g‘ri va yaxshi bo‘lishiga qaramasdan ikkita konstruktorda ortiqcha kod paydo bo‘ladi. Bu idealdan ancha yiroq, chunki kodni bir nechta joyda o‘zgartirish kerak, qoidalarni o‘zgartirish hamda yuzaga kelgan holatni to‘g‘irlash usullaridan biri Motorcycle sinfdagi usulni aniqlashdan tashkil topgan, u kiruvchi argumentlarni tekshiruvini bajaradi. Agar har bir konstruktor bu usulni maydonlarga qiymatlarini o‘zlashtirishidan oldin chaqirilsa. Biroq bunday yondashuv kodni muzlatishga ruxsat beradi bunda biznes qoidalarini o‘zgartirishda yangilashga to‘g‘ri keladi, endi boshqa ortiqchalik paydo bo‘ladi.

```
class Motorcycle
{
  public int driverIntensity;
  public string driverName;
  // Constructors.
  public Motorcycle() { }
  public Motorcycle(int intensity)
  {
    SetIntensity(intensity);
  }
  public Motorcycle(int intensity, string name)
  {
    SetIntensity(intensity);
    driverName = name;
  }
  public void SetIntensity(int intensity)
  {
    if (intensity > 10)
    {
      intensity = 10;
    }
    driverIntensity = intensity;
  }
...
}
```

Yana tushunarli yondashuv konstruktorni belgilashni nazarda tutadi. U argumentlarni maksimal qiymatini qabul qiladi “master constructor” sifatida qolgan konstruktorlar this kalit so‘zini ishlatishlari mumkin uni asosiy konstruktorga kiruvchi argumentlarini uzatish uchun va zaruratta xohlaganda ham qo‘shimcha parametrlarni yetkazib berish uchun ishlatiladi. Natijada faqatgina butun sinf uchun konstruktorni yagonaligini quvvab-quvatlash haqida xavotir olishga to‘g‘ri keladi, o‘sha vaqtta qolgan konstruktorlardek asosdan bo‘shligicha qoladi.
Quyida Motorcycle sinfini yakuniy amalga oshirilishi keltirilgan. Konstruktorlarni zanjirga bog‘lashda e’tibor bering this kalit so‘zi jismdan tashqarida joylashgan.

```
class Motorcycle
{
  public int driverIntensity;
  public string driverName;
  // Constructor chaining.
  public Motorcycle() {}
  public Motorcycle(int intensity)
    : this(intensity, "") {}
  public Motorcycle(string name)
    : this(0, name) {}
  // This is the 'master' constructor that does all the real work.
  public Motorcycle(int intensity, string name)
  {
    if (intensity > 10)
    {
      intensity = 10;
    }
    driverIntensity = intensity;
    driverName = name;
  }
...
}
```

This kalit so‘zini – konstruktorlar chaqiruvlarini zanjirga bog‘lashda ishlatish unchalik shart emas. Biruq bunday metodni qo‘llash bizga yanada qisqa kodni aniqlash imkonini beradi.
Bunday texnika yordamida masalalarni yechimini qisqartirish mumkin, chunki real ish yagona konstruktorga delegatsiyalanadi, o‘sha vaqtta qolganlaridek unga oddiygina javobgarlikni beradi.

Konstruktor oqimini tavsifi

Oxirida aytish mumkinki, asosiy ajratilgan konstruktorga argumentlarni qayda ishlangan v konstruktor unda chaqiruvchi barcha konstruktor qolgan barcha operatsiyalarni bajarishni davom ettiradi. G‘oyani aniqlash uchun Motorcycle sinfining konstruktorlarini modifikatsiyalaymiz unga Console.WriteLine() chaqiruvini qo‘shgan holda:

```
class Motorcycle
{
  public int driverIntensity;
  public string driverName;
  // Constructor chaining.
  public Motorcycle()
  {
    Console.WriteLine("In default ctor");
  }
  public Motorcycle(int intensity)
```

```
    : this(intensity, "")
  {
    Console.WriteLine("In ctor taking an int");
  }
  public Motorcycle(string name)
    : this(0, name)
  {
    Console.WriteLine("In ctor taking a string");
  }
  // This is the 'master' constructor that does all the real work.
  public Motorcycle(int intensity, string name)
  {
    Console.WriteLine("In master ctor ");
    if (intensity > 10)
    {
      intensity = 10;
    }
    driverIntensity = intensity;
    driverName = name;
  }
...
}
```

Motorcycle ob'ekti misolida Main() metodida quyidagilar kiritiladi:

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with class Types *****\n");

  // Make a Motorcycle.
  Motorcycle c = new Motorcycle(5);
  c.SetDriverName("Tiny");
  c.PopAWheely();
  Console.WriteLine("Rider name is {0}", c.driverName);
  Console.ReadLine();
}
```

Oldingi Main() usulini bajarish natijasida olingan xulosa quyidagi ko'rinishda bo'ladi.

```
***** Fun with class Types *****
In master ctor
In ctor taking an int
Yeeeeeee Haaaaaeewww!
Yeeeeeee Haaaaaeewww!
Yeeeeeee Haaaaaeewww!
Yeeeeeee Haaaaaeewww!
Yeeeeeee Haaaaaeewww!
Yeeeeeee Haaaaaeewww!
Rider name is Tiny
```

Konstruktorlarni oqimi mantig'i quyida tasvirlangan.

- Avvalo konstruktor chaqiruvi hisobiga ob'ekt yaratiladi, u int tipidagi bitta argumentni qabul iladi.
- Konstruktor olingan berilganlarni asosiy konstruktorga uzatadi va qo'shimcha boshlang'ich argumentlarini taqdim etadi.
- Asosiy konstruktor kiruv belgilarini ob'ektni berilganlar maydoniga o'zlashtiradi.

Konstruktorlarni zanjirlarini ko‘rishda ajoyib jihati shundaki, unda bu daturlash shabloni S# ni xohlangan versiyasi bilan ishlaydi va .NET platformasi bilan ham ishlaydi. Biroq agar maqsadli platform sifatida Net tur hisoblansa yoki keyingi versiya hisoblansa, dasturlash masalasini yanada soddalashtirish mumkin, uni majburiy bo‘lmagan argumentlarini ana’naviy konstruktorlar darajalarini alternativ sifatida ishlatish hisobiga bo‘ladi.

Yana bir bor majburiy bo‘lmagan argumentlar haqida

Siz [2] 4-mavzuda majburiy bo‘lmagan va nomlangan argumentlar haqida bildingiz. Eslab ko‘ring majburiy bo‘lmagan argumentlar kiruvchi argumentlar uchun standart qiymatlarni aniqlash imkonini beradi. Agar chaqiruvchi kodga bu standart qiymatlar qanoatlantirsa unda ajoyib qiymatlarni ko‘rsatish shart emas, biroq buni ob’ekt maxsus berilganlari bilan ta’minlash talab etilgan. Motorcycle sinfini keyingi versiyasini ko‘rib chiqamiz u endi ob’ektlarni konstruktorlashni bir nechta imkoniyatlarini ko‘rsatadi.

Nomlangan argumentlarning sintaksisi standart o‘rnatishlarni o‘tkazib yuborishini eslab ko‘ring. Biroq majburiy bo‘lmagan nomlangan argumentlani qo‘llanishi konstruktorlar to‘plamini aniqlashni juda qulay usulidir u berilgan sinf bilan ishlatiladi. Doim esda tutish kerakki bu sintaksis faqatgina .NET 4,0 mavjud hisoblanadi yoki keyingi versiyalarda. Agar sinflarni ko‘rish talab etilsada ular .NET platformasiz bajarilishi lozim. Xohlagan holatta biz berilganlar maydoni bilan sinfni aniqlashimiz mumkin va har xil operatsiyalar bilan, shundayki ular konstruktorlar va metodlar. Endi formal ko‘rinishda statik kalit so‘zini rolini ko‘rib chiqamiz.

Kiruvchi kod SimpleClassExample proekti 5-mavzu katalogida mavjud.

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

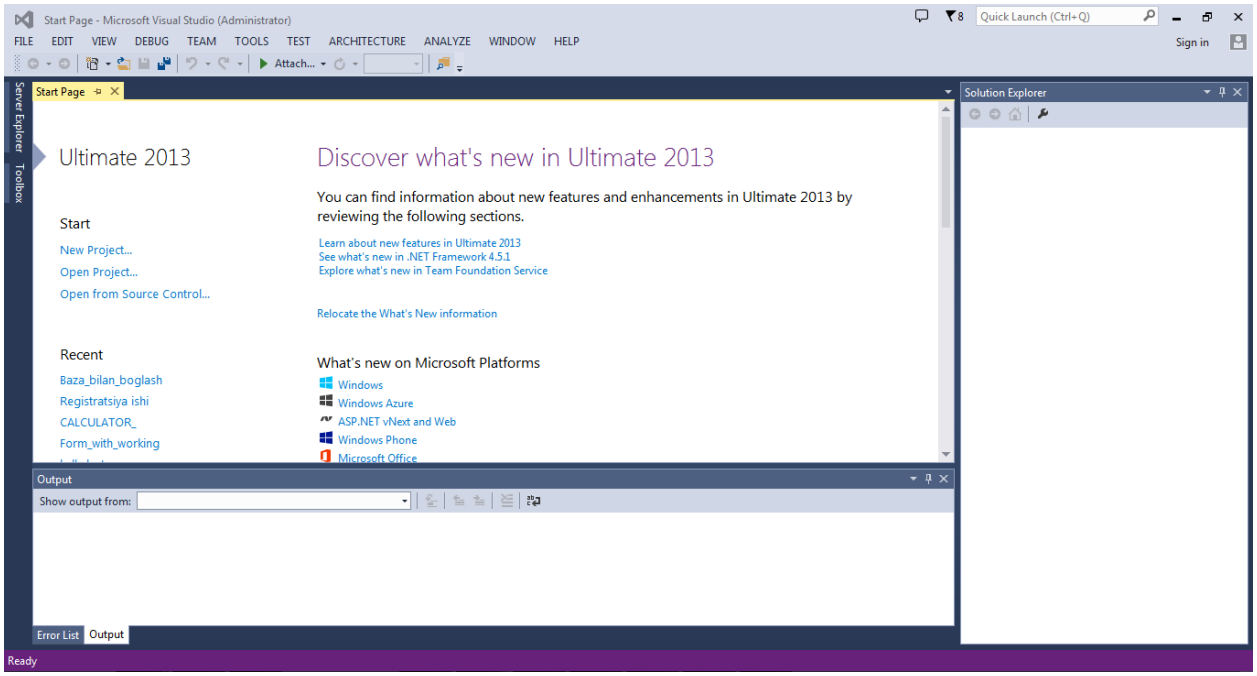
- Ob’ektga yo‘naltirilgan dasturlash (OOP) - dasturlash paradigmasi bo‘lib, ular ob’ektlarni va ularning o‘zaro aloqasini kompyuter dasturlarini qurish uchun ishlatadi. Shu orqali subektiv sohaning sodda modelini tushunish osonroq bo‘ladi, bu esa dasturchiga haqiqiy hayotdagi muammolarni mantiqan yechish imkonini beradi.
- **Klass -** Klasslar ob’ektlarni hosil qilishda ishlatiladigan shablonlar hisoblanadi. Har bir ob’ekt maydonlar va ushbu maydonlar ustida amal bajaruvchi metodlardan iborat. Masalan, mijozni anglatuvchi klass quyidagi maydonlarga ega bo‘lishi mumkin: CustmerID, FirstName, LastName va Address.
- **Klass a’zolari -** klassning ma’lumotlari va funksiyalari
- **Klass ma’lumotlari** - ma’lumotlarini o‘zida saqlovchi klass a’zolari hisoblanadi. Bularga maydonlar, o‘zgarmaslar va hodisalar kiradi

- **Konstruktor** - sinfni alohida usuli bo‘lib u ob’ektni yaratishda oshkor bo‘lmagan xolda chaqiriladi va u new kalit so‘zini ishlatish bilan amalga oshiriladi. Biroq “normal” usuldan farqi shundaki unda oshkor konstruktor xech qachon qaytariladigan qiymatga ega bo‘lmaydi va doim sinf ismiga identitik nomlanadi.**Metodlar** – ma’lum bir klass bilan bog‘langan funksiyalar hisoblanadi. Bunday funksiyalar sifatida klass nusxasi metodlari yoki klass nusxasi hosil qilinishini talab qilmaydigan statik metodlar (masalan, Console.WriteLine) tushuniladi.
- **Xususiyatlar** - mijoz tomonidan murojaat qilish imkoni mumkin bo‘lgan funksiyalar bo‘lib, klassning ochiq maydonlariga o‘xshash. C# tilida xususiyatlar bilan ishlovchi maxsus read va write sintaksislari mavjud. Xususiyatlar maxsus sintaksisga ega bo‘lib, oddiy funksiyadan farq qiladi
- **Konstruktorlar** - klass nusxasi hosil qilinganda avtomatik chaqriladigan funksiya hisoblanadi. Ushbu funksiyalarning nomi klass nomi bilan ustma-ust tushishi va hech qanday qiymat qaytarmasligi lozim. Konstruktorlar klass nusxasi osil qilinganda maydonlarga boshlang‘ich qiymat berishda foydalaniladi.
- **Destruktorlar** - konstruktorlarga o‘xshash bo‘lib, klass nusxasi xotiradan o‘chirilganda avtomatik chaqiriladi. Ular ham klass nomi kabi nomlanib, oldiniga tild belgisi (-) qo‘yiladi. Dasturning keraksiz ma’lumotlardan tozalashni CLR bajarishini inobatga olib, qachon destruktor chaqirilishini aytish qiyin. C# tilida destruktorlar kamroq qo‘llaniladi
- **indeksatorlar** - ob’ektlarni massiv va kolleksiya kabi indekslash uchun qo‘llaniladi.
- Lokal o‘zgaruvchi - qism dastur ichida e’lon qilgan yoki blok chegarasida ko‘rinish sohasiga ega
- ***Global o‘zgaruvchi - har qanday qism dasturlardan tashqarida e’lon qilinadi va programma bajarilishining oxirigacha amal qiladi. Bunday o‘zgaruvchilarga programmadan ihtiyoriy funksiyalardan murojat qilish mumkin***

Amaliy mashg‘ulot

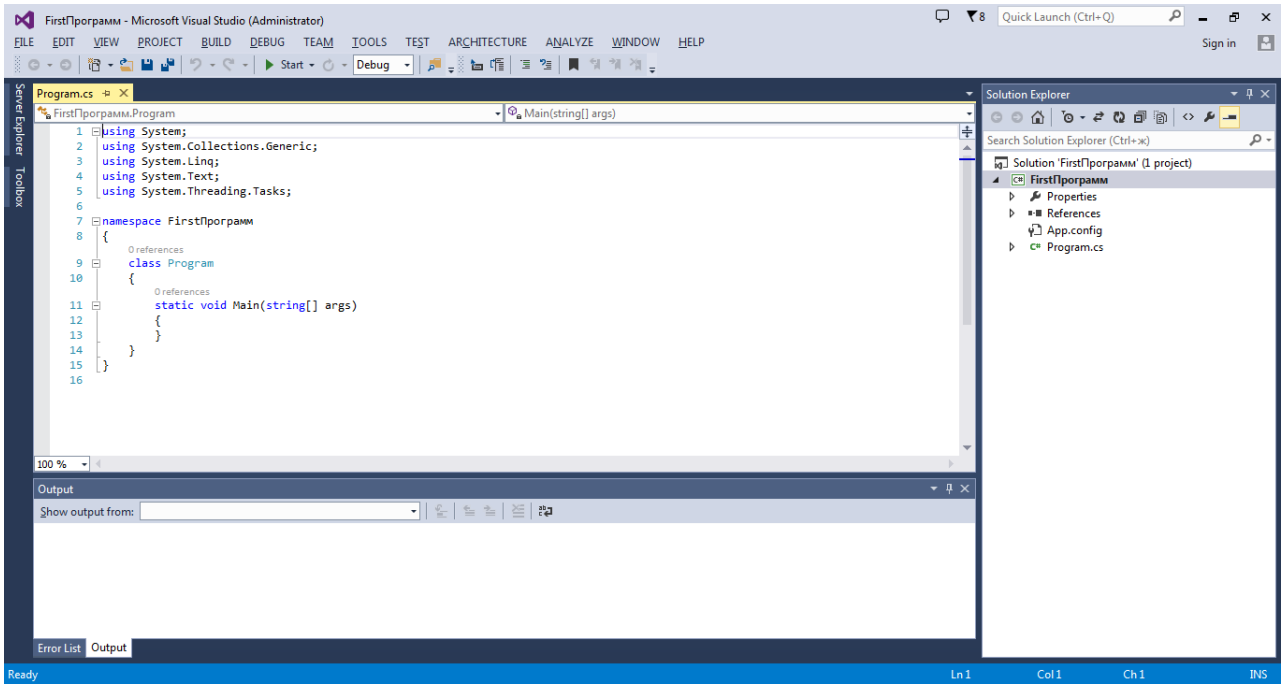
1-misol. To‘rtburchak klassidan foydalanib uchburchak klassini yarating va yuzini qayta nomlang.

Visual Studio 2013 (VS 2013) muhiti o‘rnatilgach, tizim ishga tushiriladi, **1.1 rasm**da keltirilgan foydalanuvchi interfeysi shakllantiriladi.



1.1-rasm. Visual Studio 2013 tizimining boshlang‘ich sahifasi

VS 2012 muhitida biror turdagi dasturiy ta’minotni yaratish uchun **File** menyusidagi **New Project** buyrug‘ini ishga tushirish lozim. Natijada tizimda o‘rnatilgan bir qancha turdagi shablonlar (**Installed Templates**) taqdim qilinadi. Ular orasida **Visual Basic, Visual C#, Visual C++, Visual F#** va boshqalar mavjud. So‘ngra **Visual C#** qismini tanlab, shablonlar (Templates) ichidan **ConsoleApplication** qismini tanlaymiz. Yangi hosil qilinayotgan loyiha nomi (**Name**)ni **FirstProgramm** kabi kiritib, **OK** tugmasini bosamiz. Natijada **1.2 rasm**da keltirilgan quyidagi oyna shakllantiriladi.



1.2-rasm. Dasturiy kod oynasi

Endi Shakl deb nomlangan klass yaratamiz va u klassni kodi quyidagicha:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Vorislik
{
   class Shakl
   {
      public int a;
      public int b;

      public Shakl()
      {
         Console.WriteLine("To'rtburchak elementlari");
      }
      public Shakl(int a, int b)
      {
         this.a = a;
         this.b = b;
      }
      public virtual double Area()
      {
         return a * b;
      }
   }
```

```
}
```

