O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI

BUXORO DAVLAT UNIVERSITETI

Fizika – matematika fakulteti

“Axborot texnologiyalari” kafedrasi

Mirzayev Tolib To`raqul o`g`li

C# tilida strukturali dasturlash bo`yicha o`quv-uslubiy qo`llanma yaratish

5111000-Kasb ta’limi(5330200-Informatika va axborot texnologiyalari) ta’lim yo’nalishi bo’yicha bakalavr darajasini olish uchun

BITIRUV MALAKAVIY ISHI

“Ish ko`rildi va himoya qilishga ruxsat berildi”
Kafedra mudiri.
_____ dots. O.I.Jalolov
«_____» _____2019 y.

Ilmiy rahbar _____ katta o`qit. X.U.Xayatov
“_____” _____ 2019 y.
Taqrizchi _____ kat.o`qit.B.R.Sariyev
“_____” _____ 2019 y.

“Himoyaga ruxsat berildi”
Fakultet dekani _____ p.f.n. dots.M.I.Daminov
“_____” _____ 2019 y.

Buxoro - 2019 yil

MUNDARIJA

KIRISH..... 3
I BOB. DASTURLASH TILLARI 7
1.1. Dasturlash tillarining evolyutsiyasi…….7
1.2. .Net platforma……28
1.3. Visual Studio muhiti va unda ishlash.....35
II BOB. C# TILIDA STRUKTURALI DASTURLASH 41
2.1. C# tili sintaksisi, standart tiplar…….41
2.2. Ifoda, intruksiya va operatorlar……47
2.3. Boshqaruv operatorlari……..51
2.4. Satrlar……65
2.5. Massivlar……...73
2.6. Funktsiyalar. Fayllar bilan ishlash……88
XOTIMA..... 95
FOYDALANILGAN ADABIYOTLAR 96

KIRISH

“Biz yoshlarga doir davlat siyosatini hech og’ishmasdan,qat’iyat bilan davom ettiramiz. Nafaqat davom ettiramiz, balki bu siyosatni eng ustuvor vazifamiz sifatida butun zamon talab qilayotgan yuksak darajaga ko’taramiz.”

Sh. M. Mirziyoyev

2017 yil 20 aprelda O‘zbekiston Respublikasi Prezidentining “Oliy ta’lim tizimini yanada rivojlantirish chora-tadbirlari to‘g‘risida”gi qarori qabul qilindi. Ushbu qaror oliy ta’lim tizimini tubdan takomillashtirish, mamlakatimizni ijtimoiy-iqtisodiy rivojlantirish borasidagi ustuvor vazifalarga mos holda, kadrlar tayyorlashning ma’no-mazmunini tubdan qayta ko‘rib chiqish, xalqaro standartlar darajasida oliy malakali mutaxassislar tayyorlash uchun zarur sharoitlar yaratish maqsadiga yo‘naltirilgan.

Qarorga ko‘ra, oliy ta’lim tizimini kelgusida kompleks rivojlantirishning muhim vazifalari belgilab berildi. Qarorda belgilangan vazifalarning samarali yechimini to‘liq ta’minlash maqsadida oliy ta’lim darajasini sifat jihatdan oshirish va tubdan takomillashtirish, oliy ta’lim muassasalari moddiy-texnika bazasini mustahkamlash va modernizatsiya qilish, ularni zamonaviy o‘quv-ilmiy laboratoriyalari, axborot-kommunikatsiya texnologiyalar bilan jihozlash maqsadida Oliy ta’lim tizimini 2017-2021 yillarga mo‘ljallangan kompleks rivojlantirish dasturi tasdiqlandi.

Dasturga muvofiq, 2017-2021 yillarda 48 ta oliy ta’im muassasasida jami 180 ta o‘quv, ilmiy-laboratoriya binosi, sport inshootlari va ijtimoiy-muhandislik infratuzilmalari ob’ektlarida qurilish, rekonstruksiya va kapital ta’mirlash ishlari olib boriladi. Shuningdek, 53 ta oliy ta’lim muassasasida 400 ta o‘quv laboratoriyasi bosqichma-bosqich eng zamonaviy o‘quv-laboratoriya uskunalari bilan jihozlanadi,7 ta oliy ta’lim muassasasida barcha oliy ta’lim muassasalari o‘zaro hamkorlikda foydalanadigan ilmiy laboratoriyalar tashkil etiladi.

Qarorning yana bir muhim jihati shundaki, mamlakatimiz Prezidenti tomonidan har bir oliy ta’lim muassasasi bo‘yicha konkret parametr va ko‘rsatkichlarni o‘z ichiga olgan manzilli rivojlantirish dasturilari tasdiqlandi.

Mazkur dasturda asosan mamlakatimizning har bir oliy ta’lim muassasasi bilan, xususan, AQSh, Buyuk Britaniya. Fransiya, Italiya, Niderlandiya, Rossiya, Yaponiya, Janubiy Koriya, Xitoy va shu kabi boshqa davlatlarning yetakchi ilmiy-ta’lim muasasasalari bilan hamkorlik aloqalarining o‘rnatilgani o‘ta muhim ahamiyat kasb etadi. Shu asosda har yili 350 nafardan ortiq xorijlik yuqori malakali pedogog va olimlarning mamlakatimiz oliy o‘quv yurtlariga o‘quv jarayoniga jalb etilishi ko‘zda tutilmoqda.

Shu o‘rinda aytish lozimki, Oliy ta’lim tizimini 2017-2021 yillarga mo‘ljallangan kompelks rivojlantirish dasturini amalga oshirish uchun yo‘naltiriladigan moliyaviy mablag‘lar 1,7 trillion so‘mdan ziyod bo‘lib, ulardan 1,2 trillion so‘mi o‘quv-laboratoriya binolari, sport zallari va talabalar turarjoylarini rekonsturksiya qilish va kapital ta’mirlashga, 500 milliard so‘mdan ortiq mablag‘ esa o‘quv-laboratoriya uskunalari, mebel va inventar bilan ta’minlash, umumiy tartibda foydalanishga mo‘ljallangan, barcha ta’lim musassasalariga xizmat ko‘rsatadigan laboratoriya kompelkslarini tashkil etish hamda axborot-kommunikatsiya texnologiyalarini rivojlantirishga sarflanadi. [1]

Mavzuning dolzarbligi. Keyingi yillarda amaliy dasturchilarga juda ko‘p integratsion dastur tuzish muhitlari taklif etilmoqda. Bu muhitlar u yoki bu imkoniyatlari bilan bir–biridan farq qiladi. Shundan kelib chiqadiki dastur tuzishga talab ko`p. Dastur tuzish uchun dasturlash tilini o`rganish kerak. C# tilida strukturali dasturlash bo`yicha o`quv uslubiy qo`llanma yaratish dolzarblidir.

Bitiruv malakaviy ishimizning maqsad va vazifalari. Visual Studio muhiti va C# dasturlash tilining barcha imkoniyatlari bilan yaqindan tanishish. C# tilini o`rganish. C# tilida strukturali dasturlash bo`yicha o`quv-uslubiy qo`llanma tayyorlash.

Bitiruv malakaviy ishimizning o’rganilganlik darajasi. Visual Studio muhiti interfeysi bilan to’liq tanishib chiqilgan. C# dasturlash tilida yoziladigan barcha buyruqlar bilan to’liq tanishib chiqilgan. Unda bajarilgan amaliy ishlarning bir qanchasi o’rganib chiqilgan va o’quv uslubiy qo’llanma tayorlangan.

Bitiruv malakaviy ishimizning predmeti. Kompyuter, Visual Studio muhiti va C# dasturlash tili. Elektron adabiyotlar, elektron kutubxona.

Bitiruv malakaviy ishimizning obyekti. Kompyuter, Visual Studio muhiti va C# dasturlash tili.

Bitiruv malakaviy ishimizning ilmiy farazi. C# dasturlash tilida ishlash bo'yicha o'quv uslubiy qo'llanma yaratish orqali ta'lim jarayonida bu dasturning imkoniyatlarini ochib berish. Ta'lim jarayonini yanada yuksaltirish, yangi usul va vositalarini yaratish, o'quv jarayonining samarodorligini oshirish.

Bitiruv malakaviy ishimizning yangiligi. Bugungi kunga qadar C# dasturlash tilidan o'zbek tilidagi ma'lumotlar yetarli darajada emas. Bu dastur bo'yicha o'zbek tilida o'quv uslubiy qo'llanma yaratish va o'quv jarayoniga qo'llash bitiruv malakaviy ishimning yangiligi hisoblanadi.

Bitiruv malakaviy ishimizning amaliy ahamiyati shundan iboratki, bu o`quv uslubiy qo'llanma amaliyotda qo'llanilsa, dasturlash asoslari fanida talabalarda yana bitta milliy tilda uslubiy qo`llanma paydo bo`ladi va ularning imkoniyatlar yanada oshadi.

Bitiruv malakaviy ishimizning metodologik asosini uning tarkibiy qismi, undagi o'quv va didaktik vositalarning jamlanganligidir. Bu o'quv uslubiy qo'llanma ta'lim jarayonini samarali tashkil etishda ham amaliy ahamiyat kasb etadi.

Bitiruv malakaviy ishimizning metodlari. C# tilida strukturali dasturlash bo`yicha bo'yicha o'quv uslubiy qo'llanma yaratish va undan dasturlash asoslari fanlarida qo`llanilishi mumkin.

Bitiruv malakaviy ishining tarkibi va hajmi quyidagicha. Kirish, 2 ta bob, har bir bobning qisqacha xulosasi, adabiyotlar ro'yxati, xotimadan iborat bo'lib jami 92 betga bayon qilingan. Birinchi bobda Adobe InDesign dasturi bilan tanishish, obyektlar, geometrik figuralar, matnlar bilan ishlash bo'yicha ma'lumotlar keltirilgan. Ikkinchi bobda ko'p sahifali hujjat, stillar yaratish va undan foydalanish, kitoblar va kutubxonalar haqida ma'lumotlar keltirilgan. Kirish

qismi 3 betdan iborat, tushuntirish qismi 80 betdan iborat. Bitiruv malakaviy ishida 21 ta adabiyotdan foydalanildi.

I BOB. DASTURLASH TILLARI

1.1. Dasturlash tillarining evolyutsiyasi

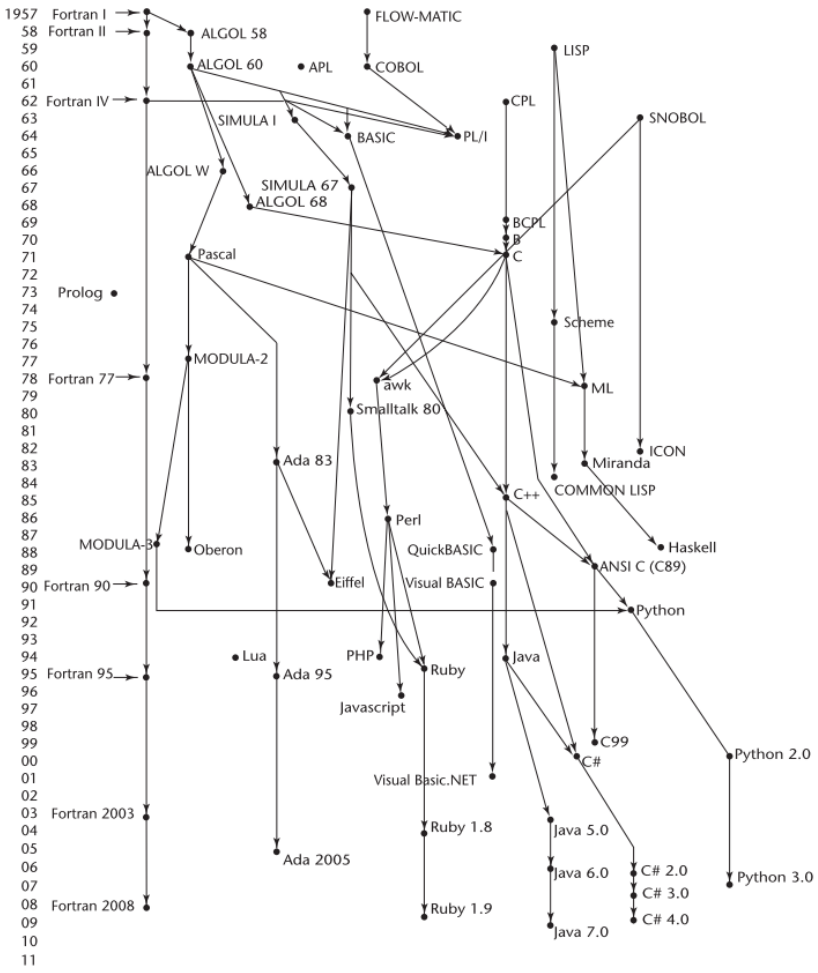
Har bir tilning yaratilgan muhiti o‘rganiladi va tilning imkoniyatlari va rivojlanishi uchun asoslariga e’tibor qaratiladi. Tilning umumiy ta’riflari keltirilmagan, buning o‘rniga, har bir tilning ba’zi yangi o‘ziga hos hususiyatlarini ko‘rib chiqilgan. Kelgusidagi dasturlash tillariga yoki kompyuter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

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

Tillarning dastlabki versiyalari xronologik 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.

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 erda dinamik massivlardan foydalanilmagan [2].

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



1.1.1 – rasm. Dasturlash tillarining evolyutsiyasi sxemasi

Susning “Plankalkul ” tili

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.

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 echish algoritmlarini shu tilda yozgan.

Plankalkul nihoyatda to‘liq, ayniqsa ma’lumotlar strukturasi sohasidagi eng yangi o‘ziga xosliklarga 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 (C 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 etganligi 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.

Susning qo‘l yozmalari 1945-yilgacha yozilgan har qanday dasturlardan qiyin bo‘lgan dasturlardan tashkil topgan. Masalan, sonlar massivini tartiblash, berilgan grafni bog‘langanligini tekshirish, butun va haqiqiy sonlar ustida amallar, kvadrat ildiz va qavslar, 6 xil ustunlikdagi operatorlardan tashkil topgan mantiqiy formulalarni tahlil qila olgan. Ehtimol, u o‘zi yaxshi o‘ynay olmaydigan, 49 betdan ifodat shaxmat o‘yini uning eng mashxur kodidir.

Agar kompyuter olimlari Susning Plankalkuli haqidagi ta’rifni 1950-yildan oldin topganlarida edi, bu tilning amalga oshishi uchun to‘sqinlik qiladigan yagona tarafi bu belgilash bo‘lar edi. Har bir operator 2 yoki 3 qatordan ifodat bo‘lgan. Birinchi qator hozirgi tillardagiga operatorga o‘xshagan. Ikkinchi qator, ihtiyoriy bo‘lib, birinchi qatordagi massiv elementlarining indekslaridan tashkil topgan quyi

skriptdan ifodat bo‘lgan. Huddi shu usulni CHarles Babbeyj ham o‘zining XIX asr o‘rtalaridagi “Analitik mashinasi” da qo‘llagan. Har bir Plankalkul operatoridagi oxirgi qator, birinchi qatorda foydalanilgan o‘zgaruvchilarning tiplari nomlaridan tashkil topgan bo‘ladi. Bunday belgilash birinchi ko‘rganda cho‘chitishi mumkin.

Susning ishlari 1945 – 1950 yillarda keng o‘rganilganida edi, dasturlash tillari qaysi yo‘nalishda rivojlangan bo‘lardi. Agar u tashqi dunyodan uzib qo‘yilgan 1945-yilgi Germaniyada emas, balki boshqa bir tinch mamlakatda, olimlar atrofida o‘z ishini davom ettirganda qanday yutuqlarga erishgan bo‘lishi mumkin edi [3].

Apparativ darajada minimal dasturlash: psevdokodlar

1940yil oxirlarida yaratilgan kompyuterlar kabi qulay bo‘lmagan. Undan tashqari ular ishonchsiz, qimmat, juda sekin ishlaydigan juda kichik xotiraga ega bo‘lganlar. Ularda yordamchi dasturlar yo‘qligi sababli dasturlash juda qiyin bo‘lgan.

Yuqori darajali dasturlash tillari va asembler tillari bo‘lmagani uchun dasturlash mashina kodlarida amalga oshiriladi. Bu juda mashaqqatliish va ko‘p xatoliklarga olib keladi.Muammolardan biri deb guruxlarni ko‘rsatishda butun sondan foydalanishni aytsak bo‘ladi. Masalan: ADD buyrug‘i 14 sonini matn yordamida emas balki bitta belgidan ifodat kod yordamida aniqlashi lozim edi. Bu dasturni o‘qishni ancha qiyinlashtirgan. Asosiy muammolardan biri dasturni murakkablaydigan madifikatsiyani absolyut (to‘liq) adresatsiyasi hisoblanadi. Agar bizning xotiramizda mashina kodi bor deb faraz qilamiz. Bunday dasturlarning ko‘p buyruqlari boshqa dasturdan tashqari xotira yacheykalariga yo‘naltiriladi, ko‘p hollarda birqancha buyruqlar yoki ma’lumotnomolarga yo‘naltiradi.

YAngi buyruqlarni kiritish mavjud dastur tarkibini dasturchi adresga yo‘naltirilgan xamma dasturlarni topib o‘zgartish lozim. Bu muammo buyruqlarni o‘chirishda xam yuzaga keladi.Mashina tilida ko‘p hollarda “ bo‘sh operatsiya” buyrug‘idan foydalaniladi.Unda dasturning umimiy tarkibi buzmasdan o‘chirilgan buyriq o‘rniga qo‘yish mumkin.

Bu standart muammolar xamma mashina kodlarida mavjud bo‘lib, assembler tilida assemblerni yaratishga sabab bo‘lgan. Undan tashqari u paytda dasturlashga bog‘liq bo‘lgan xar bir vazifa asosan hisoblovchi xarakterga ega edi. Ularni hal qilishda nuqtalarni o‘zgaruvchan sonlar uchun arifmetik operatorlarni amalga oshirish talab qilinardi. 1940 yil oxiri va 1950 yilning boshidagi kompyuterlarda bunday imkoniyat yo‘q edi. Bunday kamchiliklarni bartaraf qilish uchun yuqori darajali tilni yaratish lozim edi [2].

Short code tili

Short code tilining yangi nushasi 1949 yili BINAC kampyuteri uchun Djon Mochli tomonidan yaratilgan. Keyingi Short code tili UNIVAC I kompyuteriga o‘tqazilgan va bir necha yil oralig‘ida mashinalarning eng asosiy dasturlash vositasi edi. Short code tili haqidagi ma’lumotlar juda kam bo‘lib, UNIVAC I kompyuter nusnasi uchun dasturlash bo‘yicha saqlangan. Bu ikki nusha bir biriga judao‘xshash edi.

UNIVAC I tilinig so‘zlari 72 bit va 12 baytdan ifodat edi. Short code tili matematik ifodaning kodlangan nusxasidan ifodat edi.

Buyruq 2 bitli miqdorini bildirib ko‘p ifodalar bitta so‘zga joylashtirardi. Short code tili mashina kodiga aylantirmagan edi. interpretator yordamida shakllangan edi. Vaqt o‘tishi bilan bu jarayon avtomatik dasturlash deb nomlandi. U dasturlash jarayonini ancha osonlashtirdi. Short code tili mashina kodi interpretatsiyasidan 50 marta sekinroq amalga oshirilardi.

Speedcoding tizimi

Bu vaqtda boshqa joylarda imperativ tizim ishlab chiqilgan. Bu tizim mashina tilining uning haqiqiy sonlarining harakatiga qo‘shimcha bo‘lib kengaytiradi. Bunga IBM 701 kompyuteri uchun Djon Bekus tomonidan yaratilgan Speedcoding tizimi misol bo‘la oladi. Speedcoding tizimlari interpretetori 701 kompyuterda faoliyat ko‘rsatdi. Bu tizim nuqtalari o‘zgaruvchan 4 ta arifmetik operatsiyani bajaruvchi son sinus, arktangens, eksponentlar. Logorifmlar hisobi, ildiz kvadratini chiqarishni o‘z ichiga olar edi. Bunday tizimni cheklanishini tasvirlash uchun interpretitorni ish bilan ta’minlangandan so‘ng operativ xotirada

faqatgina 700 so‘z qolar edi va ikki son bichimi buyrug‘i uni amalga oshirish uchun 4,2 millisekundlarni talab qilardi. Boshqa bir tarafda Speedcodingindeksiv registorlarning avtomatik holatda kattalashish imkonyatiga ega edi. Apparativ ta’minotda 1962 yilgacha UNIVAC 1107 kompyuterlari yaratilmaguncha bunday imkoniyat yo‘q edi. Bunday vositalar tufayli Speedcoding tizimida 12 ta buyruqni sarflab matritsalarni ko‘paytirishni amalga oshirish mumkin edi. Bekusning aytishicha ikki xaftada amalga oshiradigan vazifani Speedcoding tizimlarida bir necha soatda amalga oshirish mumkin [3].

UNIVAC “kompilyasion” tizim

1951 va1953 yillar o‘rtalarida Greys Xopper boshchiligidagi gurux UNIVAC kompyuteri asosida psevdokod tuzuvchi mashinalashgan kodlar bo‘lgan A-O, A- 1 va A-2 o‘lchamli “kompilyasion” tizimlar seryasini tuzishdi. Bu (dastur) kompilyatorlarda yozilgan psevdokodlarda dastur juda oddiy bo‘lib mashinalashgan kod bilan taqqoslanganda bu dastur allaqachon takomillashtirilgan edi. Bu guruhga bog‘liq bo‘lmagan holda 1952 yilda bu yondashuv Uilkson tomonidan taklif qilingan.

O‘zaro bog‘langan ish

Shu vaqtda dasturlashtirishga oid boshqa dasturlar ham tuzilgan edi. Kembridj universitetida Devid Dj. Uiler absalyut adresatsiyalarning mavsadini qisman hal qiluvchi adreslarni o‘zgartiruvchi bloklarning foydalanuvchi usulini ishlab chiqdi. Bu g‘oyani alohida kichik dasturlarni taqsimlovchi asembler dasturini ishlab chiqishda amalga oshirdi. Bu haqiqatdan ham muhim va fundamental erishuv edi.

IBM 704 kompyuteri va FORTRAN tili

Hisob kitob texnikasida buyuk yutug‘lardan biri bu 1954- yilda IBM 704 kompyuterini yaratilishi bo‘ldi va aynan bu imkoniyat FORTRAN tilini tuzishga sababchi bo‘ldi. SHuni aytish joizki agar IBM korporatsiyasi o‘zining IBM 704 kompyuteri va FORTRAN tili bilan bo‘lmaganda ham boshqa biron bir tashkilot o‘zining kompyuteri va yuqori darajadagi tili bilan bo‘lgan bo‘lar edi. SHu bilan

birga nafaqat hodisalarning rivojlanishini ko‘ruvchu balki bu ish bilan shug‘ullanishga resurslarni topa olgan IBM korporatsiyasi birinchi bo‘ldi.

Fortran tilini ishlab chiqish haqidagi rejalar IBM 704 tizimini 1954- yilning may oyida tuzishdan oldin bor edi. IBM korparatsiyasida ishlovchi Djon Bkus va uning guruhi 1954- yilning noyabr oyida “IBM matematik formulalar translyator tizimi: Fortran” mavzusi asosida hisobotni nashrdan chiqarishdi. Bu hujjatda Fortran tilining birinchi Fortran 0 deb ataydigan nusxasi chop etilgan va Fortran tilining rivojlanishida katta yordam bo‘lgan. Ilmiy ishda yozilishicha Fortran tili boshqa dasturlar kabi samarali, unda qo‘lda foydalanishi, dasturlash ham tizimida interplitatsiya qilingan psevdo kodlar kabi qulay. Hujjatda dasdiqlanishicha Fortran tili dasturlashdagi xatoliklarni ham bartaraf qiladi. Bu ma’lumatlarga asoslanib Fortran tili birinchi kompilyatorda faqatgina sintaksisga oid xatolarni aniqlay olgan. Fortran tili quyidagi holatlarda ishlab chiqilgan:

1. Kompyuterlarning hammasi hali kichik, sekin ishlovchi uncha ishonchli emas edi.
2. Kompyuterlar asosan ilmiy hisob kitoblarda foydalanilar edi.
3. Dasturchilarning ish haqqilaridan tashqari kompyuterlarning yuqori narhidan kelib chiqib FORTRAN tilining birinchi kompyuterlarining asosiy vazifasi generativ kodni bajarishda yuqori darajadagi tezligi edi [2].

FORTRAN tilining dastlabki tushunchasining tasnifi shu holatda rivojlangan.

FORTRAN I tili haqida qisqacha ma’lumot

FORTRAN 0 tili 1955- yilning yanvar oyidan 1957- yilning oralig‘i ichidagi rivojlanish davrida holati ancha o‘zgardi. Biz rivojlangan til deb nomlaydigan FORTRAN I tili boshqarmasi foydalanuvchining “Programmer’s Reference Manual” da yozilgan va 1956- yilning oktyabr oyida nashr etilgan FORTRAN tilida o‘zgaruvchan ismlar 6 ta belgidan (simvol) ko‘p bo‘lmasligi kerak (FORTRAN 0 da 2 ta) foydalanuvchi tomonidan aniqlanuvchi kichik dasturlar shartli operator IF va DO siklining alohida kompilyasiya qila olmasdi.

FORTRAN 0 tilida operator algebraik formulalarida yoziladigan, mantiqiy IF operatoriga ega. Masalan: “ko‘p” munosabatni bildirish uchun “>” belgisi ishlatilgan. IBM 704 kompyuterining alifbosiga “>” belgisi bo‘lmagani uchun keyinchalik bu munosabat operatorlaridan voz kechmoqlikkerak edi. Mashina 3 xil yacheyka xotirasidan kata bo‘lgan tarmoqlanishga ega edi, shuning uchun IF mantiqiy operator shakllariga ega bo‘lgan arifmetik ifoda bilan almashtirilgan edi.

IF (arifmetik ifoda) N1, N2, N3. Bu erda N1,N2,N3 operator belgilarini bildiradi.Agar ifoda qiymati manfiy bo‘lsa belgi N1 ga o‘tardi, agar u nolga teng bo‘lsa belgi N2 ga, musbat bo‘lsa belgi N3 ga o‘tardi. Bu operator hozir ham FORTRAN tilining bir qismi hisoblanadi. FORTRAN I tilining sikl operatori quyidagi tarkibdan ifodat:

FORTRAN I tilining hamma boshqaruvchi operatorlari IBM 704 kompyuterining buyruqlariga asoslangan edi. Hozirda FORTRAN I ning IBM 704 kompyuteri konstruktoriga bog‘liq boshqaruvchi strukturasining shakli bor bo‘lganligi yoki FORTRAN I tilining mutahasislari bu shakldagi buyruqni IBM 704 kompyuterida foydalanishni taklif qilganmi bu aniq emas.

FORTRAN tilining dastlabki yutug‘i 1958- yilning aprel oyidagi ma’lumotda aks ettirilgan.

FORTRAN II haqida qisqacha ma’lumot

FORTRAN II kompyuteri 1958 –yil bahorida keng tarqalishni boshladi. U FORTRAN I tili kopilyasiyasi tizimini ko‘p xatolarini to‘g‘irladi va undan tashqari tilga bir necha mihim bositalarni qo‘shdi. Ularning ichida eng muhim vosita bu dasturning mustaqil kompilyasiyasi edi. Bunday vositaning yo‘qligi sababli har bir o‘zgartirish barcha programmalarni qayta kompilyasiya qilishni talab qilar edi. FORTRAN I tilida mustaqil kompilyasiya yo‘qligi IBM 704 kompyuterining ishonchli emasligi bilan birga dasturlarning ko‘lamini 300-400 qatordan ko‘p bo‘lishini cheklardi. Ko‘p ko‘lamli dasturlar kompilyasiyani amalga oshirishda kam imkoniyatlarga ega edi. Dasturda kichik dasturlarni kiritish mashinaga oid tillarda kompilyasiya jarayonini qisqatirar edi.

FORTRAN IV, FORTRAN 77 va FORTRAN 90 tillari

FORTRAN III tili ishlab chiqilgan , lekin keng tarqala olmagan. FORTRAN IV esa u paytda dasturlash tilida juda keng foydalanilgan. Bu tilning rivojlanishi 1960- yildan 1962- yilgacha FORTRAN 77 tili chiqqunga qadar tilning standart nusxasi bo‘lib qolgan. Ko‘p ma’lumotlarga ko‘ra FORTRAN IV FORTRAN II dan ko‘ra ancha avzallikga ega bo‘lgan.

FORTRAN 77 tili FORTRAN IV tilining ko‘p vositalarini qo‘llab quvvatlagan, undan tashqari unda belgilar qatorini ishlovchi sikllar yordamidagi mantiqiy operator boshqaruvi, ELSE operatori IF tarkibida ishlatish imkoniyatlari bo‘lgan.

FORTRAN (ANSI, 1992) tilining keying nucxalaridan biri FORTRAN 90 hisoblanadi. FORTRAN 90 tili FORTRAN 77 tilidan ancha farq qiladi. Eng muhim farqlari quyida qisqacha keltirilgan.

Massivlar bilan harakatni amalga oshiruvchi funksiyalar. Ularning tarkibiga: POTPRODUCT, MATMUL, TRANSPOSE, MAXVAL, MINVAL, PRODUCT va SUM kiradi. Ularning bajaradigan vazifalarini nomlanishidan bilish mumkin. Bu ifodali va samarali funksiyalargan bir nechtasi xolos.

Agar massivlar ALLOCATABLE sifatida e’lon qilingan bo‘lsa, ular buyruq bo‘yicha dinamik holda xotiraga joylashadi va o‘chadilar. FORTRAN tilining oldingi nusxalaridan farqli uning oxirgi imkoniyati radikal o‘garishidir.Ko‘rsatgichlar ham FORTRAN 90 tilining bir qismi hisoblanadi.

Yangi CASE ko‘p variantli tarmoqlanish operatori , EXIT barvaqt chiqib ketishi uchun, uzatish boshqaruvi sikldan turib uning boshiga o‘tishi uchun SYCLE operatori boshqaruv operatori qo‘shilgan edi.

Kichik dasturlar rekursiv yoki uncha muhim bo‘lmagan va muhim parametrlarga ega .

Ada tili tarmog‘iga o‘xshash modullardan foydalanuvchi imkoniyatlar qo‘shilgan edi. Modullar o‘z ichida ma’lumotlar va kichik dasturlar haqida ma’lumotlarga ega bo‘lishi mumkin.Ularning har biri tashqi aloqani boshqarish

uchun spetsifikatorlar (yordamida) PRIVATE yoki PUBLIC yordamida tavsiflanishi mumkin.

FORTRAN 90 tilini tavsiflashda oldingi xossalariga tegishli til vositalarini o‘chiruvchi yangi konsepsiya paydo bo‘lgan. FORTRAN 90 tili FORTRAN 77 tilining barcha barcha vositalariga ega edi va unda ikkita visitalar ro‘yxati tuzilgan edi. Bu ro‘yxat FORTRAN ning keyingi nusxalarida aks ettirilishi lozim edi. Bular IF arifmetik operator va GOTO belgilangan operator. Foydalanishini tavsiflanmagan vositalar ro‘yxatida FORTRAN 90 tilini davom ettiruvchi uning ikkinchi nusxasida undan bos kechish rejasi tuzilgan edi. Bu ro‘yxatga COMMON, Equi VALENSE operatorlari hisoblovchi GOTO operatori va undan tashqari operator funksiyalar bor edi.

FORTRAN tilini birinchi nusxasining mutahasislarining aytishlaricha tilni ishlab chiqarishdagi vazifalari uchun kerak.Undan tashqari FORTRAN tili nafaqat IBM kompyuterlarida balki boshqa korporatsiyalarda ham ishlashi mumkinligi ularning hayoliga ham kelmagan edi.Haqiqatdan ham ular IBM 704 korporatsiyasi tomonidan yaratgan boshqa mashinalarga kompilyatorlar o‘ylab topishga majbur edilar, shuning uchun ham IBM 704 kompyuterining davomchisi IBM 709 FORTRAN tilidan oldin yaratilgan edi va bu IBM 701 kompyuter uchun moslashtirilgan edi. SHuni aniq aytish mumkinki faqatgina FORTRAN tili haqiqatdan ham kompyuterlarda foygalangan edi qolgan tillar esa o‘zining kelib chiqishi dasturini har xil sohalarda ko‘rib chiqqan.