Endi ushbu klassdan Shakl klasssini vorislab Uchburchak klassini hosil qilamiz va yuzini qayta aniqlaymiz.

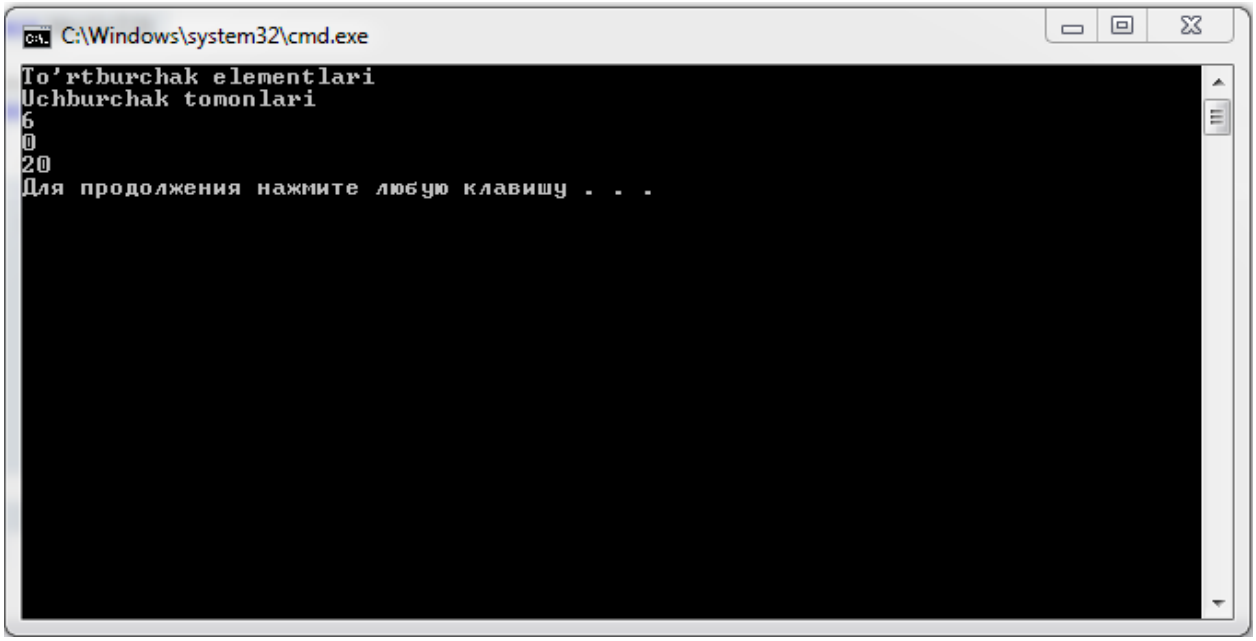
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Vorislik
{
   class Shakl
   {
      public int a;
      public int b;

      public Shakl()
      {
         Console.WriteLine("To'rtburchak elementlari");
      }
      public Shakl(int a, int b)
      {
         this.a = a;
         this.b = b;
      }
      public virtual double Area()
      {
         return a * b;
      }
   }
}
```

Main funksiya orqali chaqiramiz. Quyida chaqirish kodi berilgan.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Vorislik
{
   class Program
   {
      static void Main(string[] args)
      {

         Uchburchak a = new Uchburchak();
         Uchburchak a1 = new Uchburchak(3, 4, 5);
```

```
            Console.WriteLine(a1.Area());
            Shakl a2 = new Shakl(4,5);
            Shakl a3 = new Uchburchak(2, 5, 7);
            Console.WriteLine(a3.Area());
            Console.WriteLine(a2.Area());

        }
    }
}
```

Dasturni **F5** tugmasi orqali ishga tushiramiz va sinovdan o‘tkazamiz. 1.3-rasmda keltirilgan natijaga erishamiz.



1.3-rasm. Konsol oynasi.

Agar dasturni ishlatish jarayonida biror xatolik sodir bo‘lsa, uni **<F10>** yoki **<F11>** funksional tugmalari orqali tekshirib ko‘rish mumkin. Ushbu holda dasturning har bir operatori ketma-ket bajarilib, zarur o‘zgaruvchi qiymatini uning nomiga sichqonchani yaqinlashtirib ko‘rish mumkin. Shuningdek, dasturning zarur tugun nuqtasiga **<F9>** tugmasini bosib yoki **stop** operatoridan foydalanib, ushbu tugun nuqtada o‘zgaruvchilarning qiymatlarini tekshirib ko‘rish imkoniyati mavjud.

Amaliyot topshiriqlari

1. 10 sanoq sistemasida berilgan sonni 2,8 va 16 sanoq sistemasidagi ko’rinishini chop qiluvchi SONNI_CHOP_QILQISH() funksiyasi polimorf funksiya ko’rinishida aniqlansin.
2. Kvadrat va bikvadrat tenglamalar ildizini topadigan KV_ILDIZ() polimorf funksiya aniqlansin.

3. Matritsa uchun MATRITSA taynch sinfi yaratilsin. Uning vorisi sifatida to'gri burchakli va kvadrat matritsalar ustida qo'shish (+), ayirish(-) va ko'paytirish (*) amalani bajaradigan voris sinflar yaratilsin. Kvadrat matritsa elementlari bosh diagonalga nisbatan simmetrik va u uchburchak ko'rinishda berilgan. Yuqoridagi amallarni bajaruvchi funksiyalar polimorf qilib aniqlansin.
4. n o'lchamli fazoda koordinatalari bilan berilgan ikkita nuqtalar orasidagi masofani hisoblaydigan MASOFA tayanch sinfi aniqlansin. Uning vorisi sifatida Dekart va Chebishev fazosida nuqtalar orasidagi masofani hisoblaydigan DEKART va CHEBISHEV sinflari yaratilsin. Nuqtalar orasidagi masofani hisoblaydaydigan funksiya polimorf qilib aniqlansin.
5. Butun sonlarning chiziqli ro'yxatini qayta ishlash uchun RUYXAT tayanch sinfi yaratilsin. Uning vorisi sifatida stek va navbat tuzilmalari uchun STEK va NAVBAT sinflari hosil qilinsin va elementlarni joylash, olish amallarini bajaruvchi funcsiyalar polimorf tarzda aniqlansin.
6. Yulduzcha va Shina topologiyalarida to‘r hosil qilish uchun umumiy harajatni hisoblovchi programma tuzilsin. Bunda umumiy TOR sinfi yasalsin. Sinfda qurilmalar soni n, qurilmalargacha bo‘lgan masofalar a[n], sim narxi q, konnektor narxi p saqlanadi.

   TOR sinfidan YULDUZ va SHINA sinflari yasang. YULDUZ sinfida harajatlar quyidagicha hisoblanadi:

   $$S = \sum_{k=1}^{n} (a_k q + 2p)$$

   SHINA sinfida harajatlar quyidagicha hisoblanadi:

   $$S = \left(\max_{k=1,n} a_n \right) q + np$$

   Berilgan topologiya va o‘lchamlar yordamida S harajatni hisoblaydigan programma tuzilsin.
7. Telefon muloqoti harajatini hisoblovchi TARIF sinfi tuzilsin. TARIF sinfida kiruvchi va chiquvchi qo‘ng‘iroqlar daqiqalari soni saqlansin. TARIF sinfidan UNIVERSAL va PROGRESS sinflari tuzilsin. UNIVERSAL tarifida harajat S=nA+mB formula yordamida hisoblanadi. Bu yerda n - kiruvchi qo‘ng‘iroqlar soni, m – chiquvchi qung‘iroqlar soni, A=0, B=0.03$.