FORTRAN I va uning davomli vositalaridan biri FORTRAN 90 dan tashqari bu yuqori optimizatsiya qilingan kopilyatorlarni yaratishga imkon yaratilishi va bu dasturlarni amalga oshirishi ularni xotirasida muxirlanib borishidir. FORTRAN tilining oldingi nusxalarida yaratiladigan dasturlar o‘zining kelib chiqishidan hisoblovchi bo‘lgan. FORTRAN tilini global yutug‘ini qaytadan baholash qiyin u haqiqatdan ham kompyuterlardan foydalanish usullarini keskin o‘zgartirdi. Bu uning birinchi yaratilgan va keng foydalanuvchi yuqori darajali til bo‘lgani bilan bog‘liq. Keyinchalik yaratilgan til va konsepsiyalar FORTRAN ni dog‘da qoldira oladi. Ford tomonidan 1910 yilda yatilgan “Model T” avtomobili

1998-yilda yaratilgan “Ford Mustang” dan ancha orqadaligi shunga misol bo‘la oladi.Uning kamchiliklariga qaramay FORTRAN tiliga juda katta investisiya qo‘yilgan va unga yuqori darajali foydalanuvchi tillar ichidan joy bergan [2].

Alan Perlis ALGOL 60 tili yaratuvchilaridan biri 1978- yil shunday degan: “FORTRAN tili-bu kompyuter dunyosining *lingua franca* sidir ” . U yashagan va yana yashaydi, chunki u kommersiyachilik faoliyatida juda muhim foydali bo‘lishi uchun yaratilgan” (Wexelblat, 1981) quyida FORTRAN 90 tilidagi xujjatlar misol qilib ko‘rsatilgan.

Funksional dasturlash : LISP tili

Birinchi funksional dasturlash tili ro‘yxatlarni tuzishga til vositalariga yordam sifatida yaratilgan. Uni yaratishga zarurat birinchi suniy intellekt soxasidagi ishlar yaratishda tug‘ildi.

Suniy intellekt sohasidagi ishlar manbai va ro‘yhatlar tuzish

Suniy intellektga qiziqish 1950- yilda har xil sohalarda paydo bo‘ldi. Unga ehtiyoj asosan tishunoslikda va qisman falsafa, matematikada bo‘ldi. Tilshunoslar tabiiy tilda matnlarni yaratishga etibor qaratishdi. Psixologlarni inson (modellashtirishi) eslab qolish va xotiralarni eslash qobilyatini modellashtirish , undan tashqari inson tafakkurini fundamental jarayonlari qiziqtirdi. Matematiklar teoremalarni isbotlash sifatida fikr mulohaza jarayonini mexanizmlarini o‘rganishdi.Bu hamma tadqiqotlar bir xil xulosaga kelishdi: bog‘liq ro‘yxatlar sifatida saqlangan kompyuterlarni belgili ma’lumotlarni ishlab chiqishiga keltiradigan biroq xil usullar tuzilgan bo‘lishi kerak. SHuni belgilash kerakki u vaqtda hamma hisoblar miqdoriy ma’lumot kelib chiqilgan va massiv shaklida saqlangan.

Ro‘yxatlar tuzish konsepsiyasi Allen Nyuel Dj.Shou va Gerbert Saymon tomonidan ishlab chiqilgan. U dastlab klassikaga doir Logical Theorist dasturi yani dastlabki suniy intellekt dasturidan biri haqidagi ishda nashr etilgan. IRL-I nomli til xech qachon tadbiq qilinmagan.Uning keying nusxasi IRL-II Rand Corporation

korporatsiyasidagi Johnniac kompyuterida amalga oshirilgan edi. IPL tilining rivojlanishi 1960- yilgacha davom etgan. U paytda IPL-V tilining tasnifi nashr etilgan. IPL ning keng tarqalmaganiga uning darajasining pastligi sabab bo‘ldi. Gipotetik kompyuterga ro‘yxatlar tuzish buyrug‘i reamuatsiya qilingan va bu tillar assembler tillar bo‘lganlar. Ularning dastlabki reamuatsiyasi Johnniac mashinasida qilingan va IPL oilasiga tegishli tillar keng tarqalmagan.

IPL oilasiga mansub tillar merosi dasturlash tillar rivojlanishi ularning ruxiatli tarkibi va aniq demonstratsiyasi edi, shuning uchun ro‘yxatlarni tuzish mumkin va foydali hisoblanadi.

Funksional dasturlash jarayonlari

LISP tili funksional dasturlash tili sifatida yaratilgan. Funksional dasturda hamma hisoblar funksiyaning argumentga aylanishi yo‘li orqali ishlab chiqiladi. Na operatorlar na imperitiv tillardagi o‘zgaruvchanlik funksional dasturlash tilida yozilgan dasturlarda kerak bo‘lmaydi. Undan tashqari interativ jarayon rekursiv funksiyalar chaqiruvchi orqali aniqlanadi. Funksional dasturlashning bu asosiy konsepsiyasi imperative tillardagi dasturlashdan farq qiladi.

LISP tilining sintaksisi

LISP tili imperativ tillardan farq qilishi shundaki u funksional til hisoblanadi va unda yozilgan dasturlar FORTRAN yoki C sintaksis tillarida yozilgan dasturlardan farq qiladi. Til sintaksisi ingliz tili va algebralarning qiyin aralashma (qorishma) hisoblanib, til sintaksisi LISP – oddiylik etaloni hisoblanadi: (A B C D)

LISP tili asrning chorak qismidan beri suniy intellekt soxasida juda baland ko‘tarilgan va hozirgacha bu soha keng tarqalgan. LISP tilidagi ko‘pgina sabablari bartaraf etish samarali emas. Skampilyasilangan ko‘p zamonaviy realizatsiya va dasturlarning rezultatsiyasidan ko‘ra tezroq amalga oshiriladi. Suniy intellekt sohasidagi yutuqlardan tashqari LISP tili funksional dasturlashning etakchisi hisoblanib, o‘zining qatiy faoliyatini dasturlash sohasidagi til sohasidagi kabi isbotladi. Birinchi bo‘limda ko‘rsatilganidek dasturlash til ko‘p tadqiqotlarni

takidlashicha funksional dasturlash dasturlashni ta’minlashda imperative tillardan foydalangandan ko‘ra yaxshiroq to‘g‘ri keladi.

1970 va 1980 – yillar o‘rtasida LISP tilining har xil ko‘pgina dialektlari ishlab chiqilgan. Bu esa o‘tqazuvchanlik muammosiga olib keldi. Bu yuzaga kelgan jarayonni bartaraf etishda COMMON LISP tuzilgan.

LISP tiling ikkita avlodi

Bizning davrimizda LISP tilining ikkita dialekti - COMMON LISP va Scheme tillari. Ikkala til ham keyingi bo‘limlarda ko‘rib chiqilgan.

Scheme tili

Scheme tili 1970- yilning o‘rtalarida MIT institutida ishlab chiqilgan. Unga ma’lumotlarning statistik foydalanishida kichik o‘lchamdan foydalanish va birinchi klass ob’ektini qayta ishlashi xosdir. Scheme tili vazifasi shu holatda ifodaning ma’nosi va ro‘yxatlarda elementi bo‘lishi mumkin. LISP tilining oldingi nusxalarida bu hamma imkoniyatlar bo‘lmagan ma’lumotlarning statistik qisqaqacha ma’lumoti foydalanilmagan kichik bo‘lmagan til kabi oddiy sintaksiz va semantik Scheme tili funksional dasturlash kurslari va dasturlashning kirish umumiy kurslari kabi shunday ilmiy takliflarda to‘liq yaroqlidir. Scheme tili 14 bo‘limda tasvirlangan.

COMMON LISP tili

COMMON LISP tili 1980- yilning boshida ishlab chiqilgan LISP bir necha xil dialektiga xos xususiyatlarni bitta tilga birlashtirishga urinishda tuzilgan. COMMON LISP tilining shunday kombinatsiyasi katta va qiyin strukturaga ega. Uning asosi LISP tili hisoblanib uning sintaksisi asosiy funksiyasi va prinsipial tabiati bu tildan kelib chiqadi.

Ma’lumotlarning qisqacha dinamik holatini ta’minlovchi COMMON LISP tili mutahasislari ikkala usuldanam foydalangan. Statistik o‘zgaruvchan ma’lumotlardan foydalanishda special o‘zgaruvchan sifatida e’lon qilinsa uning ko‘rinishi sohasi dinamik bo‘lib qoladi.

COMMON LISP tili ko‘p sonli turlar va ma’lumotlar strukturasidantashkil topgan. Ularning ichida massivlar, yozuvlar, belgilar qatori va kompleks sonlar

mavjud. Undan tashqari unda funksiya va ma’lumotlar yig‘inida foydalangan paketlar shakli mavjud bo‘lib ular aloqani boshqarishni ta’minlaydilar.

Qardosh tillar

ML tili dastlab 1980- yilda Edinburg universtetida Robin Milner tomonidan Logic for Computable Function nomli verifikatsion dastur tizimi metatili sifatida yaratilgan. ML tili asosiy tarzda bu funksional dasturlash tili bo‘lib imperativ dasturlashni ham qo‘llab quvvatlaydi. ML vazifasi tilda imperativ tillardan ko‘ra ancha umumiy hisoblanadi. Ular kichik dasturlar ichida parametrlar sifatida o‘tqaziladi va polimorf ham bo‘lishi mumkin. LISP va Scheme tillaridan farqliroq ML tilining har bir ifodasi kompilyasiya jarayonida aniqlanishi mumkin. Turlar ob’ektlar bilan bog‘lanadi ismlar bilan emas. Mantiqiy ifodalar turlari kontektdan ifodasini oshirishi 6- bo‘limda tasvirlangan.

LISP va Scheme tillaridan farqli ML tili ifodasi bilan birga kelib chiqqan qavsli funksional sintaksisdan foydalanilmaydi. ML tilining sintaksisi Pascal va S turdagi imperativ tillar sintaksisiga o‘xshaydi.

Miranda tili 1980- yilning boshlarida Buyuk Biritanyadagi kenterberining Kenta universtetida Devid Terner tomonidan ishlab chiqilgan. Bu til qisman ML, SASL, va KRC tillariga asoslangan. Miranda tili o‘zlashtirish operatori o‘zgaruvchan tarkibga ega bo‘lmagan aniq funksional til hisoblanadi. Noskell tilining farqli xususiyatlaridan biri uning “leniboy” hisoblagochidir. Hech qanday ifoda uning ma’nosini talab qilinmay turib hisoblanmaydi.

ML va Naskell tillari keyingi ma’vzularda keltirilgan.

ALGOL 60 tilini mukammallashtirishga birinchi qadam

ALGOL 60 tili undan keying dasturlash tillaring rivojiga ko‘zga ko‘rinarli darajada ta’sir ko‘rsatgan va har qanday til haqidagi ma’lumotlarda muhim o‘rin tutadi.

ALGOL 60 tili – bu universal tilning tuzilishiga xarakat 1954- yilda Lening va Serler algebraik tizimi hammaning nazariga tushdi va FORTRAN tilida birinchi hisobot nashr etildi. FORTRAN tili 1957- yilda yaratilgan, shu yilda birqancha yuqori darajali tillar ham yaratilgan. Ularning ichida eng mashxur tillar bu

koernigitexda Alan Perlis tomonidan yaratilgan IT tili va UNIVAC kompyuterlari uchun yaratilgan MATH-MATIC va UNICODE. Dasturlash tillarining sonining o‘sishi foydalanuvchilar o‘rtasidagi aloqani qiyinlashtiradi. Undan tashqari hamma yangi tillar yoki UNIVAC kompyuterlari uchun yoki IBM 700 korporatsiyasi uchun yaratilar edi.

1955- yilda amaliy matematika va mexanika jamiyati hamma turdagi kompyuterlar uchun yakka universal algoritmik tilni yaratishdi. 1957- yilda AQSH da paydo bo‘lgan yuqori darajali tillardan deb GAMM jamiyati komissiyalarni ishontira oldilar va undan keyin 1858- yil Frits Bauer ASM farmal rejasi bilan hammani tanishtirdi.

GAMM jamiyati va ASM assatsitasi qo‘shma konstruktor ishni kengashga olib chiqib har bir guruhdan 4 tadan vakil yuborishlarini aytishdi. Bu uchrashuv 1958-yilning 27- mayidan 1- iyungacha Syurixeda yangi tilning quyidagi maqsadlaridan shakllandi.

• Til sintaksisi umumiy tanlangan matematik ifodalar dasturlariga maksimal darajada yaqin bo‘lishi va undan yozilgan barcha hujjatlar kichik qo‘shimcha shoxlarsiz oson o‘qishi lozim.

• Til hisob jarayonlarini gazeta jurnallarda yozib borish uchun foydalanilishi lozim.

• Yangi tilda yozilgan dasturlar mexanik tarzda mashina tiliga o‘girilgan bo‘lishi lozim.

Birinchi maqsad tilning ilmiy dasturlashdagi imkoniyatlarini bildiradi.

Ikkinchi maqsad kompyuter sohasida yangi edi. Oxirgi maqsad har bir til uchun juda qiziq edi.

Syurixedagi uchrashuv muhim natijaga yoki ayovsiz baxslarga olib kelgan. Haqiqatdan ham uchrashuv nafaqat alohida insonlar o‘rtasida va ikki taraf o‘rtasida ham kompraislarga olib keldi.

ALGOL 58 tili haqida qisqacha ma’lumot

Syurixedagi uchrashuvda yaratilgan til IAL deb nomlangan konstruktorlik ishi jarayonida esa unga ALGOL deb nom berishgan. Lekin bu ism komitetning

xalqaro tarkibini ko‘rsata olmagani uchun o‘zgartirilgan. Keyingi yili bari bir ham ALGOL ga o‘zgartirilgan va til ALGOL 58 nomini olgan.

Ko‘p parametrlarga ko‘ra ALGOL tili FORTRAN tilining davomchisi hisoblanadi. Unda FORTRAN tilining ko‘p vositalari jamlangan va bir qancha yangi konstruksiya va konsepsiyalar qo‘shilgan.

ALGOL 58 tilida ma’lumotlar turlarining konsepsiyasi shakllangan. Unda ko‘p tillarda bo‘lganidek tashkiliy operatorlar g‘oyasi ishlab chiqilgan. ALGOL 58 tilida o‘zlashtirish tilining bir qancha shakllantirilgan Plankalkul Suz tilida o‘zlashtirish operatorining quyidagi formasi ishlatilgan.

ifoda =>(qiymat)

Keyin Amerikaliklarning qatiy talablaridan so‘ng u quyidagicha shaklni oldi:

ifoda :=(qiymat)

Evropaliklar buning teskari shaklini afzal ko‘rishdi.

1958-yilning dekabr oyida ALGOL 58 tili haqidagi hisobot katta e’tibor bilan kutib olingan. AQSH da yangi til strukturasining dasturlash tili sohasida g‘oyalar to‘plami bo‘ldi. Bu hisobot dasturlash tilida 3 ta asosiy ishlar ichidan joy oldi. SHatning Mechigan universtetida MAD tili yaratildi. Amerikali xarbiy dengiz elektiron guruhi tomonidan NELIAC tili yaratildi tizimilarni yaratish korporatsiyasi System Devolopment Corparation tomonidan JOVIAL tilini yaratishdi. U ALGOL 58 tilining yagona nusxasi hisoblanib JOVIAL tilni yaratuvchisi sifatida keng tarqalgan.

JOVIAL tili AQSH da keng foydalanilgan.

ALGOL60 tilining yaratilish jarayoni

1959- yilgacha ALGOL 58 tili Evropa va AQSHda juda keng tarqalgan. 1960- yilda komitetning ALGOL tilini yaratish bo‘yicha ikkinchi majlisi Parijda bo‘ldi. Bu majlisda 80 takliflar ko‘rib chiqilishi lozim edi. SHu bilan birga ALGOL tilining yaratilishiga Danyalik Piter Naur SYURIX guruhining a’zosi bo‘lmaganligiga qaramay uning bu tilga chuqur qiziqishi bor edi. Aynan Naur ALGOL Bulletinni yaratdi va topshirdi. U ko‘p vaqt ichida Bekausning ishlarini

o‘gandi va u shakldan 1960- yildagi majlis natijasining formal yozuvida foydalanish mumkin degan hulosaga keldi.

1960- yildagi majlis bor yo‘g‘i 6 kun davom etishiga qaramay ALGOL 58 tiliga ancha o‘zgartirishlar kiritilgan edi. Ularning ichida quyidagilar bor edi:

- Blok tizimining konsepsiyasi yaratilgan. Bu dasturlarni yangi ma’lumotlar kiritib dastur qismlarini tarqalib ketmasligiga yordam berdi.
- Ikki usul orqali kichik dasturlarga parametrlarni uzatish imkoniyati yaratildi, bular : ma’nosi va ismi bo‘yicha uzatish
- Rekursiv protseduralarni tuzish imkoniyati yaratildi. ALGOL 58 tili tasnifi bu savolda uncha aniq yoritilmagan edi.
- Avtomatik massivlar yaratildi. Massiv o‘lchami uning xotirada aralashgan vaqtda o‘lchanadi. Bu massiv dasturlarini amalga oshirish jarayonida sodir bo‘ladi.

ALGOL 60 hisoboti 1960- yilning may oyida nashr etildi.Tilni tasnifida hali ham ko‘p noaniqliklar bor edi va 1962- yilning aprel oyida Rimda komitetning uchinchi majlisi natijalari “Revised ALGOL60” da nashr etildi.

Bir parametr bo‘yicha ALGOL 60 ning yutug‘i hayratlanarli darajada bo‘lsa, ikkinchi parametr buyicha uning katta muvoffaqqiyatliligi ko‘rinadi.

1960-yildan so‘ng yaratilgan dasturlashning har bir imperativ tili ALGOL 60 tilidan nimanidir o‘zlashtirishgan. Ularning ko‘pchiligi uning davomchisi bo‘lgan masalan PL/I, SIMULA 67, ALGOL 67,ALGOL 68, S, Ada, C++ va JAVA.

ALGOL 58, ALGOL 60 tillarini yaratish jarayonida ko‘p narsalar birinchi qilingan. Xalqaro guruh dastlab dasturlash tilini yaratishga harakat qildilar. ALGOL tili birinchi mashinaning mustaqil tili bo‘lgan. U yana formal yozilgan sintaksisli til bo‘lgan. Qo‘shma SHtatlarda ALGOL 60 tili hech qachon keng tarqalmagan. U hatto Evropada ham ustun bo‘lgan til bo‘lmagan. ALGOL 60 tilini Rutishauzer shakllantirgan. Xulosa qilib aytgan FORTRAN tilini foydalanish sohasida mustaxkamlangan ko‘p muammolar bor bo‘lgan va IBM korporatsiyasi tomonodan ham qo‘llab- quvvatlanmagan.

ALGOL 60 tilining ustida ishlangan har doim noaniqlik bu tilning tasnifini bir qismi bo‘lgan quyida ALGOL 60 tilidagi dasturlarga misol keltirilgan.

COBOL tilida savdo sotiq yozuvlarini kompyuterlashtirish

COBOL –tili tarixi noodatiy. Odatda ko‘maklashadi, boshqa u yoki bu hollar programmalashtirish strukturasi anchayin mukammallashgan. PL- tili bundan mustasno.

U hozirgi kunda eng keng qo‘llaniladigan programmalar tili hisoblanadi. Eng asosiy sababi, COBOL tilini oson o‘zlashtirish mumkin undan keyin kompyuterda ishlash uchun gaplarning turlari juda kam, lekin ohirgi paytlarda COBOL tilining imkoniyatlari biroz kengaygan. Bu tilning keng foydalanishlarining yana bir sababiuni nomlash oson, negaki kompyuterni xotirasiga o‘rnatish oson bo‘lganligi sababi oxirgi 15 yil ichida kichik bir sohasiga ham kirib kelgan.

COBOL tili ALGOL 60 tili bilan bir paytda paydo bo‘lgan, bu til ham insonlar orasida ishlatiladi, qisqa muddat ichida ma’lumotlarga ega bo‘lish uchun.

1959- yilda unchalik mukammal bo‘lmagan kompyuter tili bo‘lgan, bundan bir necha yil avval esa FORTRAN tili ishlatilgan 1957- yilda esa ancha rivojlangan kompyuter darslari uchun qo‘llaniladigan FLOW_MATIS tili yaratilgan, u bir ishlab chiqaruvchi tomonidan ishlangan bu UNIVAC kompaniyasi bo‘lgan kompyuterlar uchun aynan shu kompaniya ishlab chiqgan. A yana boshqa tili AIMACO, BBS AQSH da foydalanilgan, lekin u FLOW_MATIC tilining bir turi edi. IBM korparatsiyasi til programmalarida kompyuter gaplari uchun COMTRAN (SOMmercial TRANslator – kompyuter tarjimani) dasturini ishlab chiqishdi, lekin bu til xozirgi vaqtgacha qo‘llaniladi. SHunday qilib yana bir qancha kompyuter tillari ishlab chiqilgan.

FLOW – MATIC tili

FLOW_MATIC tilining paydo bo‘lishi qisqa suxbatlarga sabab bo‘lgan. Bu til COBOL tilining paydo bo‘lishiga zamin yaratgan. 1953-yil dekabr oyida Jpeys Xopper UNIVAC Remington_Rand kompaniyasi bir necha gaplar tuzilmasini yaratgan. U matematik programma bo‘lib, matematik amallarni bajarishda qulayliklar yaratadi, yana bu programma ingliz tilida yozadi. 1953- yilda programmachilarni xayratda qoldirib ingliz tilida ishlaydigan kompyuterlar ishlab

chiqildi. Faqatgina 1955- yida UNIVAC kompaniyasi tomonidan yanada imkoniyatlari kengaytirilgan, lekin o‘shanda ham oxirgi xulosaga kelish uchun sinov ob’ekti kerak bo‘lgan. Avval ingliz kalit so‘zlaridan foydalanishgan, keyin fransuz tilidan, undan keyin esa nemis tilidan. UNIVAC kompaniyasi boshchiligida yana bir qancha imkonoyatlar yaratilgan va faktlar echimi esa Xopper rejasi nomi ostida chaqirilgan.

1- mukammal kompyuterlar uchun tillar to‘plami AQSH hukumati tomonidan Rentagoneda 1959- yida 28- va 29- mayda amalga oshirilgan. Ularning fikricha qo‘shilgan hamda yangi kompyuter tili CBL nomi ostida ishlab chiqilgan.(Sommo business Language umumiy kompyuter tili) yana bir qancha xarakterlari ham ishlab chiqilgan. Yuqori darajada ingliz tilidan foydalanishni ko‘pchilik maqullashdi va keng miqqiyosida matematik belgilarni ifodalashda ishlatila boshlangan.

Kompyuterda programmalashtirilgan til oson hisob-kitobga qulay va keng miqqiyosida ishlatila boshlandi va nihoyat bu loyihani rivojlantirishda muammolar bilan chegaralanib qolmaslik kerak.

Bir qancha katta ishlar amalga oshirildi, mukammal til asta –sekin rivojlanib bordi. RSA va Sylvania kompyuterlar tijoratiga oid loyixasini ishlab chiqdi. Bu tilni programmalashtirishda qisqa vaqt ichida o‘rganish uchun Short Range Committee tuzib chiqilgan.

Avval til turlari 2 ta kategoriyaga ajratilgan.

Biri ma’lumolarni yoritib bersa ikkinchisi esa bir qancha amallarni bajaradi.Bu ikki kategoriya ham programmalarning xar hil qismlari hisoblanadi. Indeks amallarini bajarishaga mo‘ljallangan yana bir programma Short Range Committee ishlab chiqilgan.Ko‘plab tajribalardan kelib chiqib aytish mumkinki – indeks tushunchasi mutaxasisligi matematika bo‘lmagan insonlar uchun anchayin qiyinlik qiladi.

Asosan bu matematik amallarni bajarish jarayonida ishlatilgan. 1959- yil dekabr oyiga kelib bu dastur anchayin mukammallashtirilib imkoniyatlari kengaygan keyinchalik esa bu COBOL 60 tili deb nomlangan.

COBOL 60 tiling dastlabki boshqarma tizimi tipografiyasi 1960- yilning aprel oyida Government Printing Office tizimi ishlab chiqilgan va u “boshlang‘ich” deb nomlangan. 1961 va 1962- yillarda uning o‘zgargan versiyasi Department of Defense ishlab chiqilgan. 1968- yilda esa AQSHning milliy standartlashtirish institute ma’lum bir normaga solingan kompyuter tilini yaratishdi. Keyingi ishlangan 2 ta versiyasi bu kompyuter tilini aynan shu institute tomonidan 1974 va 1975- yillarda yana qayta ishlandi. Xozirgi kungacha bu tilning rivojlanishi davom etmoqda.

COBOL tilining paydo bo‘lishi oxir oqibat yana bir qancha kompyuter tillaring yuzaga kelishiga asos bo‘ldi. Masalan makrosov uchun birinchi eng yuqori darajali bo‘lgan til COBOL tilidagi DEFINE fe’lidan olingan yani “yoritish, ifodalamoq” ma’nosini bildiradi.Eng muhim faktlardan yana biri bu: birinchi bor Plankalkul tili ham COBOL tilining rivijlanishidan kelib chiqqan. COBOL tili birinchi kompyuter tili bo‘lib, unda bir qancha uzun nashrlar bo‘lib 30 ga yaqin belgilari bor va ular so‘zlar bilan birgalikda qo‘llaniladi.

Yana bir dastur tillaridan biri bu COBOL tili singari BASIC tili hisoblanadi.

BASIC tili COBOL tilining oldingi versiyasi hisoblanadi.

BASIC tili juda mashhur dasturlash tili bo‘lib, 1979- yilning oxiri 1980- yilning boshlarida vujudga kelgan. Bu tilni o‘rganish oson lekin, fan bilan bo‘linganda uni uncha ko‘p bo‘lmagan shiva so‘zlari bo‘lib, xotirasi ancha kam. BASIC tilining mukammal shakli 1990- yilning boshlarida Visual BASIC nomi ostida chiqdi.

BASIC tili buyruqlarini boshlash uchun umumiy to‘plamlar yig‘indisi bo‘lib, Dartmutskom kollejida ishlab chiqilgan. Dartmouth Solleje, Jonom Kesheni va Tomosom Kursom 1960- yilning boshlarida FORTRAN va ALGOL 60 tilida bir qancha dialektlarni yaratilgan. Ularning talabalari tabiiy fanlarni o‘rganishgan ularga bu til bilan ishlash qiyin bo‘lgan. Bu talabalar odatda tabiiy va texnika fanlarini o‘rganishgan,lekin atigi 15% o‘zlashtira olishadi. 1963- yilning baxorida yangi kompyuter tili ishlab chiqiladi, eng muhimi talabalar gumanitar fanlarni

o‘rgana boshlashdi. Kompyuterning yangi tilida ko‘plab terminlardan foydalanish kerak edi. Bu tilning rivojlanish bosqichlari davom etmoqda.

1. U o‘rganishi uchun oson bo‘lishi va talabalar tabiiy fanlarni o‘rganishda foydalana oladigan bo‘lishi kerak.
2. U yoqimli va do‘stona bo‘lishi kerak.
3. Bu tilda uy ishlari uchun ma’lumotlarni tezda qabul qilib olish kerak.
4. U erkin holda ifodalanishi yoki qarama- qarshiliklarga uchrashi mumkin.
5. Kompyuterda ishlash uchun aniq vaqt belgilangan bo‘lishi kerak.

Bu komyuter tilining oxirgi maqsadi tezlik bilan rivijlanish bo‘ldi. U butun dunyo bo‘ylab tarqaldi. Hozirgi kunga kelib ular biroz qimmat hisoblanadi.

BASIC tili o‘z oldiga yana 2-3- va 4- rivojlanish bosqichlarini qo‘ydi. 1960-yilning boshlarida esa faqatgina bir yo‘nalishda ishlaydigan kompyuter terminlari bo‘lgan.

BASIC tili uchun 1- versiyani ishlab chiqqan, bu kompyuterdagi GE 225 uchun o‘chirish amalini bajarishda ishlatilgan. 1963- yilning kuzida BASIC tili uchun ishlash va dasturlar tizimi ishlab chiqilgan. 1964- yil 1- may soat 4 da chop etilgan va BASIC tilida 1- dastur tizimidan foydalanishga vaqt belgilandi. Iyun oyiga kelib esa terminallar tizimi 11 gacha kuzga kelib esa 20 gacha o‘sib bordi.

BASIC tilining versiyasi uncha katta, ommamavzu bo‘lmagan va bu termin unchalik ishlatilmagan. Dastlabki BASIC tili faqatgina 14 ta operator turiga ega bo‘lgan. Ularning hammasi muxim hisoblangan, lekin ularning ayrimlari ishlatilgan. Umuman olganda bu til juda muhim hisoblanadi va o‘rganish uchun ancha osondir.

BASIC tilining kelib chiqishi ko‘p tomonlama FORTRAN va ALGOL 60 ga borib taqaladi. So‘ngra bu til bir qancha rivojlanish bosqichlarini boshidan o‘tqazdi. ANSI institute BASIC tili uchun kichik hajmdagi normalarni yaratdi yani Minimal BASIC ni, bunda faqatgina kichik hajmda qo‘llanilgan. Faktlarga qaraganda hozirgi BASIC tili Minimal BASIC ga juda o‘xshaydi.

Umuman olganda Digital Equipment Corporation tomonidan BASIC tili uchun bir qancha versiyalar yaratgan, bu til BASIC-PLUS deb nomlangan. 1970-yilda kichik kompyuter PDP-11 uchun RSTS operatsion tizimisi ishlab chiqilgan.

BASIC tilining dasturi haqida bir qancha tanqidlar bo‘lgan. BASIC tilining versiyalarida bir qancha kamchiliklar bo‘lgan. Oxirgi versiyasida esa o‘zgartirishlar kiritilgan.

Har holda BASIC tili mashxur bo‘ldi uni o‘rganish osonligi va kichik kompyuterlarga ham o‘rnatish mumkinlik jihatidan.

BASIC tilining zamonaviy versiyasi 2 ta: Visual Basic .NET va Visual BASIC. Bularning ikkalasi ham katta kompyuterlarda ishlaydi. Visual BASIC tilining asosi Visual Basic .NET tilidir, lekin u dasturlash tizimida ishlatiladi

1.2. .Net platforma

Microsoft .NET platformasi dasturlar uchun yagona ijro muhitidir ularning rivojlanishini qo'llab-quvvatlaydi. .NET platformasini qurishda asosiy vazifalar quyidagilardir:

1. Tarqatilayotgan korporativ dasturlarning rivojlanishini qo'llab-quvvatlash, web-xizmatlar va XML asosida server va mobil komponentlarini o'z ichiga oladi.

2. Barcha ilovalar uchun yagona funktsiyalar kutubxonalari, ishlatiladigan dasturlash tillariga bog`liq emas.

.NET platformasi bitta ob`ektga – mo`ljallangan modelga asoslangan; barcha xizmatlar uchun yagona sinfning ierarxiyalari shakllangan bo`lib, dasturchiga platforma taqdim etilgan. .NET platformasi modeli katta darajada rivojlanishni osonlashtiradi. ilovalar uchun Windows platformalari, dasturiy ta'minoti bu erda deyarli barcha funksiyalar ishlab chiqaruvchiga taqdim etildi. WindowsAPI – da funktsiyalar jamlanmasi esa Windows platformaga strukturalashmagan.