   PROGRESS tarifida

   $S=nA+ m_1B_1+ m_2B_2+ m_3B_3$

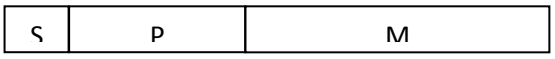
Bu yerda n - kiruvchi qo‘ng‘iroqlar soni, m – chiquvchi qung‘iroqlar soni, A=0.01$, B_1=0.02$, B_2=0.01$, B_3=0.005$

agar m≤50 bo‘lsa, $m_1 = m$, $m_2 = m_3 = 0$.

agar 50<m≤100 bo‘lsa, $m_1 = 50$, m_2 =m-50, $m_3 = 0$.

agar m>100 bo‘lsa, $m_1 = 50$, m_2 =50, m_3 = m-100.

Berilgan tarif, kirish/chiqish qo‘ng‘iroqlari soni yordamida oylik harajatni hisoblovchi programma tuzilsin.

8. Haqiqiy son kompyuter xotirasida 

ko’rinishidagi formatda saqlanadi. Bu erda S –son ishorasini aniqlaydi. Agar son musbat bo’lsa S=0, aks holda S=1 bo’ladi. P-son tartibi (Q-sanoq sistemasi asosining darajasi). M-mantissa (0<M<1). Har qanday ixtiyoriy son $(-1)^S M * Q^P$ ko’rinishiga keltirilib saqlanadi.

Oldindan berilgan format o’lchamlari M, P ko’ra berilgan N sonining ichki formatini 2,10,16- sanoq sistemasida ko’rsatadigan funcsiyalari polimorf bo’lgan FLOAT_FORMAT sinfi va uning avlodlari aniqlansin.

9. To’rtburchak yuzasinin hisoblash uchun TURTBURCHAK tayanch sinfi va ROMB, KVARAT, TUGRITURTBURCHAK, TRAPETSIYA, PARALLELOGRAM avlod sinflari yaratilsin va ularda yuzani hisoblovchi YUZA() funcsiyasi polimorf qilib aniqlansin.

1. 10 sanoq sistemasida berilgan sonni 2,8 va 16 sanoq sistemasidagi ko’rinishini chop qiluvchi SANOQ_SISTEMA sinfi yaratilsin.
2. Komplex sonlar ustida arifmetik amallar bajaradigan KOMPLEX tayanch sinfi yaratilsin. Undan voris sinf sifatida komplex koeffitsientli kvadrat tenglama ildizini topadigan KOMP_KV_TENGLAMA sinfi yaratilsin.
3. Berilgan natural n va m o’lchamdagi haqiqiy elementli matritsa uchun xotiradan joy ajratich, qiymatlarini o’qish va chop qilish amallarini bajaradigan MATRITSA taynch sinfi yaratilsin. Berilgan A va B matritsalar ustida A+B, A-B, A*B, amallarini bajaradigan ARIFM_MATRITSA sinfi MATRITSA sinfidan voris sifatida yaratilsin.
4. Berilgan n o’lchamli fazoda koordinatalari bilan berilgan vectorni tavsiflovchi, y`ni xotirada saqlash, qiymatlarini o’qish va chop qilish amalini bajaruvchi VECTOR tayanch sinfi aniqlansin. Uning vorisi bo’lgan VECTOR_AMAL sinfida vectorlarni qo’shish va ayirish orqali yangi vectorlar hosil qiluvchi, ikkita vectorning skalyar ko’paytmasini, vector uzunligini va ikkita vectorlar orasidagi burchak kosinusi hisoblovchi funksiyalar–

a`zolalar aniqlansin.

5. Ko’phad darajasi va koeffitsientlari bilan berilgan bitta o’zgauvchili ko’phadni xotirada saqlash, qiymat o’qish va chop qilish amalini bajaruvchi KO_PHAD tayanch sinfi yaratilsin. Ushu sinf vorisi sifatida berilgan butun k soni uchun k-tartibli Chebishev ko’phadi koeffitsentlarini hisoblaydigan va berilgan haqiqiy turdagi argumenti uchun ko’phad qiymatini hisoblovchi SHEBISHEV sinfi yaratilsin.
   Chebishev ko'phadlari Tn(x) quyidagi formula bilan aniqlanadi:
   T0(x)=1, T1(x)=x; Tn(x)=2xTn-1 (x) - Tn-2(x), n=2,3,...
6. Kitobning nomi, muallifi, nashriyoti nomi va chop qlingan yili bo’yicha berilganlarni xotirada xotirada saqlash, qiymat o’qish va chop qilish amalini bajaruvchi KITOB taynch sinfi yaratilsin. Uning vorisi bo’lgan UY_KUTUBXONASI sinfida - uy manzili, kutubxona egasi familiya, ismi haqida ma’lumotlar va unda ixtiyoriy sondagi kitoblar bilan ishlash, qandaydir alomati boyocha kitobni izlash (muallif yoki yil bo’yicha), yangi kitobni qo’shish va o’chirish imkoniyatlari bo’lsin.
7. Yon daftarni o’zida aks ettiruvchi YON_DAFTAR sinfi yartilsin. Unda ixtiyoriy sondagi yozuvlar bilan ishlash, qandaydir alomati boyocha yozuvni izlash (familiya, tug’ilgan yili yoki telefon nomeri bo’yicha), yangi yozuvni qo’shish va o’chirish imkoniyatlari bo’lsin.
8. Shaxsning familiyasi va ismi, tug’ilgan yili, jinsi, yashash manzili va telefon nomeri bo’yicha ma’lumotlarni xotirada xotirada saqlash, qiymat o’qish va chop qilish amalini bajaruvchi SHAXS sinfi yaratilsin. Uning vorisi qilib talabalar guruhini tavsivlovchi TALABA_GURUHI sinfi yaratilsin. Unda qo’shimcha ravishda talabaning o’qiydigan guruh nomi, kursi haqida ma’lumotlar bo’lishi, hamda ixtiyoriy sondagi talalbalar bilan ishlash, qandaydir alomati boyocha talabani izlash (familiya, tug’ilgan yili yoki telefon nomeri bo’yicha), yangi yozuvni qo’shish, o’chirish va tartiblash amallari bajarilsin.
9. Vector yordamida to’plamni hosil qilish amalinini bajaruvchi, to’plamda qiymatlarni chop qiluvchi TO‘PLAM sinfi yaratilsin. To’plam ustida asosiy amallarni – to’plamga yangi element qo’shish va o’chirish, tuplamlar keshishmasini, birlashmasini, hamda ayirmasini bajaradigan funksiyalar-a`zolari bo’lgan TO‘PLAM_AMALLARI sinfi TO‘PLAM sinfi vorisi qilib aniqlansin.
10. Berilgan satrni oqimdan o’qish, saqlash, chop qilish amallarini bajaradigan MATN sinfi yaratilsin. Uning vorisi sifatida faqat lotin harfida yozilgan matnni shifrlaydigan va qayta tiklaydigan SHIFRLASH sinfi aniqlansin. Shifrlash uchun lotin harflar alfaviti olinadi. Jarayon matndagi har bir harf bo’yicha chapdan o’ng tomonga ketma-ket ravishda amalga

oshiriladi. Har qadamda alfavitni ko’rsatilgan songa siklik chapga suriladi va matndagi ayni harfni uning hosil bo’lgan alfavitdagi o’rnidagi (indexidagi) harf bilan almashtiriladi. Har bir qadam uchun alfavitni surish soni beriladi. Masalan, 5,3,2,4 sonlari berilgan bo’lsin. Birinchi qadamda (matnning birinchi harfini kodlashda) alfavit 5 marta chapga siklik suriladi va yani paytda qaralayotgan harf uning hosil bo’lgan alfavitdagi o’rni-soni bilan almashtiriladi. Ikkinchi qadamda alfavit yana 3 marta chapga suriladi va hakoza. To’rtinchi qadamdan keyin, ya`ni alfavit 4 marta chapga surilgandan keyin, surilshlar ro’xati takrorlanadi.

11. Stekni amalga oshiruvchi STEK sinfi aniqlansin. Unda stekni tozalash, unga qiymat joylashtirish, o’chirish amallari bajarilsin. Ushbu sinfning vorisi bo’lgan LABIRINT sinfidan labirintdan chiqish masalasini echishda foydalanilsin. Labirint kvadratlardan tashkil topgan matritsa ko’rinishida beriladi. Har bir kvadrat ochiq yoki yopiq bo’ladi. Yopiq kvadratga kirish mumkin emas. Agar kvadrat ochiq bo’lsa uning yon tomonidan kirish mumkin (burchagidan kirish mumkin emas). Har bir kvadrat uning matritsadagi koordinatalari bilan beriladi. Labirintdan chiqich amalga oshirilganda topilgan yo’l chop qilinadi (kvadratlar koordinatalari juftliklarining ketma-ketligi).
12. Shaxmat katagi ikkita belgidan tashkil topgan k katak ko’rinishida berilgan: lotin harfi (a dan h gacha) va raqam (1 dan 8 gacha), masalan a2 yoki g5. Ularni farzin joylashgan shaxmat taxtasidagi katak koordinatalari sifatida qarab, farzin «uradigan» kataklarni «X», boshqa kataklarni «0» bilan belgilab, shaxmat taxtasining ko'rinishi chop qilish imkonin beruvchi SHAXMAT sinfi aniqlansin.
13. Ko’rsatgich asosida yaratilgan butun sonlardan iborat navbatni tavsivlovchi NAVBAT sinfida navbat bilan ishlash, ya'ni elementlar oxiriga qo'shish, boshidan o'chirish («birinchi kelgan – birinchi ketadi») bilan bog’liq quyidagi funksiya aniqlanishi zarur bo'ladi:
    Tozalash()- bo'sh navbatni yaratuvchi (navbatni tozalovchi);
    BushNavbat()- navbatni bo'shligini tekshiruvchi;
    Navbatga() - navbat oxiriga yangi element qo'shuvchi;
    Navbatdan()- navbatdagi birinchi elementni qaytaruvchi va uni navbatdan o’chiruvchi.
    NAVBAT sinfining vorisi sifatida sonlarning umumiy navbatidan sonlarni musbat sonlar navbatiga va musbat bo’lmagan sonlar navbatiga ajratuvchi MUSBAT_MANFIY_NAVBATLAR sinfi aniqlansin.
14. Tassodifiy son hosil qiluvchisini shakllarning yuzasini va hajmini hisoblashda qo’llash mumkin. Shunday usullardan birini Monte-Karlo usuli deyiladi va uning mohiyati

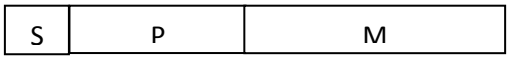
quyidagicha: faraz qilaylik, M shakl birlik kvadrat ichida to'laligicha yotibdi. Tassodifiy son hosil qiluvchisi yordamida birlik kvadrat ichida n son tanlandi va $\nu(n)$ orqali M shakl ichiga tushgan sonlar miqdorini belgilaylik. U holda geometrik ma'lumki, M shakl yuzasi taqriban $\frac{\nu(n)}{n}$ qiymatiga teng bo'ladi va n qanchalik ko'p bo'lsa yuzaning haqiqiy qiymaiga yaqinlashamiz. Tassodiffiy tanlangan nuqta sifatida $(r_1, r_2), (r_3, r_4), \ldots$ koordinatalari bilan berilgan nuqtalarni olish mumkin, bu erda $r_1, r_2, \ldots$ tassodifiy son hosil qiluvchisi tomonidan olingan sonlar. Xuddi shunday, uch o'lchamli fazodagi nuqtalarni (r_1, r_2, r_3) koordinatalari bilan tanlash orqali birlik kub ichidagi shakl hajmini hisoblash mumkin.

Monte-Karlo usulini amalga oshiruvchi MONTE_KARLO sinfi aniqlansin va uning yordamida analiltik ko'rinishi bilan berilgan shakl yuzasi (hajmi) hisoblansin.

15. Bitta qurilmadan ikkinchisiga kanal orqali 0 va 1 raqamlaridan iborat xabar jo'natayotganda halal beruvchi shovqinlar ta'sirida xabar xato qabul qilinishi mumkin (0 o'rniga 1 yoki 1 o'rniga 0). Bunday xatolikni bartaraf qilish yo'llaridan biri – har bir uzatiladigan raqmlarni uch marta takrorlashdir. Masalan, 1,0,1 xabari 1,1,1,0,0,0,1,1,1 ko'rinishida uzatiladi. Qabul qilishda esa har bir uchta raqamlar guruhi unda eng ko'p uchragan raqam bilan almashtiriladi orqali xabar tiklanadi.

    Yuqorida keltirilgan usul bilan berilgan matnni (satrni) «junatadigan» va «qabul» qiladigan amallarni bajaruvchi HABARNI_KODLASH taynch sinfi yaratilsin. Uning vorisi qilib berilgan matnni yuqorida keltirilgan usulda shifrlash orqali jo'natidigan va qabul qiladigan MATN_SHIFRLASH voris sinfi yaratilsin. Bu erda matn belgilarining ASCII kodi asosida jo'natiladi va qabul qilinadi va belgi tiklanadi.

16. Haqiqiy son kompyuter xotirasida



    ko'rinishidagi formatda saqlanadi. Bu erda S –son ishorasini aniqlaydi. Agar son musbat bo'lsa S=0, aks holda S=1 bo'ladi. P-son tartibi (Q-sanoq sistemasi asosining darajasi). M-mantissa (0<M<1). Har qanday ixtiyoriy son $(-1)^S M * Q^P$ ko'rinishiga keltirilib saqlanadi. Oldindan berilgan format o'lchamlari M, P ko'ra berilgan N sonining ichki formatini 2,10,16- sanoq sistemasida ko'rsatadigan FLOAT_FORMAT sinfi aniqlansin.