Ob’ektga mo`ljallangan qo`shimcha funksiyalarni yaratish muammosi Windows API turli dasturlash tillarida mustaqil ravishda hal qilindi. Dasturlash tarixida birinchi marta .NET platformasi paydo bo'lishi bilan dasturlarni yaratish

uchun turli tillarni bir xil ishlatish imkonini beruvchi yagona model. .NET bazasi klasslari barcha dasturiy tizimlar uchun umumiy bo`lganligi sababli, bu til va asosiy funktsiyalar o`rtasidagi munosabatlar o`zgardi degan ma`noni anglatadi. Ilgari, har bir tilda Windows API vazifalari uchun, masalan, VC ++ da MFC, hozirda dasturlash tili platformaga moslashtirilgan o`z sinfining o`z kutubxonasini o`z ichiga olgan. Sinf kutubxonasi MFC statik ob`ekt modullari to`plamidir. Ular dasturning bajariladigan modulini yaratish bosqichida dasturga ulanadi va shu tufayli uning ajralmas qismi bo`ladi. Shu bilan birga, .NET Class Library - operatsion muhitning bir qismi bo`lgan (bajariladigan modulning maxsus turi) dinamik sinfi kutubxonasidir va dastur tomonidan faqat bajarilish vaqtida qo`llaniladi. .NET ishlab chiquvchilari statik ob`ekt kutubxonalari (LIB) va sinf kutubxonalari (DLL) o`rtasida tanlanadilar (C/C++ da yozadiganlar bundan mustasno).

3. Resurslardan va xavfsizligidan samarali foydalanish nuqtai nazaridan dasturni boshqarishni yaxshilash.

.NET da resurslarni boshqarish.

.NET platformasi avtomatik resurslarni boshqarish imkonini beradi. Barcha tillar uchun umumiy ish vaqti muhiti dastur ma`lumotlarini boshqaradi, ya`ni ularning tuzilishi va ilovadagi ob`ekt murojaatlarni, ularni yo`q qilish kerak bo`lganda murojaat qilishni to`xtatadi. Avtomatik xotira boshqaruvi xotirada qolgan keraksiz obyektlarni jamlash (garbage collection - sborka musora) yordamida.NET-da amalga oshiriladi. Bu xotira yo`qotilihi, resursni qayta ozod etish va hokazo kabi ko`plab umumiy muammolarni hal qiladi.

.NET da xavfsizlik

.NET uchun yaratilgan kod xavfsizlik uchun tekshirilishi mumkin. Ushbu dastur, foydalanuvchi zarar etkazishi mumkin emas yoki operatsion tizimining ishlashini buzishi mumkin emas. Xavfsizlik - .NET ning asosiy toshi. Dasturlarni yaratish va bajarishning barcha bosqichlarida kodlarga kirish huquqini, resurslar ruxsatiga tekshirishdan boshlab turli tekshiruvlar o`tkaziladi. Xavfsizlik tekshiruvlarining ayrim turlari:

Tiplarning xavfsizligi. Ma`lumotlarni muhofaza qilishni kafolatlaydigan dasturlar

faqat ular uchun ajratilgan xotira joylariga. Ob`ektlarga kirish faqat xavfsizlik tekshiruvlari o`rnatilgan maxsus interfeyslar orqali amalga oshiriladi.

Kodning haqiqiyligi. Sinfda yuklangan barcha manba matnlari haqidagi ma`lu-motlarni saqlaydi. Shunday qilib, ba`zi kodning atributlari (kodni joylashtirilgan joydan, muallif kim va h.k.). Ushbu ma`lumot ishlatish huquqini berish uchun foydalanish mumkin.

Resurslarga kirish uchun ruxsat. Resurslar odatda tizim bilan bog`liq. Resurslar fayllar, tarmoq ulanishlari, boshqarilmaydigan API (unmanaged APIs) deb nom berish huquqi bo`lishi mumkin. E`tibor bering, kirish huquqi faqat chaqiruvchi assotsiatsiyasi uchun emas, balki ayni paytda to`plamdagi boshqa barcha stekdan chaqirilayotganlar uchun ham tekshiriladi. Bu esa, ruxsatsiz komponentning resursga kirish huquqini qo`lga kiritadigan klassik hujumga yo`l qo`yishga imkon beradi.

Deklarativ xavfsizlik. Ushbu mexanizm, sinflarni, maydonlarni yoki metodlarni tushuntirish orqali to`g`ridan – to`g`ri kodni xavfsizlik tekshiruvlarini joylashtirish qobiliyatini ta`minlaydi. Tekshirish ochilish paytida yoki doimiy ravishda amalga oshirilishi mumkin (masalan, har bir metod boshlanganda).

Imperativxavfsizlik. Boshlang`ich jarayonda ushbu operatsiyani amalga oshirish huquqini tekshiradigan ishlab chiqilgan metod ichida odatiy kod. Bunday tekshiruvlar fayllar, foydalanuvchi interfeyslari va h.k larga, kirish uchun muhimdir.

.NET da xavfsizlik.

- Boshqa xavfsizlik modellari:
  - Huquq politikasi modeli
  - Rollar model
- Masofadan turib ishlashda xavfsizlikning ahamiyatini oshirish
- Foydalanuvchilarning ilovalariga joylashtirish uchun tayyor kriptografik muhofaza usullari.

--40-41--

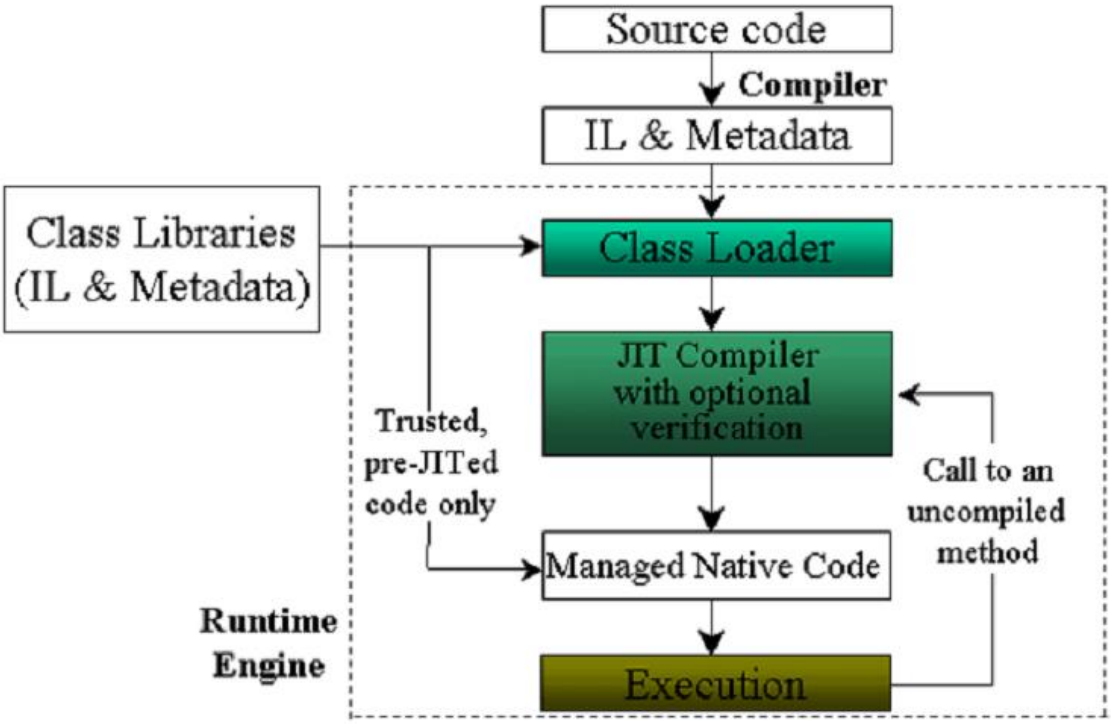
Umumiy ijro etuvchi muhit (Common Language Runtime)

Sinf kutubxonasi - bu strukturaning statik tarkibiy qismi. Dinamik komponent butun CLR ish vaqtini belgilaydi. Ushbu muhitning o`rni juda katta - uning funktsiyalari xotira, oqimlar, xavfsizlikni boshqarish, kodni tekshirish CLS ga muvofiqlikka, qidiruv byte kodidan mashina kodiga tuzish va yana ko`p narsalarni o`z ichiga oladi. CLR-ning muhim elementi - kucha tipli xotirani boshqaradigan kuchli musor sborkasi. CLR asosida dasturlarning ishonchliligi va xavfsizligini oshirish, shuningdek, platforma mustaqilligini ta`minlashning asosiy vazifalari hal etilmoqda.

.NET ilovalarining barcha bajariladigan modullari (ularni CLR modullari deb ataymiz) bajariladigan kod va meta ma`lumotlardan iborat. Meta-ma`lumotlar (misol uchun, turli deklaratsiyalar, metodlar, maydonlarlar va hodisalar), COM - texnologiyasida keng qo`llaniladi, bu oddiy ikkilik DLL lardan farq qiladi. CLR metadata o`z ichiga oladi sezilarli darajada kengaytirildi, bu versiyalarni yanada samarali nazorat qilish, dastur manbalarining ishonchliligini tekshirish va h.k.

CLR modullari mashina kodi (mahalliy kompyuter, bu kompyuter uchun "native") shaklida emas, balki oraliq til MicroSoft Intermediate Language (MSIL) spetsifikatsiyasiga muvofiq bayt kodi deb ataladi.

Boshqacha qilib aytganda, har bir .NET-mos keluvchi derleyici yuqori sathli til manba kodini ikkilik MSIL kodiga aylantirishi kerak, bu esa keyinchalik CLR da bajariladi.



1.2.1 – rasm. CLR modullari

Java`dan farqli o`laroq CLR klassik tarjimon rejimida kodni bajarmaydi, balki dasturning alohida qismlarini yoki butun dasturni mashina kodiga oldindan kompilyatsiya qilish orqali amalga oshiriladi.

Birinchi variant - MSIL ni mos keluvchi protseduralar (masalan, qo`llanilmagan dastur qismlari to`liq yig`ilmaydi) sifatida kompyuter kodiga aylantirishni amalga oshiradigan "Just-In-Time" nomli kompilyator yordamida asosan bajariladi.

MSIL virtual mashinaning assemblersi sifatida ko`rish mumkin. Bu assembler tipli emas, chunki u yuqori sathli tillarga xos bo`lgan ko`plab konstruktsiyaga ega: masalan, nomlangan soha, sinflar, metodlarni chaqirish, maydonlar, hodisalar va istisnolardan foydalanish uchun tavsiflarga ega. Bundan tashqari, MSIL statik tipli tekshiruvga ega bo`lgan yig`im turidir. Bu sizga ba`zi odatdagi xatolarni kuzatib borish imkonini beradi.

MSIL – qo`shimcha obstrakt darajadan o`z ichiga oladi. Kodni bir platformadan boshqasiga o`tkazishni, platforma razryadini o`zgartirishni yengillashtiradi. Java bytecode dan farqli o`laroq, MSIL 32 bit yoki boshqa bitli razryadga bog`liq emas. Bugungi kunda mobil 16-bit qurilmalar uchun MSIL

versiyalari (.NET Compact Framework), standart 32-bit versiyasi va borgan sari keng tarqalgan 64-bit qurilmalar bilan ishlaydigan maxsus versiya mavjud. Ushbu oraliq vositasi tufayli .NET ko`rinishi har qanday platformaga bog`liq emas, .NET arxitekturasida yaratilgan ilovalar ko`p platformali. .NET ning oraliq vakili har qanday platformaga bog`liq emasligi sababli, .NET arxitekturasida yaratilgan ilovalar ko`p platformali hisoblanadi.

MSIL, manba dasturida ishlatiladigan nomlar haqida juda ko`p ma`lumot saqlaganligiga e`tibor bering: sinflar, metodlar va istisnolar nomlari saqlanadi va teskari yig`ilish paytida olinishi mumkin. Biroq, MSILdan olingan manba matnlarni dizasembler qilish vaqtida lokal o'zgaruvchilar, konstantalar va parametrlarning nomlari faqat disk otladka versiyasida saqlanadi.

Standart turdagi tizim (Common Type System)

.NET platformasining boshqa qismi, umumiy toifa tizimi (CTS, standart turdagi tizim). .NET ning asosiy maqsadlaridan biri ular yaratilgan tillardan qat`i nazar, turli komponentlarning yuqori darajadagi muvofiqligini ta`minlashdir. Bu ikki yo`nalishda amalga oshiriladi: umumiy ijro muhitining barcha tarkibiy qismlarini taqdim etish va umumiy turdagi tizim yaratish. Standart turdagi tizimlar ba`zi ma`lumotlar turlarining boshqalar bilan qanday aloqa o`rnatishi va qanday qilib ishlashini to`liq aniqlaydi. Ular .NET metadata formatida bo`ladi. Dasturlash tilidagi turdagi tizim statik dasturni tekshirishni ta`minlash uchun mo`ljallangan. Bu til tuzilmalari taqiqlangan xatoliklarga yo`l qo`ymaslik shartlarini belgilaydigan qoidalar majmui.

.NET platformasi turli xil dasturlash tillarini qo`llab-quvvatlash uchun ishlab chiqilganligi sababli, uning umumiy turi tizimi (CTS) mavjud bo`lgan asosiy umumiy tillar turlarini birlashtiradi. Buning natijasida barcha .NET tillari (ob`ektga yo`naltirilgan, protsessual, funktsional) umumiy turdagi tizimga ega bo`lib, bu turli tillarda yozilgan dasturiy komponentlarning o`zaro ta`sirini ta`minlaydi. .NET-da umumiy turdagi tizim mavjudligi statik dasturni faqat kompilyator darajasida emas, balki ijro etuvchi tizim darajasida tekshirish imkonini beradi. Boshqacha qilib aytadigan bo`lsak, tizim ikkilamchi olib

boriladigan fayllarni ishga tushirishdan oldin tasdiqlashi mumkin. Bu kodning xavfsizligini ta'minlaydi.

.NET muhitida amalga oshiriladi va shuning uchun avtomatik xotira boshqarish (musora sborka) imkoniyatini beradi.

Strukturaviy va murojaat turlari

.NET-dagi barcha turlar ikkita asosiy turga bo`linadi: tizimli turlari (value-based yoki qiymatga asoslangan) va murojaat turlari (reference-based). Qiymatlar turlarini har doim o`z qiymatlarini nusxalash bilan bog`lashadi va murojaat turlari bilan ishlash har doim ularning qiymatlari manzillari orqali amalga oshiriladi.

Strukturaviy turlar qiymatni o`z ichiga oladi, unga murojaatni emas (ya`ni ishlashni yaxshilaydi). Tuzilishi turlari barcha raqamli ma`lumotlar turlarini (int, float, va hokazo), shuningdek birlashmalar va tuzilmalarni o`z ichiga oladi. Tuzilmalar bu primitiv turlari emas. Tip-qiymat boshqa turning merosidan olib bo`lmaydi. Struktura turlari uchun xotira stekdan ajratiladi. Bitta tuzilmaviy turni boshqasiga tayinlashda, turning o`ziga xos emas, balki uning bittadan nusxasi.

Murojaat turlari obyektlarning manzillari deb nomlanadigan ob`ektli murojaatlar (object references) ni ta`riflaydi. Ular uchun xotira doimo xotira nomi bilan ajratilgan maxsus maydonda ajratiladi, ular null sifatida belgilanadi, faqat manzil belgilangan bo`lsa ko`chiriladi, qiymatning o`zi emas. Murojaat tiplari sborka musora ob`ektlari bo`lib, (ularning Finalize metodi mavjud).

Sinflar

O`z-o`zini anglaydigan turlarning asoslari - bu sinflar. Sinflar boshqa turdagi qiymatlarni birlashtirishi hamda bir-biridan meros olishlari mumkin (faqat. NETda faqat bitta meros qo`llab quvvatlanadi). Sinflar quyidagi elementlardan iborat bo`lishi mumkin:

- Maydonlar (fields). Maydonlar boshqa yipli qiymatlarni saqlaydigan yacheykalardir.
- Metodlar (methods). sinf funktsiyalari bu metodlardir. Ular statik (static method) va ob`yektli (instance method) bo`ladi. Obyektli metod chaqirilganda har doim

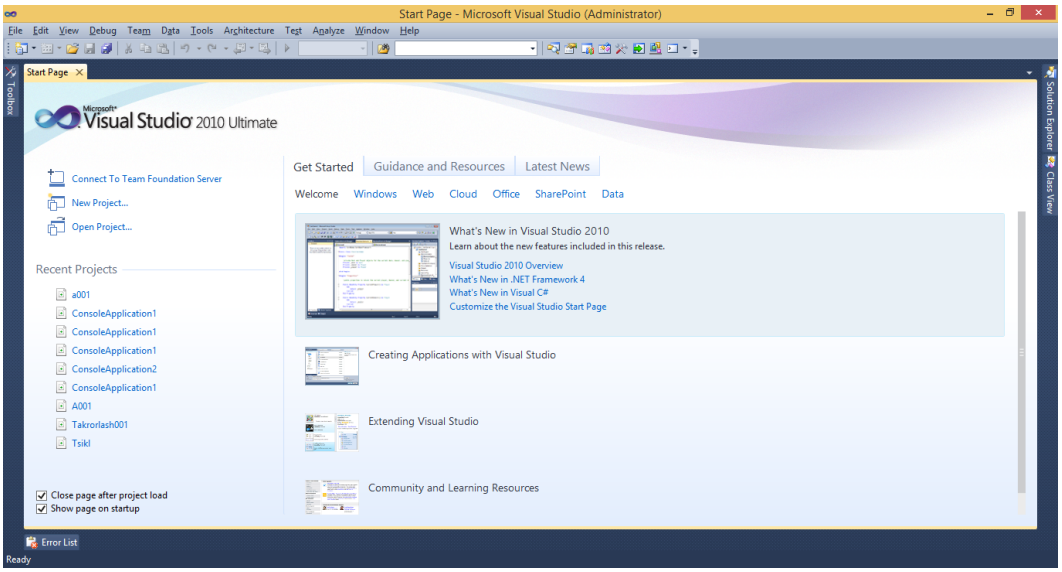
chaqirilayotgan ob`yektga havola oladi. Ob`yekt metodlari virtual va virtual bo`lmagan-larga bo`linadi.

- Xususiyatlar (properties). Xususiyat - bu juft metod bo'lib, ulardan biri qiymatni qaytaradi, ikkinchisi esa qiymati qabul qiladi.

1.3. Visual Studio muhiti va unda ishlash

Net platformasining asoslari bilan tanishib bo’lgandan so’ng to’g’ridan to’g’ri amaliyotga o’tamiz. Buning uchun biz dasturlash muhiti bilan tanishib chiqamiz.

Ishni Visal Studio .Net ni ishga tushirshdan boshlaymiz. Muhitni o`rganish davrida men 2010 –versiadan foydalandim. Ko’z oldimizda Asosiy oyna ochiladi, asosiy sahifasida bir nechta panellardan iborat



1.3.1 – rasm. Visal Studio .Net ning asosiy oynasi

Oynaning markazida ish sohasi joylashgan bo`lib, unda fayllar vkladkalar (yorliqlar) sifatida bo`ladi. U orqali mavjud fayllarni ochishimiz mumkin. Hozirgi vaqtda, ish sohasida HTML sahifa ochilgan bo`lib, yangi yoki mavjud bo’lgan loyihani ochishimiz mumkin.

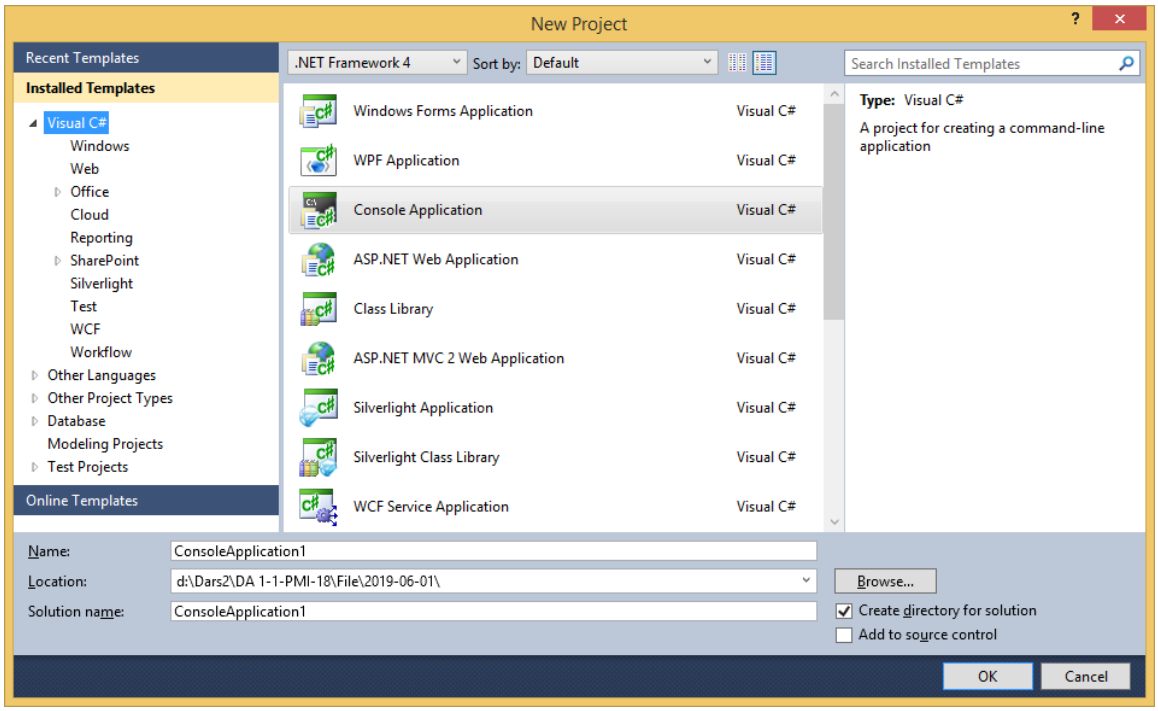
Asosiy sahifasining **Recent Projects** bo`limida – oldingi ishlangan loyihalardan 10 tasi ekranda ko’rsatilgan bo’ladi, ularni ustiga sichqonchaning chap tugmasini ikki marotaba bosish orqali ishga tushirish mumkin. Shu bo’limning yuqori qismida **Open Projects…** va **CreateProjects…** bandlari

mavjud. **Open Projects…** bandi orqali xotirada mavjud bo'lgan loyihani qayta ochish imkonini beradi, **CreateProjects…** bandi orqali esa yangi loyiha yaratish mumkin.

Loyiha bilan ishlash

Visual Studioda har qanday programma loyihadir. Loyiha – bu fayllar uchun papka (katalog) day. U zaruriy xusisiyatlarga ega (masalan platforma va til uchun yaratilgan prayekt) va dastur matni fayllarini o'z ichiga olib, bajariluvchi faylni kompilyatsiya qilib berishi mumkin. Solution (Loyiha) da bir nechta birlashtirish mumkin. Loyiha haqida yanada ko'proq ma`lumotlarni olish uchun, kompakt - diskning Documents/Visual Studio 2010.doc faylidan o`qishingiz mumkin.

Yangi loyiha yaratish uchun menyudan **File/New/Project** ni tanlaymiz. 1.3.2 - rasmda ko'rsatilgandek, **New Project** oynasi ochiladi. Loyihalar daraxti chap qismiga joylashgan. Bu yerda biz loyihaning turini tanlashimiz mumkin. Loyihalarning barcha turlari papkalarda joylashgan.



1.3.2 – rasm. Yangi loyiha yaratish oynasi

Loyiha xususiyati.

C# da har bir loyiha bir qator xususiyatlarga ega. Visual Studio muhit buni visual o'zgartirish imkonini beradi. Solution Explorer daraxtidan ildiz elementni belgilang va View/Property Pages. Common Propeties/General bandi o'zida forma

haqida asosiy ma’lumotlarni saqlab turadi. Ushbu oynaning tasnifi quyida keltirilgan:

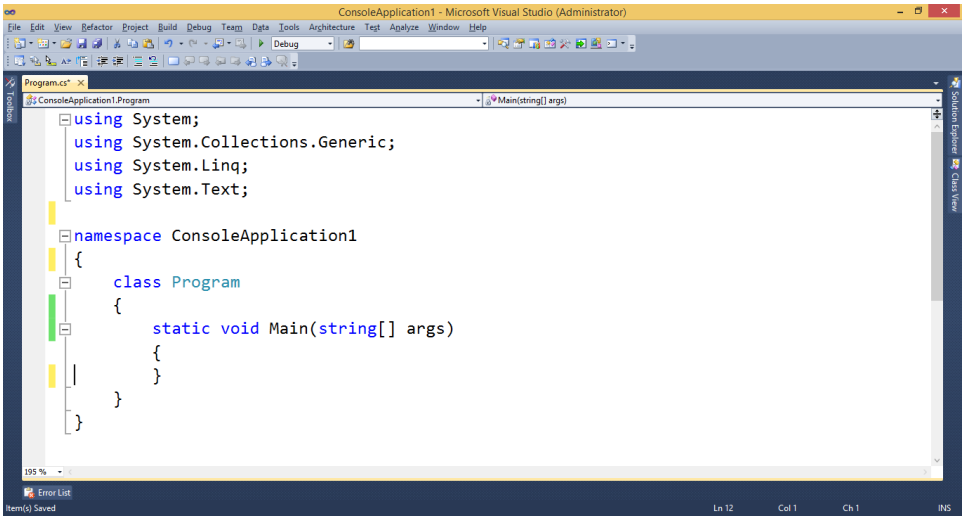
- ***Assembly Name*** – to’plamning nomi .
- ***Output type*** – dastur turi. Bu yerda Windows Application , Console Application yoki Class Library ni tanlash mumkin. Odatda Windows formasi uchun Windows tanlangan (o’rnatilgan) bo’ladi.
- ***Default Namespace*** – Joriy paytda odatiy nomlar yig’indisi .
- ***Startup Object*** – dastur ishga tushishini ta’minlovchi, Main metodiga ega sinf nomi.
- ***Application Icon*** – dastur belgisi (rasmi) ga ega bo’lgan faylning manzili.
- ***Project File*** – loyiha haqida ma’lumotni o’zida saqlab turuvchi fayl nomi .
- ***Project Folder*** - loyiha fayllarini saqlab turuvchi catalog (papka) ning manzili.
- ***Output File*** – chiquvchi fayl nomi, bunday fayl sizning dasturingiz yaratilayotgan paytda paydo bo’ladi.
- ***Wrapper Assembly Key File*** va ***Wrapper Assembly Key Name***. Bu xususiyat haqida keyinchalik aytib o’tamiz.
- ***Conditional Compilation Constants*** – dastur komplatsiyasi paytida aniqlangan o’zgarmaslar.
- ***Optimize code***- Bu xususiyatning yoqilishi(**true**) uning ishlash paytida dastur tezligi 10 marotabagacha oshadi.
- ***Check for Arithmetic Over flow/ Under flow*** – qaytarilgan qiymatning chegaradan chiqishi boshqarish.
- ***Allow unsafe code blocks*** – loyiha unsafe kalit so’zining ishlatilishiga ruxsat etish.
- ***Warning Level*** – kompliyatsiya paytida ogohlantirishlar darajasini aniqlash.
- ***Treat Warnings As Errors*** – barcha ogohlantirishlarni xato sifatida e’lon qilish.
- ***Output path*** - chiquvchi faylning qayerda hosil bo’lishi.
- ***XML Documentation File*** – dasturdagi barcha izohlar yoziladigan fayl nomi.

– ***Generate Debugging Information*** - dastur xatoligida oldingi qiymatni qaytarish.

Asosiy tushunchalar.

C# tili o'zining interfeysi va sintaksisi uning yuqori sathli dasturlash tillariga kirishiga olib keladi. Uning butun dasturlash tanasi bo'ylab boshqarishni EHM ga topshiradi va faqat bajarilishi lozim bo'lgan shartnigina dasturchi tomonidan yozilishi va shu bilan birga, u tomonidan qo'llaniladigan "aqlli" dasturlash (ya'ni har bir kodning bohlang'ich harfi yoki belgisi kiritilganida u kodning qolgan qismini o'zi namoyish etadi) ham bu dasturlash tilining keng ommaga ma'qul kelishini ta'minlab berdi. Visual Studio dagi **C#** tili ham xuddi shu vazifani bajarib beradi. Console Application qismida qora oynali muloqot oynasi yaratiladi va shu qora oyna dastur ishga tushganda paydo bo'ladi. Uning ko'rinishi avtomatik tarzda Visual Studio tomonidan shakllantirilgan bo'lib u dasturchi tomonidan o'zgartirilishi mumkin emas.

Consolda ishlatiladigan barcha komponentalar, ularning kodlari, parametrlar boshqa loyihalarda ham qo'llanilishi mumkin. Endi Consolda dasturlashni ko'rib chiqamiz. Birinchi navbatda **New->Project->Visual C#->Console Application** buyruqlar ketma ketligini bajaramiz.



1.3.3 - rasm. Consol ilova oynasi

C# tilining C,C++ dan farqi.

Andres Xijisberg (***C# tilining asoschisi***) bu tilni yaratayotganda o'z oldiga quyidagi vazifalarni asosiy maqsadi sifatida ishlata boshladi :

1. Birinchi bo'lib C/C++ tillari oilasida obyektga yo'naltirilgan dasturlash tilini yaratish;

2. Shunday obyektga yo'naltirilgan dasturlash tilini yaratish kerakki, unda hamma narsa obyekt sifatida yaratilsin (o'zgaruvchilar, formalar, massivlar, sinflar) ;

3. C++ tilini osonlashtirish, lekin shunday yo'l bilanki, C++ tilining kuchi va konstruksiyalari saqlanib qolsin.

Bu tilning eng katta yangiligi uning obyektga murojaati bo'lib, komponentlar yangi loyihalar yaratishdagi tuzilmalarni tuzishdagi barcha muammolarni hal etadi. Komponentlar tuzilishi faqatgina dasturlash tiliga bog'liq bo'lib qolmasdan, balki, uning qanday platformaga ega ekanligiga ham bog'liq [5].

Formalar ko'rinishi (dizayni).

Siz ekranda ko'radigan barcha ko'rinishlar formalar dizayni deb ataladi. Bu oynada siz yozadigan kodlar grafik ko'rinishda paydo bo'ladi.Dizayn foydalanuvchi uchun qulay bo'lgan interfeys yaratishda foydalaniladi.Form dizaynining eng asosiy elementlariga quyidagilarni kiritish mumkin :

- ***Properties Window;***
- ***Layout Toolbar;***
- ***Toolbox.***

Oldin aytib o'tilganidek , dizayner oynasida visual komponentlar faqatgina grafik ko'rinishda hosil bo'ladi.Sizning barcha ma'lumotlaringiz C# tilidagi kodlar yordamida saqlanadi.Biz tomonimizdan yaratilgan formaning kodini ko'rish uchun Solution Explorer oynasidagi MainFormning View Code bandini tanlashimiz mumkin unda kodlar ketma – ketligi oynasi paydo bo'ladi :

Dastur komplyatsiyasi

Shu paytgacha aytib o'tilgan biror dasturimiz ishchi dastur emas edi. Bu kodlar faqatgina kompilator ishga tushganida qilinishi kerak bo'lgan qoidalar edi xolos. Dasturni kompilatsiya qilish uchun **Build/Build Loyiha Nomi** bandini tanlash zarur (Visual Studioning keyingi versiyalarida Debug yordamida ishga

tushirilishi mumkin). Visual Studioning pastki qismida Task List yoki Error Listda xatoliklar ko’rsatilgan bo’lishi mumkin.

Xulosa.

Bitiruv malakaviy ishning birinchi bobi “Dasturlash tillari” bo`lib unda uchta bo`lim mavjud. Birinchi bo`limi, Dasturlash tillarining evolyutsiyasi bo`lib, unda hozirgi vaqtga qadar yaratilgan tillar haqida ma`lumot keltirilgan. Bobning ikkinchi bo`limida esa, .Net platforma. Bu bo`limda Visual Studio muhitida ishlashda platforma tushunchasi, bir tilnda yozilgan ikkinchi tilga o`tkazila olishi haqida ma`lumot keltirilgan. Platformadagi buyruqlar barcha tillarga ishlata olishi haqida ham ma`lumot keltirilgan. Bobning uchinchi bo`limi, Visual Studio muhiti va unda ishlash deb nomlangan. Bo`limda muhit bilan tanishish, muhitdagi imkoniyatlar haqida ma`lumotlar keltirilgan.

II BOB. C# TILIDA STRUKTURALI DASTURLASH

2.1. C# tili sintaksisi, standart tiplar

C# dasturlash tilining alfaviti quyidagilardan iborat.