Testlar

1. Butun tipli bir o‘lchamli massiv qanday e’lon qilanadi?

A) int [] massiv = new of 5
B) int [5] massiv = new int[5]
C) int [] new massiv = int[5]
D) int [] massiv = new int[5]
E) int [] massiv = new int 5

2. Dasturni ishga tushirganda ekranda qanday natija xosil bo‘ladi?

```
using System;
class Test {
public static void Main() {
int[,] a={{-3,6,5},{-1,2,8},{3,1,-4}};
int p = 0;
foreach (int x in a)
{
if (x % 2 != 0) { p = p + x; }
} Console.WriteLine(p);
} }
```

A) 4
B) 5
C)16
D)9
E)12

3. Dasturni ishga tushirganda ekranda qanday natija xosil bo‘ladi?

```
using System;
class Test {
   public static void Main()
   { int[] mas1=new int[6];
      int[,] mas2=new int[3,3];
      int s, p; s = 0; p = 1;
      for (int i = 1; i < mas1.Length; i++)  s = s + 2;
for (int i = 1; i < mas2.Length; i++) p=p*2;
      Console.WriteLine(s);
      Console.WriteLine(p);
} }
```

A)	8	B)	12	C)	10	D)	12	E)	10
	128		256		4		8		256

4. Satrni quyidagilardan qaysisi noto‘g‘ri e’lon qilingan?

A) string str=”Bu satr”
B) char[] ch={‘s’, ‘a’, ‘t’. ‘r’};
String str=new string(ch);
C) string[] str={“bu”, “satr”}
D) string str=new string[5]
E) string str;

5. Agar str1=”satr”, str2=”ikkinchi satr” bo‘lsa str2.IndexOf(str1), str2.LastIndexOf(str1), str2.Substring(3, 4) larning qiymati mos ravishda nimaga teng?

A) 9, 4, inch
B) 9, 9, kinc
C) 9, 9, inch
D) 8, 8,.kinc
E) -1, -1, inch

6. Bitli operatorlar qaysi tiplar bilan ishlaydi?

A) butun sonli
B) mantiqiy
C)ixtiyoriy
D) belgili(char)
E) sonli

7. S# dagi murojat modifikatorlari qaysilar?

A) public, static, private
B) public, Main, internal, private
C) public, private
D) private, internal, protected
E) public, private, internal, protected

8. Quyidagi dasturda xatosi qaysi qatorda ?

```
using System;
class Test{
   private int a;
   public int b;
   public int ab(int i, int j){
      a=i*a;
      b=j*b;
      return a * b;}}
class tek{
   public static void Main(){
      Test ob = new Test();
      int s;
      ob.a = 2;  ob.b = 3; s=ob.ab(3, 4);
      Console.WriteLine(s); }}
```

A) 3
B) 11
C)13
D)14
E)6

9. Dasturni ishga tushirganda ekranda qanday natija xosil bo‘ladi?

```
using System;
class Test {
int x,y;
```

```
public Test(int i, int j) {
x=i; y=j;}
public void nom1(Test ob) {
if (x<ob.x) x=ob.x;
if (y<ob.y) y=ob.y;
x=-x; y=-y;}
public void chiq() {
Console.Write(x); Console.WriteLine(y);}
public static void Main(){
Test ob1=new Test(-5, 4);
Test ob2=new Test(-2, 3);
ob1.nom1(ob2); ob1.chiq(); }}
```

A) 5, -4
B) 4, 3
C) -4, -3
D) 2, -4
E) 2, -3

10. Dasturni ishga tushirganda ekranda qanday natija xosil bo‘ladi?

```
using System;
class Test{
public void Mref(ref int x, ref int y){
      x = x + 2; y = y + x;   }
   public static void Main(){
      Test ob = new Test();
      int a, b; a = 0; b = 0;
      for (int i = 1; i <= 5; i++) ob.Mref(ref a, ref b);
      Console.Write(a +",");
      Console.WriteLine(b);
   }}
```

A) 0, 0
B) 2, 2
C) 10, 30
D) 8, 25
E) xatolik beradi

11. Quyidagi dasturda xatosi qaysi qatorda ?

```
using System;
class Test {
public void MTest() {
Console.WriteLine("Harakat boshlanmadi");}
public int MTest(int t, int v) {
Console.Write(t+" soat vaqtda "+v+" tezlik bilan: ");
return v*t;}
public double MTest(int t, int v, int a) {
Console.Write(t+" soat vaqtda "+v+" boshlangich tezlik bilan "+a+" tezlanishda: ");
return v*t+a*t*t/2;}
```

```
public static void Main(){
Test ob1=new Test();
int yul;
ob1.MTest();
yul=ob1.MTest(2,10); Console.WriteLine(yul+" yulni yurdi");
yul=ob1.MTest(2,10,2); Console.WriteLine(yul+" yulni yurdi");}}
```

A) 10
B)15
C) 23
D) xatolik bermaydi
E) 17

12. b1=1; b2=-1; bk=bk-2+bk-1 k=3,4,… ko‘rinishdagi ketma ketlikni xosil qilish rekursiya funksiyasi

A)
```
public void recurs(int k) {
int b;
if (k==1) return 1; if (k==2) return -1;
b=recurs(k-2)+ recurs(k-1);
return b;}
```

B)
```
public int recurs(int k) {
int b;
if (k==1) return 1; if (k==2) return -1;
b=recurs(k-2)+ recurs(k-1);
return b;}
```

C)
```
public int recurs(int k) {
int b;
if (k==1) return 1;
else if (k==2) return -1;
else b=recurs(k-2)+ recurs(k-1);
return b;}
```

D)
```
public int recurs(int k) {
int b;
if (k==1) return 1;
else if (k==2) return -1;
b=recurs(k-2)+ recurs(k-1);
return b;}
```

E)
```
public int recurs(int k) {
int b;
if (k==1) return 1;
else return -1;
b=recurs(k-2)+ recurs(k-1);
return b;}
```

13. Quyidagi dasturda xatosi qaysi qatorda ?

```
class Test {
public int x;
public static int mymeth1(Test ob){
return ob.x/5;}
public static void mymeth2(){
x=30; Console.Write(mymeth1(x));}}
```

A) 5
B) 4
C) 6
D) 3
E) 2

14. Dasturni ishga tushirganda ekranda qanday natija xosil bo‘ladi?

```
using System;
class Test {
int x,y;
public Test(){x=y=0;}
public Test(int i, int j ){x=i; y=j;}
public static Test operator +(Test ob1, Test ob2){
Test nat=new Test();
nat.x=-(ob1.x+ob2.y);
nat.y=-(ob1.y+ob2.x);
return nat;}
public void kur(){
Console.WriteLine(x+ ", " +y);}}
class Tek{
public static void Main() {
Test a=new Test(-2, 4);
Test b=new Test(5, -1);
Test c=new Test();
c=a+b;
c.kur();}}
```

A) 3, -9
B) -3, -3
C) 3, 3
D) -2, -4
E) xatolik beradi

15. Dasturni ishga tushirganda ekranda qanday natija xosil bo‘ladi?

```
using System;
class test {
public bool Mout(int x, int y, out int t, out int z){
int i;
int max;
if (x>y) max=x; else max=y;
bool f=true; t=1; z=1;
for (i=2; i<=max/2+1; i++){
```

```
if((y%i==0)&(x%i==0)){
if (f) {t=i; f=false;} z=i;}}
if (t!=1) return true; else return false;}}
class tek
{
   public static void Main()
   {
      test ob = new test();
      int p, q;
      if (ob.Mout(54, 36, out p, out q))
      {
         Console.Write(p+”,”);
         Console.WriteLine(q);
      }
   }
}
```

A) 3, 9
B) 4, 12
C) 1, 1
D) 2, 18
E) xatolik beradi

11-Mavzu. Xatoliklar bilan ishlash

Bu mavzu ko‘pchilik zamonaviy dasturlarning bog‘langan ikki qismi uchun dasturlash tili ta’minotini muhokama qiladi: Istisno boshqaruv va hodisa boshqaruv. Istisno boshqaruv ham hodisa boshqaruv ham oldindan hal qilib bo‘lmaydigan vaqtda sodir bo‘la oladi va ikkalasi ham maxsus til konstruksiyasi va jarayoni bilan eng yaxshi boshqariladi. Bu konstruksiyalar va jarayonlarning ba’zilari, misol uchun – istisno va hodisa boshqaruv uchun o‘xshash.

Bu dastlab istisno boshqaruvning fundamental konsepsiyasini tasvirlaydi, qaysiki qattiq va yumshoq disk tekshiruv istisnolari, istisno boshqaruvi va istisnolarni o‘stirish. Keyin istisno boshqaruvi dizayni tanishtiriladi va muhokama qilinadi qaysiki istisno boshqaruviga istisnolarning ustini yuklash, davomiylik, **default** boshqaruv istisno ishlamaslik o‘z ichiga oladi. Bu mavzu 3 ta dasturlash tilining istisno boshqaruv xususiyatlatining shakllanishi va tasvirlanishini ergashtiradi: **Ada, C++** va **Java**.

Bu mavzuning ikkinchi qismi hodisa boshqaruvi haqida. Biz dastlab hodisa boshqaruvini asosiy konsepsiyasiga kirishni namoyish qilamiz. Bu **Java** va **S#** ning hodisa boshqaruv bilimining muhokamasi ko‘rib chiqamiz.

Kirish

Ko‘pchilik kompyuter apparat tizimlari muayyan haqiqiy son (**floating-point overflow**)ga o‘xshagan **run time** holatini aniqlash imkoniga ega. Ilk dasturlash tillari foydalanuvchi dasturi na aniqlay oladiga na xatolarni tuzata oladigan yo‘lda ishlaydi va dizaynlashtiradi. Bu tillarda xatoga o‘xshagan holatlar dasturni tugatishga va operasion tizimga o‘tkazilishiga sabab bo‘ladi. **Run time error** ga nisbatan odatiy operasion tizim diagnostic xatlarni ko‘rsatish qaysiki ma’noli, foydali va yuqori darajada himoyalangan. Xabarni ko‘rgandan keyin dastur tugatiladi.

Kirish va chiqish vaziyatida, bunga qaramay, vaziyat biroz turlicha. Misol uchun, **Fortran** o‘qish operatori (**Read** ifoda (**statement**)) ga kirish xatolari va fayl halatini tugatish (**end of file** shart (**condition**))ni ushlab qoladi, ikkalasi ham kiritish qurilmasini texnik ta’minoti (kirish (**Input**) **device** texnik ta’minot) tomonidan aniqlanadi. Ikkala vaziyatda ham o‘qish operatori (**The Read** ifoda (**statement**)) vaziyat bilan shug‘ullanadigan foydalanuvchi dasturlardagi bazi ifoda (**statement**) yorliqlarini maxsuslashtiradi. Faylni tugashi vaziyatida shart (**condition**) har doim xato hisoblanmasligi aniq. Ko‘p holatlarda jarayonlarning bir turi boshlangan

boshqasi tugagan. Kirish (**Input**) jarayoni xatosiga o‘xshash doimiy xato bo‘lgan xodisalar va faylni tugashi o‘rtasidagi aniq farqga qaramasdan, **Fortrain** ikkala jarayonni bir xil mehanizm bilan bajaradi. Quyidagi **Fortrain** mehanizmiga qarang:

Read (Unit=5, Fmt=1000, Err=100, End=999) Weight

The Err clause o‘qish operasiyasida sodir bo‘ladigan 100 **if an error** ni yorliqlaydigan ifoda (**statement**) ni o‘tkazilishini nazorat qiladi. **The End clause the end of file**ni nazorat qiladigan o‘qish tizimidagi 999 **if** ni yorliqlashtiradigan ifoda (**statement**)ga o‘tkazishni nazorat qiladi. Shuning uchun, **Fortrain** kirish (**Input**) xatolik uchun ham faylni tugatish uchun ham oddiy tanalardan foydalanadi.

Bu yerda texnik ta’minot tomonidan aniqlanmaydigan, lekin kompilyator tomonidan yaratilgan kod orqali aniqlanadigan jiddiy xatolar kategoriyalari bor. Misol uchun, **array subscribe range errors** hech qachon texnik ta’minot tomonidan aniqlanmaydi, lekin ular dasturda muammoga uchramaguncha ko‘rinmaydigan jiddiy xatolarga boshlaydi.

Detection of subscript range errors ba’zida til dizayni tomonidan talab qilinadi. Masalan, **Java complers** odatda har bir **subscript expression** ning to‘g‘riligini tekshiradigan kodlarni yaratadi (Ular **subscript expression range value** dan tashqari mavjud bo‘lmagan vaqtda aniqlanadigan kodlarni yaratolmaydi). S da **subscribe range** tekshirilmaydi chunki tekshirishning qiymati aniqlashning foydasiga arzishiga ishonishmaydi.

Ko‘pchilik zamonaviy tillar dizayneri aniq **run-time-error**ning standart yo‘lida harakat qiladigan dasturlarga ruhsat beradigan mehanizmlarni o‘z ichiga oladi. Dasturlar bu hodisalarga qarshi harakat qilish uchun aniq hodisalar texnik ta’minot yoki **system** dasturiy ta’minot (**software**)tomonidan xodisalar aniqlanganda bildirilishi mumkin. Bu mehanizmlar **exeption handling** deb nomlanadi.

Ehtimol eng haqiqiy sabab **exeption handling** ni o‘z ichiga olmaydigan bazi tillar tilga qoshimchadir.

Biz texnik ta’minot tomonidan aniqlangan xatolarni ko‘rib chiqamiz xuddi **disk read error** ga va **unusual** shart (condition)ga va **end-of-file**ga o‘xshash. Biz qo‘shimcha dasturiy ta’minot (software) detectible bo‘lgan notabiiy holat yoki xatolarni o‘z ichiga oladigan istisnolar konsepsiyasiyalarni kengaytiradi (dasturiy ta’minot (**software**) tarjimon yoki foydalanuvchi kodi orqali). Shularga ko‘ra biz texnik

ta’minot yoki dasturiy ta’minot (**software**) tomonidan aniqlanadigan va maxsus jarayonni talab qiladigan har qanday noodatiy holat, **erroneous** yoki yo‘q bo‘lish istisnolarini aniqlashtiramiz.

Istisnolar xatoga uchraganda talab qilinishi mumkin bo‘lgan maxsus jarayon **exeption handling** deb nomlanadi. Bu jarayon **exeption** ishlab chiquvchi deb nomlangan **code unit** yoki segment orqali bajariladi. **Exeption** unga bogliq xodisa sodir bo‘lganda ***raised*** bo‘ladi. Ba’zi S ga asoslangan tillarda **exeptions *raised*** dan ko‘ra ***thrown*** deb aytiladi. **Detection of end-of-file** dearli har doim ba’zi maxsus dastur harakatlarini talab qiladi. Lekin aniq o‘sha harakat **array index range error exeption** uchun mos bo‘lmaydi. Bazi vaziyatda harakat xato habarlarni shakllanishi va dasturni tartibli tugashidir.

Ba’zi hollarda, u aniq texnik ta’minot-**detectable exceptions**ni e’tiborsiz qoldirishga moyil bo‘lishi mumkin, masalan, bir vaqt uchun bo‘linish. Bu harakat **exeption** ni to‘xtatish orqali bajariladi. Ishdan to‘xtagan **exeption** keyinroq yana ishga tushishi mumkin.

Tildagi ajratilgan yoki maxsus **exeption-handling** xususiyatlarining yo‘qligi **the handling of user-defined**, dasturiy ta’minot (**software**)-**detected exceptions** ni oldini olmaydi. Programma **unit**ni aniqlashtiradigan **exeption** tez-tez **unit**ning chaqiruvchisi yoki **invoker**ni bajaradi. Bir bo‘lishi mumkin bo‘lgan dizayn status o‘zgaruvchisi sifatida foydalaniladigan yordamchi parameterni jo‘natishdir. Status o‘zgaruvchanligi **computing**ning normalligi yoki to‘g‘riligiga ko‘ra **subprogram** deb nomlanadigan **valuen** ni boshqaradi. Tasodifan nomlangan **unit**dan qaytishda **caller status** o‘zgaruvchanligini tekshiradi. Agar **value exeption** sodir bo‘lganini ko‘rsak, **calling unit**da tegishli bo‘lishi mumkin bo‘lgan ishlab chiquvchi harakat qila oladi. S standart kutubxona funksiyasining ko‘pchiligi bu **approach**larning variantlaridan foydalanadi: Qaytish **value**si **error indicator** sifatida foydalaniladi.

Uchinchi ehtimol nomi **unit** deb nomlangan parametr sifatida o‘tgan ajratilgan s**ubprogram** sifatiga aniqlashtiradigan ishlab chiquvchi mavjud. Bu vaziyatda ishlab chiquvchi **subprogram caller** tomonidan ta’minlanadi, lekin nomlangan **unit exception raised** bo‘lganda ishlab chiquvchini chorlaydi. Bu **approach** bilan bog‘liq bir muammo shuki har bir chaqiruvli ishlab chiquvchi **subprogram**ni parameter sifatida zarur yoki zarurmas kabi ishlab chiquvchi **subprogram**ni oladigan har bi **subprogram**ni chaqiradi. Bunga qo‘shimcha tarzda **exeption** ning har xil turlari bilan

shug‘ullanish bir nechta turli ishlab chiquvchi routines kodlarni tugatilishi, o‘tkazilishi zarur.

Exeption handlingni tilga aylantirishning bir nechta aniq afzalliklari bor. Birinchidan, **exeption handling**siz kodlar dasturni tartibga sola oladigan xato vaziyatlar to‘g‘rilashni talab qiladi. Masalan, taxminiy **subprogram mat** deb nomlangan matrix elementlarni 10 ta **reference**ni o‘z ichiga oladigan **expression**larni qamrab oladi va ulardan xoxlagan birida **index out of range error** bo‘lishi mumkin. Qoshimcha taxminlar da til **index range checking**ni talab qilmaydi. **Built-in index ranging checking**siz bu operasiyalaning har biri bo‘lishi mumkin bo‘lgan **index range error**larni topadigan kodlar tomonidan amalga oshirilishi mumkin. Masalan, quyidagi 10 qator, 20 ustundan iborat mat elementiga **reference**ga qarang:

```
if (row >= 0 && row < 10 && col >= 0 && col < 20)
sum += mat[row][col];
else
System.out.println("Index range error on mat, row = " +
row + " col = " + col);
```

Tilda **exception handling** ishtiroki har bir **array element source** dasturini ma’qullashi, qisqartirishi va soddalashtirishidan avval tekshirishlar uchun mashina kodi joylashtiruvchi **complier**ga ruhsat beradi.

Exception handling uchun til ta’minotining boshqa ustunligi exeption propagationning natijasida kelib chiqqan. **Exception propagation** bir dastur uniti ichida dinamik va statik avlodlarini tutib turish uchun mo‘ljallangan dasturda **exeption raised**ga ruhsat beradi.

Exeption handlingni ta’minlaydigan til foydalanuvchilarini dastur **execution**i davomida sodir bo‘ladigan va ular qanday saqlanishi ko‘rsatadigan hamma holatlarni o‘ylashga undaydi. Bu **approach** hech narsa yomon bo‘lmasligiga ishonadigan va ehtimollar haqida o‘ylashdan yaxshiroq.

Nihoyat, xatosizlik bilan shug‘ullanuvchi maxsus dastur bor, lekin noodatiy holat **exeption handling** bilan soddalashtira olinadi, va dastur strukturasida usiz butunlay tartibsiz bo‘lishi mumkin.

Biz hozir **exception handling** tizimi uchun dizayn **issue**sining ba’zilarini kashf etamiz, qachonki u dasturlash tilining bir qismi bo‘lganda. Bunday tizimlar **predefined exception**ga ham **user-defined exceptions** ga ham **exeption** ishlab

chiquvchiga ham ruhsat berishi mumkin. **Predefined exeption raised** bo‘lganini qayd qiling, garchi **user-defined exceptions** foydalanuvchi kodi orqali **raised** qilingan bo‘lsa ham. Quyidagi **raised exeption** uchun mo‘ljallangan **exception-handling** mehanizmini o‘z ichiga oladigan **subprogram** skeletiga qarang:

```
void example() {
. . .
average = sum / total;
. . .
return;
/* Istisno holatlar bilan ishlash */
when zero_divide {
average = 0;
printf("Error–divisor (total) is zero\n");
}
. . .
    }
```

Raised bo‘lgan nol orqali **exception** bo‘linishi keyin bajariladigan mos ishlab chiquvchini o‘tkazish nazoratiga sabab bo‘ladi.

Exception handling uchun birinchi dizayn **issue**si qanday qilib **exeption** ni istisno holatlar bilan ishlashga bog‘anishidir. Bu issue ikkita turli xil darajada sodir bo‘ladi. **Unit** darajasida turli xil **point**larga ko‘tarilgan bir xil **exception** qanday bo‘lishi haqida savollar bor, **unit**da unut bilan bog‘liq turli xil ishlab chiquvchila ulanishi mumkin. Masalan, **subprogram** misolida aniq ifoda (**statement**)da nol orqali bo‘linishlar bilan shug‘ullanuvchi yozilish uchun ko‘rinadigan **exception** nol orqali bo‘linish uchun ishlab chiquvchi bor (ko‘rsatilgan). Lekin funksiyalar taxmini bo‘lish operatori bilan birga bir nechta boshqa ifodalarni o‘z ichiga oladi. Bu operatorlar uchun ishlab chiquvchi mos bo‘lmasligi mumkin. Shuning uchun u asosiy ishlab chiquvchilarning asosiy ifoda (**statement**)lari tomonidan **raised** bo‘lishi mumkin bolgan **exception**larni yuklashi mumkin bo‘lishi mumkin garchi bir xil **exception** ko‘p turli ifoda (**statement**)lar tomonidan **raised** bo‘la olsa-da.

Istisno holatlar bilan ishlash bajarilgandan so‘ng, nazorat ishlab chiquvchi kodining tashqarisidagi dasturda o‘tkazilishi yoki dastur bajarilishi tugatilishi mumkin. Biz ishlab chiquvchi bajarilishidan keyin bu davomiylik nazorati savolini ko‘rib chiqamiz yoki oddiygina davomiylik. **Termination** aniq eng oddiy tanlov, va ko‘pchilik error exception holatida ing yaxshi. Bunga qaramay, boshqa holatlarda asosan bu noodatiy lekin xato hodisalarga bog‘lanadi, davomiylikni bajarish tanlovi eng yaxshi. Bu dizayn **resumption** deb nomlanadi. Bu vaziyatlarda ba’zi qulayliklar qaerda **execution** davom etishi tanlanishi kerak. Ifoda (**statement**)ga qaytish tanlovi **raised** qilganda **exception** yaxshiga o‘xshab tuyulishi mumkin, lekin xato **exception** vaziyatida u foydali agar ishlab chiquvchi birozgina raised bo‘lishga sabab bo‘ladigan **value**lar yoki operasiyalar aniqlaydi. Aks holda, **exception** osongina qayta **raised** bo‘lishi mumkin. Error exception uchun talab qilingan modifikasiya tez-tez juda bashorat qilish uchun juda qiyin bo‘ladi. Hatto bo‘lishi mumkin bo‘lganda ham bunga qaramay u voz amaliyoti bo‘lmasligi mumkin. U dasturga sabasni o‘chirmasdan muammoning simtomini o‘chirishga ruhsat beradi.

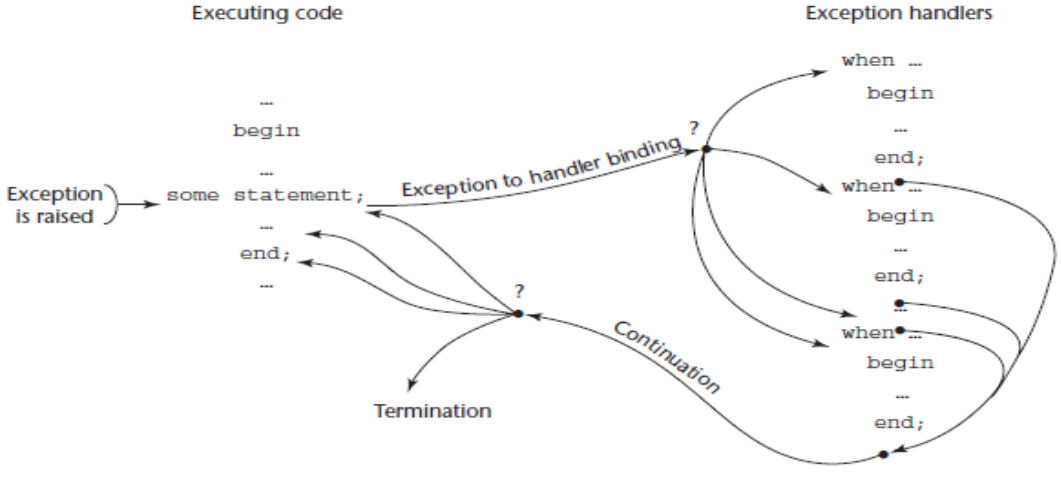
history note

PL/I (ANSI, 1976) pioneered the concept of allowing user programs to be directly involved in exception handling. The language allowed the user to write exception handlers for a long list of language-defined exceptions. Furthermore, PL/I introduced the concept of user-defined exceptions, which allow programs to create software-detected exceptions. These exceptions use the same mechanisms that are used for the built-in exceptions.

Since PL/I was designed, a substantial amount of work has been done to design alternative methods of exception handling. In particular, CLU (Liskov et al., 1984), Mesa (Mitchell et al., 1979), Ada, COMMON LISP (Steele, 1990), ML (Milner et al., 1990), C++, Modula-3 (Cardelli et al., 1989), Eiffel, Java, and C# include exception-handling facilities.

Ishlab chiquvchi **exception** va davomiylikni birlashtiradigan ikkita issue 14.1 rasm da ko‘rsatilgan.