Alfavit (yoki literallar yig'indisi) C# tilida ASCII kodlar jadvali bilan birgalikda quyidagi belgilarni o'z ichiga oladi :

- Lotin harflari
- 0 dan 9 gacha raqamlar
- "_" belgisi (harf sifatida ham ishlatiladi)
- maxsus belgilar to'plami : {}, 1 [] + - % / \ ; : ^ ? <> = ! & # ~ *
- boshqa belgilar

C# alfaviti so'zlarni tuzishda hizmat qiladi, ya'ni leksemalarni tuzishda. Leksemlarning 5 turi bor :

- Identifikator
- Kalit so'z
- Amallar belgilari
- Literallar
- Ajratuvchilar

Deyarli barcha leksemalar o'zining tuzilishiga ega. Ular ko'p alfavitlidir.

Kalit so'zlar va nomlar.

Quyidagi ro'yxatda C# tilining kalit so'zlari va nomlari berilgan bo'lib, dastur tuzilishi paytida ularni boshqa maqsadda ishlatish (masalan o'zgaruvchi nomini inisializatsiya qilishda) mumkin emas.

abstract	*do*	*in*	*protected*	*true*
as	*double*	*int*	*public*	*try*
base	*else*	*interface*	*readonly*	*typeof*
bool	*enum*	*internal*	*ref*	*uint*
break	*event*	*is*	*return*	*ulong*
byte	*explicit*	*lock*	*sbyte*	*unchecked*
case	*extern*	*long*	*sealed*	*unsafe*

catch	*false*	*namespace*	*short*	*ushort*
char	*finally*	*new*	*sizeof*	*using*
checked	*fixed*	*null*	*stackalloc*	*virtual*
class	*float*	*object*	*static*	*void*
const	*for*	*operator*	*string*	*volatile*
continue	*foreach*	*out*	*struct*	*while*
decimal	*goto*	*override*	*switch*	
default	*if*	*params*	*this*	
delegate	*implicit*	*private*	*throw*	

C# tilida boshqa tillarda bo'lgani kabi dasturning har bir qismiga izoh yozish mumkin. Bu izohlar dastur kompilatsiyasida ishtirok etmaydi va dastur ishiga hech qanday ta'sir ko'rsatmaydi. C# da izoh yozish uchun /* */, // belgilaridan foydalanish mumkin.

// belgisi shu belgidan keyin to shu satr oxirigacha bo'lgan barcha belgilarni izoh sifatida qabul qiladi. /* */ bu orqali istalgan qismni izohga olish mumkin.

Literallar.

C# tilida 5 xil literal mavjud ;

- Butun tipli literal
- Haqiqiy tipli literal
- Belgili literal
- Satr tipli literal
- Mantiqiy tipli literal

Literallar – bu tilning maxsus tushunchasidir. Har bir literallar to'plami uchun alohida yozilish qoidasi mavjud. Masalan:

- Butun tipli literallar: 5, 7, 8, -12, 234
- Haqiqiy tipli literallar: 3.6, -56.8, 0.9
- Belgili literallar: 'a', 'b', '?',
- Satr tipli literallar: "salom", "aka", "abcd"
- Mantiqiy tipli literallar: true, false

Consol bilan ishlash.

Consol bilan ishlash uchun .NET da Console sinfi ishlatiladi. Bu sinfning afzalligi 2 ta qismdan iborat bo'lib: uning barcha metodlari o'zgarmas, sanoqli bo'lib , uni ishlatish uchun nusxalash shart emas. U kiritish , chiqarish va xatoliklarni chiqarishni o'z ichiga olgan. Odatda kiritish/chiqarish standart consol da amalga oshiriladi (agar u bo'lmasa, masalan oynali masalalarda chiqarish amalga oshirilmaydi), lekin kiritish va chiqarish qurilmalarini o'zgartirish mumkin. Consol bilan ishlashda asosan 4 metod ishlatiladi: Read, ReadLine, Write, WriteLine, birinchi ikkitasi kiritish, qolgani chiqarish metodlari.

Read metodi.

Read metodi kiritish qurilmalaridan belgini qabul qiladi. U int tipida kiritilgan belgi kodini qaytaradi va hech narsa kiritilmagan bo'lsa , -1 (minus bir) ni qaytaradi.

Masalan

```
do
{ int i = Console.Read();
  if (i != -1)
  Console.WriteLine("{0} <=> {1}", (char) i, i) ;
  else break;
 } while (true);
```

Bu dastur kiritilgan belgi kodini ekranga chiqarib beradi.

Readline metodi.

Readline metodi kiritish qurilmalaridan matnning satrini qabul qiladi (uning qiymati keyingi satrga o'tish belgisi bilan tugaydi). U string tipidagi qiymat yoki null (agar kiritish amalgam oshmagan bo'lsa) qiymatini qaytaradi.

Masalan

```
do
{ string s = Console.ReadLine();
  if (s != null)
    Console.WriteLine("Kiritilgan satr: " + s) ;
```

```
else break;
} while (true);
```

Write va WriteLine metodlari. Write metodi unga yuborilgan o’zgaruvchi qiymatini ekranga chiqarish vazifasini bajaradi. U string tipini qabul qiladi. U barcha standart tiplar uchun ishlaydi. Shuning uchun uni parametr sifatida chaqirish mumkin.

```
Console.Write (I);
Console.Write (0.7);
Console.Write("Hello!"),
```

Undagi satrga o’zgaruvchi qiymatini qo’shib e‘lon qilish uchun quyidagi kodni yozish kifoya :

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

WriteLine metodining farqi shundaki, undan keyin, keyingi (yangi) satrdan boshlab o’ziga yuborilgan o’zgaruvchi qiymatini ekranga chiqarib beradi.

C# tilida standart tiplari.

C# tili juda tiplashgan til hisoblanadi. Uni ishlatish paytida har bir o’zgaruvchi obyektning tipini alohida e‘lon qilish kerak(masalan, butun son, satr, oyna, tugma va h.z). Xuddi C++ va Java tillari kabi C# tilida ham 2 xil ma’lumotlar tipi mavjud : birinchi aniqlangan va xotirada til tomonidan avtomatik joylashtirilgan, ikkinchi dasturchi – foydalanuvchi tomonidan kiritiladigan va aniqlanadigan. C# ning ustun tomoni unda ma’lumotlar yana ikki turga bo’linadi: o’lchamli va yo’nalishli. Ularning asosiy farqi ma’lumotlarni xotirada joylashtirishidir. O’lchamli tip o’zining aniq qiymatini stekka yozib qo’yadi, yo’nalishli tip esa bu stekka faqat qaysidir (o’zi aniqlaydigan) obyekt manzilini yozib qo’yadi, obyektning o’zi esa *kucha* da saqlanadi. *Kucha* – bu dastur saqlanadigan asosiy xotira bo’lib, unga murojaat qilish dastur tezligini biroz pasaytiradi. Lekin agar siz juda katta obyektlar bilan ishlayotgan bo’lsangiz, unda bu obyektni *kucha* da saqlashning bir muncha afzallik tomonlari bor.

Standart tiplar.

Quyidagi jadvalda C# tilida standart tiplar va ularning o’chami keltirilgan :

Tip	Qiymat oralig’i	O’lchami
sbyte	-128 to 127	Belgili 8-bit butun
byte	0 to 255	Belgisiz 8-bit butun
char	U + 0000 to U + 0000T	16-bitli Unicod
bool	true yoki false.	1 bayt
short	-32768 to 32767	Belgili 16-bit butun
ushort	0 to 65535	Belgisiz 16-bit butun
int	-2147483648 to 2147483647	Belgili 32-bit butun
uint	0 to 4294967295	Belgisiz 32-bit butun
long	-9223372036854775808 to 9223372036854775807	Belgili 32-bit butun
ulong	0 to 18446744073709551615	Belgisiz 32-bit butun
float	-1.5*10^6 to 3.4 *10^7	4 баут, aniqlik — 7 razryadli
double	-1.5*10^6 to 3.4 *10^7	8 баут, aniqlik —16 razryadli
decimal	-5.0*10^3 2 4 to 1.7 -lO3 0 8	12 баут, aniqlik — 28 razryadli

Standart tiplarni o’zlashtirish.

Bir tipga tegishli bo’lgan obyektlar boshqa tipli obyektga oshkor yoki yashirin tarzda o’zlashtirilishi mumkin.Yashirin tarzda avtomatik o’zlashtirish bo’lib, uni kompyuter sizning o’rningizda amalga oshiradi. Oshkor o’zlashtirish faqatgina siz tomoningizdan berilgan qoida bo’yicha amalga oshadi.Yashirin o’zlashtirish ma’lumotlar yo’qolishini oldini oladi.

Masalan, siz short tipidagi (2 bayt) axborotni int tipidagi (4 bayt) obyektga o’zlashtira olmaysiz, bunda axborot yo’qolishi bo’lishi mumkin. Lekin buni kompyuter avtomatik tarzda o’zlashtirganda hech qanday xatolik ro’y bermaydi.

```
short x=1;
int  y = x ; // yashirin o’zlashtirish
```

Agar siz aksincha almashtirishni amalga oshirsangiz, axborot yo’qolishiga olib keladi. Kompilyator bunday o’zlashtirishni amalga oshirmaydi.

```
short x ;
int y=5;
```

x=y; // Komplyatsiya amalga oshmaydi

Siz buning uchun oshkor almashtirishni amalga oshirishingiz kerak.

```
short x;
int y;
x=(short) y; // to'g'ri
```

O'zgaruvchilar.

O'zgaruvchi – xotiraning ma'lum bir qismini biror bir tipli axborot uchun ajratishdir. Yuqorida e'lon qilingan x va y lar o'zgaruvchilardir. O'zgaruvchilar inisializatsiya paytida (qiymat qabul qilish paytida) yoki dastur yordamida o'zgartirilishi mumkin.

O'zgaruvchilar qiymatini aniqlash.

O'zgaruvchini hosil qilish uchun siz o'zgaruvchining tipini va keyin esa uning nomini berishingiz kerak.Uning qiymatini e'lon qilish paytida yoki dastur davomida berishingiz mumkin.

Masalan: a va b sonlarni yig'indisini s ga o'zlashtirish dasturini ko'ramiz

```
using System;
namespace _02_misol
{
   class Program
   {
      static void Main(string[] args)
      {       int a,b,s; a=2;b=3;s=a+b;
              Console.Write("s="+s);
             Console.ReadKey();
    }    }    }
```

O'zgarmaslar.

O'zgarmas – bu shunday o'zgaruvchiki, uning qiymati hech qachon o'zgarmaydi.

O’zgaruvchilar – qiymatlarning o’zlashtirishning qulay usulidir. Lekin siz qiymatning dastur davomida o’zgarmasligini kafolatlashni xohlasangiz, buning uchun o’zgarmas – o’zgaruvchilardan foydalanishingiz mumkin.

Masalan, agar siz quyidagi amalni bajarmoqchi bo’lsangiz :

*y = x * 3.14159265358979323846264338327 95*

ushbu ko’paytmani,

pi=3.14159265358979323846264338327 95;

*y=x*pi;*

ko’rinishida yozishingiz afzalroq.

O’zgarmaslarning 3 ta : Literallar, belgili o’zgarmaslar va ro`yxatlar turi mavjud.

Literal : x=100 ;

100 – literal o’zgarmas.

*Belgili***.** const double pi=3.14159265358979323846264338327 95;

Pi – belgili o’zgarmas.

Masalan:

```
class Program
   {    static void Main(string[] args)
       {const  double p = 3.14159265358979323846264338327 95;
       System.Console.WriteLine(p);
       System.Console.ReadKey();
       }     }
```

Dastur natijasi : pi : 3.14159265358979323846264338327 95 ga teng.

Satr o’zgarmaslari.

Dastur yozish paytida satr o’zgarmasini e’lon qilish uchun uni ikkita qo’shtirnoq orasiga olish kerak. Masalan, “salom yoshlar”. Bu satr o’zgarmasi sifatida komplyatsiya bo’ladi. Buni siz dasturning istalgan qismida bajarishingiz mumkin. Masalan, funksiya parametrlarini o’zlashtirishda, o’zgaruvchilarni e’lon qilishda. *String a=”Salom yoshlar”.*

2.2. Ifoda, intruksiya va operatorlar

Ifodalar.

Ifoda – qiymatni aniqlovchi kod satridir. Oddiy ifodaga misol: MyValue=100;

MyValue ning o’zi bir qiymatni aniqlovchi operator bo’lsada, uni ham qiymat sifatida o’zlashtirish mumkin. Chunki u 100 qiymatini qabul qiladi. Misol uchun: MysecondValue=MyValue=100; Bu misolda 100 literali avval MyValue ga keyin “=“ o’zlashtirish operatori yordamida MySecondValue o’zgaruvchisiga o’zlashtiriladi. Bu bilan 100 qiymati har ikkala o’zgaruvchiga birdaniga o’zlashtiriladi. Bu yo’l bilan siz bir necha o’zgaruvchiga birta qiymatni o’zlashtirish imkoniyatiga ega bo’lasiz.

int a=b=c=d=g=h=l=20;

Instruksiya(Amal).

Instruksiya – bu dastur kodida tamomlangan ifodadir. C# tilidagi dastur instruksiyalar ketma – ketligidan iborat. Har bir instruksiya “;” belgisi bilan tugallanishi kerak.

Masalan,

int x,y; x=100; y=0;

Shu bilan birgalikda C# da tarkibli instruksiya ham mavjud. Bunday instruksiyalar bir necha instruksiyalardan iborat bo’ladi. Ular “{ }” figurali qavslar orasiga olinadi.

Masalan :

{ x=10; y=9; int a; }

Bu misolda 3 ta oddiy instruksiyalar birta tarkibli instruksiyada joylashadi.

Ajratuvchilar.

C# tilida probel, tabulyatsiya va keyingi satrga o’tish belgilari ajratuvchilar hisoblanadi. Instruksiyalardagi ortiqcha ajratuvchilar bekor qilinadi.

Masalan :

MyValue = 100 ; Yoki *MyValue = 100;*

Komplyator bu ikkita ifodani bir xilda komplyatsiya qiladi. Shuni aytib o’tish ham kerakki, satrda ajratuvchilar bekor qilinmaydigan payti ham bo’ladi. Agar siz Console.Writeline(“Salom yoshlar !!!!!!”), deb yozsangiz “Yoshlar ” va “!!!” orasidagi probellar (bo’sh joylar) bekor qilinamaydi balki, bo’sh joy sifatida qiymat qabul qiladi.

Har bir operator boshqa operator orasida kamida bitta bo’sh joy bo’lishi shart :

int x; // to’g’ri

intx; // noto’g’ri

C# da arifmetik amallar

Ko'p programmalar ijro davomida arifmetik amallarni bajaradi. C# dagi amallar quyidagi jadvalda berilgan. Ular ikkita operand bilan ishlatdi. C# dagi amal Arifmetik operator Algebraik ifoda C# dagi ifodasi

Qo'shish - a+b

Ayirish - a-b

Ko'paytirish - a*b

Bo'lish - a/b

Qoldiqli bo`lish – a%b

C# da qavslarning ma'nisi huddi algebradagidekdir. Undan tashqari boshqa boshqa algebraik ifodalarning ketma-ketligi ham odatdagidek. ko'paytirish, bo'lish va modul olish operatorlari ijro ko'radi. Agar bir necha operator ketma-ket kelsa, ular chapdan o'nga qarab ishlanadi. Bu operatorlardan keyin esa qo'shish va ayirish ijro etiladi.

Mantiqiy solishtirish operatorlari

C# bir necha solishtirish operatorlariga ega. Algebraik ifoda C# dagi operator C# dagi ifoda Algebraik ma'nosi tenglik guruhi

x==y x tengdir y ga

teng emas != x!=y x teng emas y ga

solishtirish guruhi

x>y x katta y dan

x<y x kichkina y dan

katta-teng >= x>=y x katta yoki teng y ga

kichik-teng <= x<=y x kichik yoki teng y ga

==, !=, >= va <= operatorlarni yozganda oraga bo'sh joy qo'yib ketish sintaksis xatodir.