Exception boshqaruvini o‘z ichiga olganda, **subprogram**ning bajarilishi ikkita yo‘lda tugatilishi mumkin: uning bajarilishi tugatilganda yoki u **exception**ni nazorat



qilganda. Ba’zi vaziyatlarda u subprogram bajarilishi qanday to‘xtatilishini e’tiborsiz amalga oshirilishi zarur. **Computation** uchun maxsuslashtirilgan qobiliyat ***finalization*** deb nomlanadi. **Finalization**ni ta’minlash tanlovi aniq istisno holatlar bilan ishlash uchun dizayn **issue**sidir.

Boshqa dizayn **issue**lari quyidagilar: Agar foydalanuvchi **exception**larni aniqlashga ruhsat bersa, bu **exception**lar qanday maxsuslashadi? Odatiy javob ular **raised** bo‘lishi mumkin bo‘lgan dastur **unit**larinig maxsus qismida e’lon qilinishi talabidir. E’lon qilingan **exception** imkoniyati odatda declorationni o‘z ichiga oladigan dastur unitining imkoniyatidir.

Boshqa issue foydalanuvchi dasturi tomonidan boshaqara olinadigan texnik ta’minot-**detectible error**lardir. Agar unday bo‘lmasa, hamma **exception**lar aniq dasturiy ta’minot (**software**) **detectible** bo‘ladi. Shunga bog‘liq savol ularda xoxlagan qaror qilingan **exception** bo‘lishi mumkin. Bu **exception**lar yoki texnik ta’minot yoki dasturiy ta’minot (**software**) orqali **raised** qilinadi.

Nihoyat, **exception** bo‘yicha savol, qaror qilingan yoki foydalanuvchi qaror qiladigan vaqtinchalik yoki doimiy ishda to‘xtatilishi mumkin. Bu savol ozroq filosofik ayniqsa qaror qilingan **error** vaziyatida. Masalan, tilni taxmin qilishni **range error** sodir bo‘lganda **raised** bo‘lgan qaror qilingan **exception** bor. Ko‘pchilik subscript range errors har doim amalga oshishi kerakligiga ishonishadi va shuning uchun u bu xatolarning bajarilishinito‘xtatish uchun dastur bo‘lmasligi kerak. Boshqalar **subscript range checking** maxsulot dasturiy ta’minot (**software**) uchun arzirli bo‘lishi, kodlar **range error** sodir bo‘lmasligi kerak erkin xato bo‘yicha baxslashadi.

Exception-handling dizayni **issue**si quyidagicha xulosalanadi:

- Qanday va qaerda istisno holatlar bilan ishlashs amalga oshadi, va uning imkoniyati qanday?
- Qanday qilib **exception occurrence** istisno holatlar bilan ishlashga bog‘lanadi?
- **Exception** haqidagi malumotlarni ishlab chiquvchiga o‘tkazishni iloji bormi?
- Agar istisno holatlar bilan ishlash o‘z ishini tugasa, undan keyin **execution** qaerda davom etadi? (Bu davomiylik va **resumption**ni savoli)
- **Finalization**ni bazi shakllari ta’minlanadimi?
- **User-defined exception** qanday ishga tushiriladi?
- Agar qaror qilingan **exception** bor bo‘lsa, o‘zini ta’minlamaydigan dastur uchun ishlab chiquvchi **default exception** holatida bo‘lishi kerakmi?
- Qaror qilingan **exception raised** holatda bo‘lishi mimkinmi?

- Texnik ta’minot **detectible errors** boshqarilishi mumkin bo‘lgani sifatida harakatlana oladimi?
- Xoxlagan qaror qilingan **exceptions** bormi?
- Qaror qilingan **exceptionsni** to‘xtatilishi kerakmi?

Biz hozir uchta vaqtinchalik dasturlash tilining **exception-handling** xususiyatini tekshirish vaziyatidamiz.

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

- **Dasturiy xatoliklar** - dasturlovchi yo‘l qo‘ygan xatolarni shunday ataydilar. Masalan, ilova boshqarib bo‘lmaydigan S++ tilida yaratilgan bo‘lsa. Agar dinamik xotira tozalanmagan bo‘lib, xotira yo‘qolishiga asos bo‘lsa, dasturiy xato kelib chiqadi.
- **Foydalanuvchilar xatolari -** boshqa tomondan, foydalanuvchi xatolari ilova kirituvchilar tomonidan ya’ni ilovalarni tuzgan kishilar tomonidan emas. Masalan, oxirgi foydalanuvchi, kiritishlar va kodlarda korrekt kiritishlarni qayta ishlash ko‘zda tutilmagan bo‘lishi mumkin.
- **Istisnolar -** istisnolar deb, dasturlash davomida nafaqat qiyin ba’zida dasturlash ilovalarida umuman ko‘zda tutib bo‘lmaydi. Istisnoga misol qilib, quyidagilarni aytish mumkin:

- **Data** - xususiyat faqat o‘qish uchun mo‘ljallangan bo‘lib, ma’lumotlarni olishga yordam beradi.
- **HelpLink** - xususiyat fayllardan xatolarni tuzatish va URL larni o‘rnatadi
- **Inter Exception** - faqatgina o‘qilishi mumkin bo‘lgan xossa bo‘lib, u oldindagi istisno yoki istisnolar haqida ma’lumot olish uchun ishlatiladi. Oldindagi istisnolar yozib olinishi eng oxirgi istisnoni konstruktorga uzatish orqali amalga oshiriladi
- **Message** - faqat o‘qish imkonini beruvchi xossa bo‘lib, matndagi berilgan xatoni qaytarib beadi. Xato haqidagi xabar konstruktor parametridek o‘rnatiladi.
- **Source** - xossa istisnolarni generatsiyaga kelganida sborka yoki obekt nomini olishga yordam beradi
- **StackTrace** - faqatgina o‘qish uchun mo‘ljallangan bo‘lib, o‘z ichida chaqiruvlar qatorini saqlaydi.bu qator esa istisnolar shakllanishiga olib keladi. Bunday xususiyat juda foydaliligini tushunish qiyin emas.
- **TargetSite** - xossa faqatgina o‘qilish imkoniga ega bo‘lib, obektni Messageda ko‘p sonli detallari bilan qaytaradi

Amaliy mashg‘ulot

Bir nechta **catch** bloklarni yozganimizda shu narsani e’tiborga olishimiz kerakki istisno xolat birinchi **catch** blok orqali qayta ishlanadi. Misol tariqasida, birinchi catch bloki nima ekanligini ko‘rsatish uchun avvalgi misolga yana bitta **CarlsDeadException va ArgumentOutOfRangeException** dan tashqari barcha istisnolarni tutib olmoqchi bo‘lgan **catch** blokini qo‘shamiz.

```
// This code will not compile!
static void Main(string[] args)
{
  Console.WriteLine("***** Handling Multiple Exceptions
*****\n");
```

```
    Car myCar = new Car("Rusty", 90);

    try
    {
      // Trigger an argument out of range exception.
      myCar.Accelerate(-10);
    }
    catch(Exception e)
    {
      // Process all other exceptions?
      Console.WriteLine(e.Message);
    }
    catch (CarIsDeadException e)
    {
      Console.WriteLine(e.Message);
    }
    catch (ArgumentOutOfRangeException e)
    {
      Console.WriteLine(e.Message);
    }
    Console.ReadLine();
  }
```

Bu istisnolarni qayta ishlash mantig‘i kompilyatsiya bosqichida xatolikka olib keladi. Muammo shundaki, birinchi **catch** bloki **System.Exception** xatoligidan tashqari barcha istisnolarni qayta ishlay oladi, **CarlsDeadException va ArgumentOutOfRangeException** larni ham. Oxirgi 2ta **catch** bloki imkonsizdir.

Bitta empirik qoidani yodda saqlang: catch bloklarini shunday joylashtirish kerakki, birinchi **catch** eng maxsus istisnolarni, oxirgi **catch** esa umumiyroq istisnolarni qayta ishlashi kerak.

Shunday qilib, agar ixtiyoriy istisnolarni qayta ishlaydigan **catch** blokini aniqlaydigan bo‘lsak, u quyidagicha aniqlanadi:

```
  // This code compiles just fine.
  static void Main(string[] args)
  {
    Console.WriteLine("***** Handling Multiple Exceptions
  *****\n");
```

```
  Car myCar = new Car("Rusty", 90);
  try
  {
    // Trigger an argument out of range exception.
    myCar.Accelerate(-10);
  }
  catch (CarIsDeadException e)
  {
    Console.WriteLine(e.Message);
  }
  catch (ArgumentOutOfRangeException e)
  {
    Console.WriteLine(e.Message);
  }
  // This will catch any other exception
  // beyond CarIsDeadException or
  // ArgumentOutOfRangeException.
  catch (Exception e)
  {
    Console.WriteLine(e.Message);
  }
  Console.ReadLine();
}



// A generic catch.
static void Main(string[] args)
{
  Console.WriteLine("***** Handling Multiple Exceptions
*****\n");
  Car myCar = new Car("Rusty", 90);
  try
  {
    myCar.Accelerate(90);
  }
  catch
  {
    Console.WriteLine("Something bad happened...");
  }
  Console.ReadLine();
```

}

Nazorat savollari

1. **System.IO.IOException hierarchy** dagi barcha istisno holatlarni toping.
2. **System.IO.IOException hierarchy** klassiga tegishli bo‘lgan va hierarchy qismi bo‘lgan barcha istisno holatlarni toping.
3. **System.IO.IOException hierarchy** dan barcha standart istisnolarni toping.
4. Istisno holatlarni qachon ishlatish va qanday qilib tutib qolish, ularni nazorat qilishni tushuntirib bering.
5. **Try – finally** qachon ishlatilishini tushuntirib bering. **Try – finally** va **string** o‘rtasidagi farqni tushuntirib bering.
6. Istisnolardan foydalanishning afzalliklarini ayting.
7. konsole oynasidan butun sonlarni o‘quvchi va undan kvadrat ildiz chiqaruvchi dastur tuzing. Agar kiritilgan ma’lumot bo‘lishsiz bo‘lsa konsole ga “ Yaroqsiz son” deb chiqarsin boshqa barcha holatlarda “Hayr” deb chiqarsin.
8. **ReadNumber(int start, int end) Console dan [start… end]** qatoridagi butun sonlarni o‘quvchi metod yarating. Aks holda kiritilgan butun son yaroqsiz yoki berilgan istisno holatlarga mos emas. Metoddan foydalanish, 10 ta butun sonni oluvchi dastur tzing
    i. **a2, …, a10 va 1 < a1 < … < a10 < 100.**
9. **Text** fayl tarkibini qaytaruvchi, uni o‘quvchi va fayl nomini oluvchi parametrli metod yarating. Qanday metoddan foydalaniladi va istisno holat bormi?
10. Binar fayl nomi parametrini, uni tarkibini va shu to‘plamni baytlarda qaytaruvchi metod tuzing.
11. **FileParseException** istisno xolati uchun ozingizni klassingizni aniqlang va internetdan ma’lumotlar to‘plang. Bu istisno holat jarayondagi fayl nomi va muammoni boshlangan nuqtasidan iborat bo‘lsin. Istisno holatlarga mos konstruksiyalarni qo‘shing. **Text file** dan butun sonlarni o‘quvchi dastur tuzing. Uni o‘qiyotgan vaqtda agar butun sonlar qatori **FileParseException** dan iborat bo‘lmasa, uni chaqirilayogan metodga qoldiring.
12. Foydalanuvchining barcha oldingi ma’lumotlarini qaytaruvchi dastur tuzing. Na’muna: **C:\Windows\win.ini),** fayl tarkibini o‘qib uni konsolega chiqarsin. **System.IOI.File.Read11Text(…)** da MSDN ni qanday foydalanishni toping.

13. Berilgan web manzildan ma'lumotni torib oluvchi dastur tuzing. **http://introprogramming.info/wp-content/themes/introprograming_en/images/Intro-Csharp-Book-front-cover-big_en.png.**

Testlar

1. Ilova boshqarib bo'lmaydigan S++ tilida yaratilgan bo'lsa. Agar dinamik xotira tozalanmagan bo'lib, xotira yo'qolishiga asos bo'lsa, bunday xato nima deb ataladi?
a. *Dasturiy xatoliklar*
b. *Foydalanuvchilar xatolari*
c. *Istisnolar*
d. Tizim xatoligi

2. Mabodo paydo bo'lgan istisnoni qayta ishlovchi (yoki ushlab turuvchi) kod bloki S# dasturlash tilida qaysi kalit so'zi yordamida namoyish qiladi?
a. try, catch, throw, finally
b. int, float, double
c. public, private, protected
d. for, while, foreach

3. Barcha istisnolar qaysi baza sinfdan yaratilgan?
a. System.Exception
b. System.Numerics
c. System.IO
d. System.LINQ

4. Data xususiyatining vazifasi qaysi javobda to'g'ri ko'rsatilgan?
a. bu xususiyat faqat o'qish uchun mo'ljallangan bo'lib, ma'lumotlarni olishga yordam beradi.
b. bu xususiyat fayllardan xatolarni tuzatish va URL larni o'rnatadi
c. bu faqatgina o'qilishi mumkin bo'lgan xossa bo'lib, u oldindagi istisno yoki istisnolar haqida ma'lumot olish uchun ishlatiladi. Oldindagi istisnolar yozib olinishi eng oxirgi istisnoni konstruktorga uzatish orqali amalga oshiriladi
d. bu faqat o'qish imkonini beruvchi xossa bo'lib, matndagi berilgan xatoni qaytarib beadi. Xato haqidagi xabar konstruktor parametridek o'rnatiladi.

5. Agar siz xaqiqatdan ham zo'r istisnolar sinfini qurmoqchi bo'lsangiz unda u har taraflama .NET talablariga to'g'ri kelmog'i lozim. Bu talablar quydagilar:
1) Exception bo'lishi kerak /ApplicationException;
2) [System.Serializable] atributi bilan belgilanishi kerak;
3) Konstruktor qurilishini aniqlash
4) Messagega tegishli bo'lganligi haqidagi konstruktori bo'lmog'i lozim;
5) "Ichki xatolar" konstruktorini ishlatishi kerak;
6) System.IO nomlar fazosini kiritish kerak
7) Klasslarni public deb e'lon qilish kerak
a. 1,2,3,4,5
b. 3,4,5,6,7
c. 2,3,4,5,6
d. 1,2,3,4,7

6. Bu faqat o‘qish imkonini beruvchi xossa bo‘lib, matndagi berilgan xatoni qaytarib beadi. Xato haqidagi xabar konstruktor parametridek o‘rnatiladi. Ushbu ta’rif qaysi xossaga tegishli?

a. Message
b. Source
c. StackTrace
d. TargetSite

7. Bu faqatgina o‘qish uchun mo‘ljallangan bo‘lib, o‘z ichida chaqiruvlar qatorini saqlaydi.bu qator esa istisnolar shakllanishiga olib keladi. Bunday xususiyat juda foydaliligini tushunish qiyin emas.ushbu ta’rif qaysi xossaga tegishli?

a. StackTrace
b. Message
c. Source
d. Inter Exception

8. Umuman olganda blok try operatorlar bo‘limini tashkil etadi. Ish davomida ular istisnolarni tashkil etadi. Istisnolar aniqlansa boshqaruv qaysi blogikga o‘tadi?

a. catch
b. for
c. While
d. Throw

9. Bu faqatgina o‘qilishi mumkin bo‘lgan xossa bo‘lib, u oldindagi istisno yoki istisnolar haqida ma’lumot olish uchun ishlatiladi. Oldindagi istisnolar yozib olinishi eng oxirgi istisnoni konstruktorga uzatish orqali amalga oshiriladi ushbu ta’rif qaysi xossaga tegishli?

a. Inter Exception
b. Message
c. Source
d. StackTrace

12-Mavzu. Satrlar va fayllar bilan ishlash

Fayl va kataloglar ro‘yxati.

Bizga ma’lumki, fayl deb xotiraning nomlangan sohasiga aytiladi. Faylda turli ma’lumotlar saqlanadi. Har bir fayl bilan fayl ko‘rsatkichi degan tushuncha biriktirilgan. Fayl bir n yecha elementlardan iborat bo‘lib, foydalanuvchi faqat faylning ko‘rsatkichi ko‘rsatayotgan ma’lumotga murojaat qilishi mumkin. Demak, fizik jihatdan biz faqat ketma-ket fayllarga egamiz. Ya’ni biz oldin birinchi, keyin ikkinchi, uchinchi va h.k. ma’lumotlarni o‘qishimiz mumkin. Fayl o‘z nomiga ega. Masalan, **d:\ malumot.txt. C#** tili dasturiy vositalari yordamida, ya’ni dasturda ham fayllarni tashkil qilish va undagi ma’lumotlarni qayta ishlash mumkin. Shu paytga qadar, **C#** dasturlash tilida bir necha o‘zgaruvchilarning toifalari bilan ishlab keldik.

Bular skalyar, oddiy va murakkab tarkiblashgan toifalardir. Bu toifadagi ma’lumotlar yordamida masalalarni yechishda boshlang‘ich ma’lumotlar klaviaturadan operativ xotiraga kiritiladi va natija ekranga chiqariladi. Ulardan boshqa dasturlarda foydalanib bo‘lmaydi, chunki ular tizimidan chiqilgandan so‘ng ma’lumotlar hech qayerda saqlanmaydi. Bu ma’lumotlarni xotirada saqlash uchun C# dasturlash tilida ma’lumotlarning faylli toifasi belgilangan. Fayl toifasi alohida o‘rin eg allaydi. Fayl toifasi bilan ishlashda ma’lum tushunchalarni o‘zlashtirish talab qilinadi. Birinchidan, fayllar toifasi nega va qachon qo‘llaniladi? Maqsad nima? Zaruriyat nimadan kelib chiqyapti? Ikkinchidan, boshqa toifalardan n yega katta farqi bor? Bu savollarga faqat foydalanuvchining nuqtai nazaridan qaragan holda javob bera olamiz:

1. Juda ko‘p o‘zgaruvchilardan foydalanganda ularning qiymatlarini har doim klaviaturadan kiritishda ma’lum noqulayliklarga duch kelamiz. Bunga katta massivlar misol bo‘la oladi.

2. Shunday masalalar uchraydiki, oldindan kattaliklarning qiymatlar soni noma’lum bo‘ladi (masalan, natijalar), bu kattaliklarni faylga yozish maqsadga muvofiq. 3.Hech qanday toifalar tashqi qurilmalarga murojaat qilib, ular bilan ishlashga imkon yaratmaydi (dasturiy til muhitida). Va nihoyat, boshqa toifalardan fayl toifasi farqliligi shundaki, u boshqa toifalar tarkibiga kira olmaydi. Fayllarning turlari.

Fayllar uchun mo‘ljallangan umumiy protsedura va funktsiyalar Faylda saqlanayotgan ma’lumotlar turiga ko‘ra, turlarga bo‘linadi:

1) toifalashmagan;

2) toifalashgan;

3) matnli.

Toifalashgan fayllar bir xil toifali elementlardan tashkil topadi. Ularni faqat ma'lum qurilmalarda uzatish mumkin, lekin ekranda o'qish mumkin emas. Faylning elementlari mashina kodlarida yoziladi va saqlanadi. Toifalashmagan fayllarda turli toifadagi ma'lumotlarni saqlash mumkin. Ular ham mashina kodlari bilan yozilgan bo'lib baytlar to'plamni tashkil qiladi. Matnli fayllar **ASCII** kodlardan tashkil topgan va qatorlarga ajratilgan bo'ladi. Matnli fayllarda nafaqat faylning yakunida fayl oxiri belgisi, balki har qatorning yakunida maxsus qator oxiri b yelgisi qo'yiladi. Fayl turidagi o'zgaruvchi fayl o'zgaruvchisi deyiladi, u faylning mantiqiy nomini belgilaydi va u mantiqiy fayl bilan tashqi (fizik) fayl o'rtasida «vositachi» vazifasini o'ynaydi. Fayl turi uchun arifmetik amallar belgilanmagan. Xatto fayllarni solishtirish va bir faylning qiymatini ikkinchi faylga o'zlashtirish amallari ham aniqlanmagan. Har bir turdagi fayllar ustida, umuman olganda, quyidagi amallarni bajarish mumkin va bu amallar uchun maxsus protsedura va funktsiyalar ishlatiladi. Hozir biz katalog va fayllar ro'yxati ustida bajariladigan amallarni qarab chiqamiz.

Kompyuterda mavjud mantiqiy disklarni aniqlash uchun **GetLogicalDrives()** metodidan foydalanamiz. Quyida kompyuterda mavjud barcha mantiqiy disklarni ro'yxatini chiqaruvchi dastur keltirilgan.

```
class Program {
static void Main(string[] args) {
string[] LogicalDrives = Environment.GetLogicalDrives();
foreach (string a in LogicalDrives) {
Console.WriteLine(a);
}
Console.ReadKey();
} }
```

Dastur natijasi:

```
C:\
D:\
E:\
F:\
```

Keyingi misolda yuzaga keladigan xatoliklarni oldini olish uchun **System.Security.SecurityException** metodidan foydalanamiz.

```
class Program
{
```

```
    [STAThread]
    static void Main(string[] args)
    {
        GetLogicalDrives();
        Console.ReadLine();
    }
    static void GetLogicalDrives()
    {
        try
        {
            string[] a =
System.IO.Directory.GetLogicalDrives();
            foreach (string b in a)
            {
                System.Console.WriteLine(b);
            }
        }
        catch (System.IO.IOException)
        {
            System.Console.WriteLine("xato");
        }
        catch (System.Security.SecurityException)
        {
            System.Console.WriteLine("xato 1");
        }
        Console.ReadKey();
    }
}
```

Diskdagi kataloglar ro‘yxatini chiqarish uchun **System.IO. Directory** sinfining **GetDirectories()** metodidan foydalanamiz. Quyida uning dasturi keltirilgan.

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            string[] a = Directory.GetDirectories(@"d:\\");
            Console.WriteLine("hamma papkalar :{0}.",
a.Length);
```

```
            foreach (string b in a)
            {
                Console.WriteLine(b);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("xato: {0}", e.ToString());
        }
        Console.ReadKey();
    }
     }
```

Dastur natijasi: D diskdagi barcha kataloglarni ro‘yxatini chiqaradi.

Fayl va kataloglar ustida amallar.

Bu bo‘limda biz fayl va kataloglar ustida bajariladigan asosiy amallarni ko‘rib chiqamiz. Bu uchun biz C# dasturlash tilida qaysi kutubxonadan foydalanamiz, qaysi sinflardan foydalanamiz va qaysi metodlardan foydalanamiz? har biriga alohida to‘xtalib o‘tamiz. C# dasturlash tilida fayl va kataloglar ustida amallar bajarish uchun juda ko‘p sinflar yaratilgan va bu sinflarda fayl va kataloglar ustida amallar bajarish uchun juda ko‘p metodlar mavjud. Mana shu metodlardan qanday qilib foydalanish jarayonini misollar orqali qarab chiqamiz.

Kataloglar ustida bajariladigan asosiy amallar **System.IO.Directory** sinfining metodlari orqali amalga oshiriladi.

1. **DirectoryInfo CreateDirectory**(string nom) – yangi katalog yaratish .

2. **void Move**(string eski nom, string yangi nom)- katalog nomini o‘zgartirish ki katalogni o‘chirish.

3. **void Delete**(string nomi, bool x)- katalogni o‘chirish, agar x parametrning qiymati **true** bo‘lsa bu metod katalog ichidagi fayllar bilan birgalikda o‘chiradi.

4. **bool Exists**(stirng nomi) – bu metod chin qiymat qaytaradi agar papka mavjud bo‘lsa, aks xolda yolg‘on qiymat qaytaradi.

Ma’lumotlarni faylda yozish va o‘qish.

Fayl bu ma’lumotlarning fundamental strukturalaridan biri. Kompyuterlarning dastur bilan ishlashi, tashqi qurilmalar bilan aloqasi fayl strukturasiga asoslangandir.

Fayllar quyidagi masalalarni yechishga asoslangandir:

1. Qiymatlarni boshqa dasturlar foydalanishi uchun saqlab qo‘yish;

2. Dasturning kiritish-chiqarish tashqi qurilmalari bilan aloqasini tashkil qilish.

Fayl tushunchasining ikki tomoni bor:

1- Tomondan fayl – tashqi xotiraning biror axborotni saqlovchi nomlangan bir qismidir. Bunday tushunchadagi fayl fizik fayl deb ataladi.

2- Tomondan fayl – bu dasturda ishlatiladigan ma’lumotlarning turli strukturalaridan biridir. Bunday tushunchadagi fayl mantiqiy fayl deb ataladi, yani u bizning tasavvurimiz bilan hosil qilinadi.

Fizik fayl strukturasi

Fizik fayl strukturasi axborot tashuvchi – qattiq yoki yumshoq magnit disklaridagi baytlarning oddiy ketma-ketligidan iborat.

Mantiqiy fayl strukturasi

Mantiqiy fayl strukturasi – bu, faylni dasturda qabul qilish usulidir.

Agar fayl strukturasini massiv strukturasiga solishtirsak, quyidagi farqlarni ko‘ramiz:

1. Massivda elementlar soni xotirani taqsimlash vaqtida o‘rnatiladi va u to‘laligicha tezkor xotiraga joylashadi. Massiv elementlari raqamlanishi uni e’lon qilishda ko‘rsatilgan quyi va yuqori chegaralarga mos holda o‘rnatiladi.

2. Faylda esa elementlar(yozuvlar) soni dastur ishi jarayonida o‘zgarishi mumkin va u tashqi axborot tashuvchilarda joylashgan. Fayl elementlarini raqamlash 0 dan boshlanadi va elementlar soni doim noaniq.

Biz yuqorida o‘zgaruvchilarning turli toifalari bilan ishlab keldik. Bular skalyar, oddiy va murakkab tarkiblashgan toifalardir. Bu toifadagi ma’lumotlar yordamida masalalarni yechishda boshlang‘ich ma’lumotlar klaviaturadan operativ xotiraga kiritiladi va natija ekranga chiqariladi. Ulardan boshqa dasturlarda foydalanib bo‘lmaydi, chunki ular tizimdan chiqilgandan so‘ng hech qayerda saqlanmaydi. Bu ma’lumotlarni xotirada saqlash uchun C# tilida ma’lumotlarning faylli toifasi belgilangan.

Toifalashgan fayllar bir xil toifali elementlardan tashkil topadi. Ularni ma’lum qurilmalarda uzatish ham mumkin. Faylning elementlari mashina kodlarida yoziladi va saqlanadi.

Toifalashmagan fayllarda turli toifadagi malumotlarni saqlash mumkin. Ular ham mashina kodlarida yozilgan bo‘lib baytlar to‘plamini tashkil qiladi.

Matnli fayllar **ASCII** kodlardan tashkil topgan va qatorlarga ajratilgan bo‘ladi. Matnli fayllarda nafaqat faylning yakunida fayl oxiri belgisi, balki har qatorning yakunida maxsus qator oxiri belgisi qo‘yiladi.

Asosiy adabiyotlar

1. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
2. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
3. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.

Glossariy

- **Sikl operatori** – takrorlash sharti deb nomlanuvchi mantiqiy ifodani rost (true) qiymatida programmaning ma’lum bir qismlaridagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajarishni amalga oshiradi (itaratsiya)
- **Break** – operatori sikldan chiqib ketishda foydalaniladi
- **BigInteger –** katta sonlar bilan ishlashga mo‘ljallangan klass
- **continue** – bu operatori xuddi break operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin programmani qurilmadan chiqib ketmasdan takrorlashning keyingi qadamiga “sakrab” o‘tishini tayinlaydi
- **foreach –** bu sikl operatori C/C++/C# dasturlash tillari oilasi uchun yangi ammo bu **VP va PHP** dasturchilari uchun yaxshi tanish dasturning

konstruksiyasida massivning barcha elementlarini va ro‘yxatdan yoki elementlar yig‘indisida (**IEnumerable**) ishlatiladi

- **Write(…) –** konsolga argumentlarni chiqarish
- **WriteLine(…) –** konsolga ma’lumotlarni chiqarish va keyingi qatorga o‘tish
- **Console.ReadKey() –** klavish bosilguncha ekranni ushlab turadi
- **KeyChar –** kiritilgan belgini ushlab turadi
- **int.Parse(string) –** satrni butun tipga o‘tkazadi
- **Convert klassi –** metodidan foydalangan holda boshqa tipga o‘tkazish uchun ishlatiladi
- **try-catch –** hatoliklarni ushlab qolish uchun ishlatiladi
- **While, do –** while, for, foreach – sikl operatorlari

Bizga ma’lumki, fayl deb xotiraning nomlangan sohasiga aytiladi.

Faylda turli ma'lumotlar saqlanadi. Har bir fayl bilan fayl ko‘rsatkichi degan tushuncha biriktirilgan. Fayl bir necha elementlardan iborat bo‘lib, foydalanuvchi faqat faylning ko‘rsatkichi ko‘rsatayotgan ma’lumotga murojaat qilishi mumkin. Demak, fizik jihatdan biz faqat ketma-ket fayllarga egamiz.

Ya’ni biz oldin birinchi, keyin ikkinchi, uchinchi va h.k. ma’lumotlarni o‘qishimiz mumkin. Fayl o‘z nomiga ega. Masalan, **d:\malumot.txt. C#** tili

dasturiy vositalari yordamida, ya’ni dasturda ham fayllarni tashkil qilish va undagi ma’lumotlarni qayta ishlash mumkin. Shu paytga qadar, C# dasturlash tilida bir necha o‘zgaruvchilarning toifalari bilan ishlab keldik.

Bular skalyar, oddiy va murakkab tarkiblashgan toifalardir. Bu toifadagi ma’lumotlar yordamida masalalarni yechishda boshlang‘ich ma’lumotlar klaviaturadan operativ xotiraga kiritiladi va natija ekranga chiqariladi.

Ulardan boshqa dasturlarda foydalanib bo‘lmaydi, chunki ular tizimidan chiqilgandan so‘ng ma’lumotlar hech qayerda saqlanmaydi. Bu ma'lumotlarni xotirada saqlash uchun C# dasturlash tilida ma'lumotlarning faylli toifasi belgilangan. Fayl toifasi alohida o‘rin eg allaydi. Fayl toifasi bilan ishlashda ma'lum tushunchalarni o‘zlashtirish talab qilinadi. Birinchidan, fayllar toifasi nega va qachon qo‘llaniladi? Maqsad nima? Zaruriyat nimadan kelib chiqyapti? Ikkinchidan, boshqa toifalardan n yega katta farqi bor? Bu savollarga faqat foydalanuvchining nuqtai nazaridan qaragan holda javob bera olamiz:

1. Juda ko‘p o‘zgaruvchilardan foydalanganda ularning qiymatlarini har

doim klaviaturadan kiritishda ma’lum noqulayliklarga duch kelamiz. Bunga katta massivlar misol bo‘la oladi.

2. Shunday masalalar uchraydiki, oldindan kattaliklarning qiymatlar soni noma'lum bo‘ladi (masalan, natijalar), bu kattaliklarni faylga yozish maqsadga muvofiq.

3. Hech qanday toifalar tashqi qurilmalarga murojaat qilib, ular bilan ishlashga imkon yaratmaydi (dasturiy til muhitida). Va nihoyat, boshqa toifalardan fayl toifasi farqliligi shundaki, u boshqa toifalar tarkibiga kira olmaydi. Fayllarning turlari.

Fayllar uchun mo‘ljallangan umumiy protsedura va funksiyalar Faylda saqlanayotgan ma’lumotlar turiga ko‘ra, turlarga bo‘linadi:

1) toifalashmagan;

2) toifalashgan;

3) matnli.

Toifalashgan fayllar bir xil toifali elementlardan tashkil topadi. Ularni faqat ma’lum qurilmalarda uzatish mumkin, lekin ekranda o‘qish mumkin emas. Faylning elementlari mashina kodlarida yoziladi va saqlanadi. Toifalashmagan fayllarda turli toifadagi ma’lumotlarni saqlash mumkin. Ular ham mashina

kodlari bilan yozilgan bo‘lib baytlar to‘plamini tashkil qiladi. Matnli fayllar **ASCII** kodlardan tashkil topgan va qatorlarga ajratilgan bo‘ladi. Matnli fayllarda nafaqat faylning yakunida fayl oxiri belgisi, balki har qatorning yakunida maxsus qator oxiri belgisi qo‘yiladi. Fayl turidagi o‘zgaruvchi fayl o‘zgaruvchisi deyiladi, u faylning mantiqiy nomini belgilaydi va u mantiqiy fayl bilan tashqi (fizik) fayl o‘rtasida «vositachi» vazifasini o‘ynaydi. Fayl turi uchun arifmetik amallar belgilanmagan. Xatto fayllarni solishtirish va bir faylning qiymatini ikkinchi faylga o‘zlashtirish amallari ham aniqlanmagan. Har bir turdagi fayllar ustida, umuman olganda, quyidagi amallarni bajarish mumkin va bu amallar uchun maxsus **prosedura** va funktsiyalar ishlatiladi. Hozir biz katalog va fayllar ro‘yxati ustida bajariladigan amallarni qarab chiqamiz.

1. Kompyuterda mavjud mantiqiy disklarni aniqlash uchun

GetLogicalDrives() metodidan foydalanamiz. Quyida kompyuterda mavjud barcha mantiqiy disklarni ro‘yxatini chiqaruvchi dastur keltirilgan.

```
class Program {
static void Main(string[] args) {
string[] LogicalDrives = Environment.GetLogicalDrives();
foreach (string a in LogicalDrives) {
Console.WriteLine(a);
}
Console.ReadKey();
} }
Dastur natijasi:
C:\
D:\
E:\
F:\
```

Keyingi misolda yuzaga keladigan xatoliklarni oldini olish uchun

System.Security.SecurityException metodidan foydalanamiz.

```
class Program
{
    [STAThread]
    static void Main(string[] args)
    {
        GetLogicalDrives();
        Console.ReadLine();
```

```
    }
    static void GetLogicalDrives()
    {
        try
        {
            string[] a =
System.IO.Directory.GetLogicalDrives();
            foreach (string b in a)
            {
                System.Console.WriteLine(b);
            }
        }
        catch (System.IO.IOException)
        {
            System.Console.WriteLine("xato");
        }
        catch (System.Security.SecurityException)
        {
            System.Console.WriteLine("xato 1");
        }
        Console.ReadKey();
    }
}
```

2. Diskdagi kataloglar ro‘yxatini chiqarish uchun **System.IO. Directory**

sinfining **GetDirectories()** metodidan foydalanamiz. Quyida uning dasturi keltirilgan.

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            string[] a = Directory.GetDirectories(@"d:\\");
            Console.WriteLine("hamma papkalar :{0}.",
a.Length);
            foreach (string b in a)
            {
                Console.WriteLine(b);
            }
        }
```

```
        catch (Exception e)
        {
            Console.WriteLine("xato: {0}", e.ToString());
        }
        Console.ReadKey();
    }
}
```

Dastur natijasi: D diskdagi barcha kataloglarni ro‘yxatini chiqaradi.

3. Endi maska orqali diskdagi kataloglar ro‘yxatini chiqarishni ko‘rib chiqamiz. Ya’ni quyida d diskdagi c harfi bilan boshlanuvchi barcha kataloglar ro‘yxatini chiqaruvchi dastur keltirilgan.

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            string[] a = Directory.GetDirectories(@"d:\\",
"c*");
            Console.WriteLine("barcha c harfi bilan boshlangan
papkalar: {0}.", a.Length);
            foreach (string b in a)
            {
                Console.WriteLine(b);
            }
        }
        catch (Exception e)
        { Console.WriteLine("Xato: {0}", e.ToString()); }
        Console.ReadKey();
    }
     }
```

Testlar

10. Ilova boshqarib bo‘lmaydigan S++ tilida yaratilgan bo‘lsa. Agar dinamik xotira tozalanmagan bo‘lib, xotira yo‘qolishiga asos bo‘lsa, bunday xato nima deb ataladi?

e. *Dasturiy xatoliklar*

f. *Foydalanuvchilar xatolari*

g. *Istisnolar*

h. Tizim xatoligi

11. Mabodo paydo bo‘lgan istisnoni qayta ishlovchi (yoki ushlab turuvchi) kod bloki S# dasturlash tilida qaysi kalit so‘zi yordamida namoyish qiladi?

e. try, catch, throw, finally

f. int, float, double

g. public, private, protected

h. for, while, foreach

12. Barcha istisnolar qaysi baza sinfdan yaratilgan?

e. System.Exception

f. System.Numerics

g. System.IO

h. System.LINQ

13. Data xususiyatining vazifasi qaysi javobda to‘g‘ri ko‘rsatilgan?

e. bu xususiyat faqat o‘qish uchun mo‘ljallangan bo‘lib, ma’lumotlarni olishga yordam beradi.

f. bu xususiyat fayllardan xatolarni tuzatish va URL larni o‘rnatadi

g. bu faqatgina o‘qilishi mumkin bo‘lgan xossa bo‘lib, u oldindagi istisno yoki istisnolar haqida ma’lumot olish uchun ishlatiladi. Oldindagi istisnolar yozib olinishi eng oxirgi istisnoni konstruktorga uzatish orqali amalga oshiriladi

h. bu faqat o‘qish imkonini beruvchi xossa bo‘lib, matndagi berilgan xatoni qaytarib beadi. Xato haqidagi xabar konstruktor parametridek o‘rnatiladi.

14. Agar siz xaqiqatdan ham zo‘r istisnolar sinfini qurmoqchi bo‘lsangiz unda u har taraflama .NET talablariga to‘g‘ri kelmog‘i lozim. Bu talablar quydagilar:

8) Exception bo‘lishi kerak /ApplicationException;
9) [System.Serializable] atributi bilan belgilanishi kerak;
10) Konstruktor qurilishini aniqlash
11) Messagega tegishli bo‘lganligi haqidagi konstruktori bo‘lmog‘i lozim;
12) “Ichki xatolar” konstruktorini ishlatishi kerak;

13) System.IO nomlar fazosini kiritish kerak
14) Klasslarni public deb e'lon qilish kerak

e. 1,2,3,4,5

f. 3,4,5,6,7

g. 2,3,4,5,6

h. 1,2,3,4,7

15. Bu faqat o'qish imkonini beruvchi xossa bo'lib, matndagi berilgan xatoni qaytarib beadi. Xato haqidagi xabar konstruktor parametridek o'rnatiladi. Ushbu ta'rif qaysi xossaga tegishli?

e. Message

f. Source

g. StackTrace

h. TargetSite

16. Bu faqatgina o'qish uchun mo'ljallangan bo'lib, o'z ichida chaqiruvlar qatorini saqlaydi.bu qator esa istisnolar shakllanishiga olib keladi. Bunday xususiyat juda foydaliligini tushunish qiyin emas.ushbu ta'rif qaysi xossaga tegishli?

e. StackTrace

f. Message

g. Source

h. Inter Exception

17. Umuman olganda blok try operatorlar bo'limini tashkil etadi. Ish davomida ular istisnolarni tashkil etadi. Istisnolar aniqlansa boshqaruv qaysi blogikga o'tadi?

e. catch

f. for

g. While

h. Throw

18. Bu faqatgina o'qilishi mumkin bo'lgan xossa bo'lib, u oldindagi istisno yoki istisnolar haqida ma'lumot olish uchun ishlatiladi. Oldindagi istisnolar yozib olinishi eng oxirgi istisnoni

konstruktorga uzatish orqali amalga oshiriladi ushbu ta’rif qaysi xossaga tegishli?

e. Inter Exception

f. Message

g. Source

h. StackTrace