1 ga oshirish va kamaytirish operatorlari (increment and decrement)

C# da bir argument oluvchi inkrenet (++) va dekrement (--) operatorlari mavjuddir. Bular ikki ko'rinishda ishlatilinadi, biri o'zgaruvchidan oldin (++f - preinkrement, --d - predekrement), boshqasi o'zgaruvchidan keyin (s++ - postinkrement, s-- - postdekrement) ishlatilgan holi. Bularning bir-biridan farqini aytin o'taylik. Postinkrementda o'zgaruvchining qiymati ushbu o'zgaruvchi qatnashgan ifodada ishlatilinadi va undan keyin qiymati birga oshiriladi. Preinkrementda esa o'zgaruvchining qiymati birga oshiriladi, va bu yangi qiymat ifodada qo'llaniladi. Predekrement va postdekrement ham aynan shunday ishlaydi lekin qiymat birga kamaytiriladi. Bu operatorlar faqatgina o'zgaruvchining qiymatini birga oshirish/kamaytirish uchun ham ishlatilinishi mumkin, yani boshqa ifoda ichida qo'llanilmasdan. Bu holda pre va post formalarining farqi yo'q..

Mantiqiy operatorlar

Bosqaruv strukturalarida shart qismi bor dedik. Shu paytgacha ishlatgan shartlarimiz ancha sodda edi. Agar bir necha shartni tekshirmoqchi bo'lganimizda ayri-ayri shart qismlarini yozardik. Lekin C# da bir necha sodda shartni birlashtirib, bitta murakkab shart ifodasini tuzishga yordam beradigan mantiqiy operatorlar mavjuddir. Bilar mantiqiy VA - && (AND), mantiqiy YOKI - || (OR) va mantiqiy INKOR - ! (NOT). Bular bilan misol keltiraylik. Faraz qilaylik, bir amalni bajarishdan oldin, ikkala shartimiz (ikkitadan ko'p ham bo'lishi mumkin) true (haqiqat) bo'lsin.

if (i < 10 && l >= 20){...}

Bu yerda {} qavslardagi ifodalar bloki faqat i 10 dan kichkina va l 20 dan katta yoki teng bo'lgandagina ijro ko'radi.

AND ning (&&) jadvali:

ifoda1 ifoda2 ifoda1 && ifoda2

false (0) false (0) false (0)

true (1) false (0) false (0)

false (0) true (1) false (0)

true (1) true (1) true (1)

Bu yerda true ni yeriga 1, false ni qiymati o'rniga 0 ni qo'llashimiz mumkin.

2.3. Boshqaruv operatorlari

O’tish operatorlari.

C# tilida o’tish operatorlari ikki xil bo’ladi : **shartli** va **shartsiz**.

Shartsiz o’tish operatorlari.

Shartsiz o’tish operatorlari ikki xil usulda qo’llanilishi mumkin.1 – funksiyani chaqirish yo’li bilan. Bunda dastur davomida komplyator funksiya nomlarini tekshirib boradi, agar shunday funksiya topilsa, dastur o’z ishini shu yerda to’xtatib funksiyaning ishga tushishini amalga oshiradi. Funksiya o’z amallarini bajarib bo’lganidan so’ng, komplyator dasturni bajarilishini funksiya nomidan so’ng turgan instrukiyaga o’tkazadi.

Masalan,

```
using System;

class Functions

{

static void Main( )

{ Console.WriteLine("T() -  metodini chaqiramiz...");

 T ( ) ;
```

Console.WriteLine ("Main metodiga qaytish."); }

static void T( ) { Console.WriteLine("T() metodi ishlayapti!"); }

Ushbu dasturni ishga tushiring, dastur natijasi quyidagicha bo'ladi :

Metod Main. T metodini chaqiramiz…

Main metodiga qaytish…

T metodi ishlayapti…

Shartsiz o'tishning ikkinchi usuli: **goto**, **break**, **return** va **throw** kalit so'zlari yordamida bajarish mumkin. Ushbu kalit so'zlar haqida quyida aytib o'tamiz.

Shartli o'tish operatorlari.

Shartli o'tish uchun **if**, **else** yoki **switch** kalit so'zlaridan foydalanish mumkin.Bunday o'tish faqat shart rost bo'lganidagina bajariladi.

If … else operatori.

If…else – bu shartli o'tish operatori bo'lib, shart **if** qismida bajariladi. Agar shart rost bo'lsa, shartdan so'ng yozilgan instruksiyalar to'plami (tarkibli instruksiya) bajariladi, agar yolg'on bo'lsa, **else** qismida yozilgan (yozilmagan bo'lishi ham mumkin) tarkibli instruksiya bajariladi.

Masalan:

if (a>b) System.Console.Writeln("kattasi="+a); else
System.Console.Writeln("kattasi="+b);

Shart operatorining natijasi bool tipiga tegishli bo'lib, **true**(rost) yoki **false**(yolg'on) bo'lishi mumkin. C# da quyidagi munosabat amallari mavjud :

= = - tenglik,

> - katta,

< - kichik,

>= - katta yoki teng,

<= - kichik yoki teng

!= - teng emas.

if (shart) operator1; else operator2;

Agar shart rost bo'lsa, *operator1* bajariladi, agar yolg'on bo'lsa, *operator2* bajariladi. Shuni alohida takidlab o'tish lozimki, agarda siz shart yolg'on bo'lganda dasturingiz hech bir ish bajarmasligini xohlasangiz, *operator2* ni yozmasligingiz mumkin.Bunda dastur **if** … **else** dan so'ng yozilgan kod bo'yicha o'z ishini davom ettiradi. Agarda *operator1* yoki *operator2* dan so'ng bajarilishi lozim bo'lgan amallar soni 1 tadan ortiq bo'lsa ular figurali {} qavslar orasida yozilishi lozim.

Masalan: a va b sonlarni kattasini aga kichigini bga yozish dasturi

```
class Program    {
    static void Main(string[] args)       {
        int a, b, c;
        a = 10; b=20;
        if (a <b) { c = a; a = b; b = c; }
    System.Console.WriteLine(a+" ,  "+b);
    System.Console.ReadKey();       }
  }
```

(Ichma ich) shart operatorlari.

Ichma ich shart operatorlari - bu C# dasturlash tilining afzalligi bo'lib, bunda bir necha murakkab shartlarni bir shart orqali tekshirish, aniqlash mumkin. Bir o'zgaruvchi qiymatini bir necha shartlar orqali tekshirish uchun ichma ich bir necha shart operatorlaridan foydalanish mumkin :

```
using System;

class Values {

static void Main( ) {

int temp = 25;

if (temp > 21) {

if (temp < 26) {

Console.WrireLine (

"Temperatura meyorda");

if (temp == 24) {

Console.WriceLine("ishlash sharoiti optimal");

}

else {

Console .WriteLine ("ishlash sharoiti optimal emas\n" +

"optimal temperatura 24");

} } }
```

Ko’p shartlilik qo’llanilishi.

Bunda bir necha shartni bir vaqtda tekshirish zarurati hisobga olinadi. C# tilida buning uchun maxsus qo’shish (shartni) kalit so’zlari mavjud : && - va, | | - yoki, ! – inkor (!= bo’lsa, teng emas ma’nosida) .

Masalan,

```
using System;

namespace Misol {

class Programm {
```

static void Main(string[] args) {

int nl = 5; int n2 = 0;

if ((nl = = 5) && (n2 = = 5)) Console.WriteLine(" Salom");

else Console.WriteLine("Yoshlar");

if((nl = = 5) | | (n2 = = 5)) Console.WriteLine("buxoro");

else Console.WriteLine("Vaqt");

} }

Bu misolda har bir **if** operatori ikkita shartni tekshirib boradi.

Switch operatori.

Juda ko'p hollarda ichma ich yozilgan shart operatorlari ko'p tekshirish olib borib bir nechta amal bajaradi. Lekin bulardan faqat bittasigina haqiqiy bo'ladi.

Masalan,

if (myValue == 10) Console.WriteLine("myValue teng 10");

else

if (myValue == 20) Console.WriteLine("myValue teng 20 ") ;

else

if (myValue == 30} Console.WriteLine("myValue teng 30 ") ;

else ....

Bunday murakkab shart tekshirilishi bo'lganda **if** operatoridan ko'ra, uning yangi versiyasi bo'lgan **switch** dan foydalanish afzal. Switch operatori quyidagicha ishlaydi :

Switch (ifoda) {

case : o'zgarmas ifoda : instruksiya

o’tish ifodasi

[default : instruksiya]

}

Misoldan ko’rinib turibdiki, switch da ham tekshirilayotgan ifoda if … else dagi kabi, ( ) orasiga olingan va operatordan keyin yozilgan.**Case**(tekshirish) va default (aks holda) bo’limlari

qaysi amal bajarilishi zarurligini aniqlab beradi.**Case** operatori albatta biror bir tenglashtirish uchun qiymat talab qiladi.

{

switch (myVslue) {

case 10: Console.WriteLine("myValue равno 10") ; break;

case 20: Console.WriteLine("myValue равno 20"); break;

case 30: Console.WriteLine("myValue равno 30"); break;

} }

Switch operatorida **default** amalini yozish shart emas, chunki u berilgan qiymatning tanlangan birorta qiymatga mos kelmaganda bajariladigan amallarni o’z ichiga oladi. Agarda berilgan qiymat birorta tanlangan qiymatga mos kelsa, u holda **case** amalidan keyin bajariladigan amallar (ular bir nechta bo’lsa, **{ }** orasiga olinadi) bajariladi, so’ng **break** amali switch operatorining ishini shu joyda to’xtatadi va switch operatoridan keyin keladigan operator ishini davom ettiradi.

Har bir **case** operatori o’zida **break** amalini ushlab turishi lozim.

Masalan,

{

switch (a) {

case 10: Console.WriteLine("a= 10") ;break;

```
case 20: Console.WriteLine("a=20"); break;

case 30: Console.WriteLine("a= 30"); break;

}
```

yoki quyidagicha berish ham mumkin :

```
using System;

namespace Misol {

class MyClass {

static void Main(string[j args)

{ int user = 0;

user = Convert.Tolnt32(Console.ReadLine( ));

switch(user) {

case 0: Console.WriteLine("Salom User 1");break;

case 1 : Console.WriteLine("Salom User2"); break;

case 2: Console.WriteLine("Salom User3"); break;

default: Console.WriteLine("Salom yangi fordalanuvchi"); break; }

} } }
```

Quyida iqtisodiy masalani yechish usuli berilgan :

```
using System;

namespace C_Sharp_Programming {

class Part {

public static void Main() {

Console.WriteLine("1: mahsulot nomini kiriting\ n2: mahsulot sonini kiriting");

int Choice = Convert.Tolnt32(Console.ReadLine());
```

```
switch (Choice) {
case 1 : string Kane; Console.Write("Mahsulot nomini kiriting " ) ;
        Name = Console.ReadLine(); break;
case 2: int Count; Console.Write("Mahsulot sonini kiriting " ) ;
       Name = Console.ReadLine();
      Count = Convert.Tolnt32(Console.ReadLine()) ; break;
default:  break;
}
```

Switch va satrlar bilan ishlash.

Yuqorida keltirilgan misollarda **user**lar butun tipga tegishli edi.Agarda siz switch operatorini satrli tipda ishlatmoqchi bo'lsangiz, u holda quyidagicha yozishingiz mumkin :

```
Case : "Anvar" ;
```

Agarda tekshirish uchun satrlar ko'p bo'lsa, butun tipli o'zgaruvchilar ko'p marotaba **case** operatorini ishlatishga majbur etadi. Quyida esa satr o'zgaruvchisi ishlatilgan switch operatori berilgan :

```
using System;
namespace SwitchStatement {
class MyClass {
static void Main(string[] args) {
string user;
user = Console.ReadLine() ;
switch(user) {
case "userl": Console.WriteLine("Salom 1 chi foydalanuvchi");break;
```

case "user2":Console.WriteLine ("Salom 2 chi foydalanuvchi ");break;

case "user3":Console.WriteLine ("Salom 3 chi foydalanuvchi ");break;

default:Console.WriteLine("Salom 4 chi foydalanuvchi "); break; }

}}}

Bu yerda siz foydalanuvchi bo'lib kirish uchun, butun tip emas balki, satr tipida kiritishingiz mumkin bo'ladi.Agar siz user1 deb yozsangiz ekranda "salom birinchi foydalanuvchi" degan yozuv paydo bo'ladi.

Takrorlash operatorlari.

Goto takrorlash operatori. Goto operatori boshqa barcha takrorlash operatorlari uchun asosiy mezon bo'lib xizmat qiladi. Lekin shu bilan birgalikda unda juda ko'p o'tishlar amalga oshiriladi va buning natijasida dastur chalkashliklarga yo'l qo'yadi. Professional dasturchilar odatda unda foydalanihmaydi, lekin C# tilini mukammal o'rganish uchun bu operator haqida qisqacha aytib o'tamiz :

1. Label (metka, belgi) yaratiladi.

2. Shu Labelga o'tish bajariladi.

Masalan:

using System;

namespace LabelStatement {

public class Labels {

public static int Main( ) {

int i = 0;

label:

Console.WriteLine ("i: {0 } ", i);

```
i + + ;

if (i < 10)  goto label;

return 0;

}}}
```

While takrorlash operatori.

Bu takrorlash operatori *“shart qanoatlantiradi – ish davom etadi”* qoidasi bo’yicha ishlaydi. Bunda bool tipiga tegishli qiymat qaytariladi.

```
While ( shart)

{   instruksiya (amallar)  }
```

Agar shart tekshirilganda rost bo’lsa, instruksiyalar bloki bajariladi, aks holda while dastur ishlashini to’xtatadi.

Masalan:

```
using System;

namespace WhileStatement {

public class Labels {

public static int Main( ) {

int i = 0;

while(i < 10) { i++;

Console.WriteLine("i: {0}",i ); }

return 0;

}}}
```

do … while takrorlash operatori.

Shunday hollar bo’ladiki, while takrorlash operatori sizning talablaringizga javob bermaydi, bunday hollarda do… while takrorlash operatoridan foydalanish qulayroq. Masalan: siz shartni boshida emas balki, oxirida tekshirishni hohlaysiz :

```
public class Labels {

public static int Main() {

int i = 0;

do { Console . WriteLine ("i : {0} ", i) ; } while(i < 10) ;

return 0;

}}
```

Bu misoldan ko’rinadiki *i* 10 dan kichik bo’ladi va hech bo’lmaganda birta amal bajaradi. Do … While operatori “*amal bajar, agar shart bajarilsa, yana bir bor bajar*” qoidasi bo’yicha ishlaydi. While operatori bu holda birorta ham amal bajarmas edi.

For takrorlash operatori.

Agar yana bir bor yuqoridagi barcha takrorlash operatorlari (while, do…while, goto) ga e’tibor bersak, shuni aniqlash mumkinki, ularda doimo oldin *i* o’zgaruvchisi inisializatsiya (nomlash) qilinadi, keyin u 1 taga ortiriladi va takrorlanish sharti (i<10) tekshiriladi. For takrorlash operatori bu amallarni birta instruksiyaga birlashtiradi.

```
For ([inisializatsiya(nomlash) ]; [ifoda] ; [iteratsiya])

{ instruksiya }
```

Yuqoridagi misolni for takrorlanish operatori bilan yechamiz :

```
using System;

namespace ForStatement {

public class Labels {
```

```
public static int Main ( ) {
for(int i = 0; i < 10; i++)
{ Console.WriteLine("i: {0}", i); }
return 0; }}}
```

Break va continue.

Dastur bajarilishida shunday holatlar bo'ladiki, dastur ishini to'xtashish yoki ma'lum qismini bajarmaslik zarur bo'lib qoladi. Bunday hollarda **break** va continue instruksiyalaridan foydalanish qulay. Agar sizga dastur ishini ma'lum paytda (holatda) to'xtatish, oxirigacha bajarmaslik zarur bo'lsa, u holda **break** dan foydalanish kerak :

```
using System;
namespace BreakContinueStatement {
class Values {
static void Main( ) {
//oddiy sonlar uchun belgi qo'yamiz
bool a;
for(int i =100; i > 1; i --)
{   //belgi qo'yamiz
    a = true;
    for (int j = i-1; j > 1; j--)
    //agar nol qoldiqli bo'luvchi bo'lsa
    if(i%j == 0)
 {  //belgini tashlaymiz
    a = false;
```

```
  // agar birorta bo'luvchi topilmasa
  //nol qoldiqli }
 if(a)
Console.WriteLine("(0} = oddiy son ", i);}
}}}
```

Continue instruksiyasi dastur ishini to'xtatmaydi, balki, u shu takrorlanish bajaradigan amalni bekor qiladi xolos.

```
for (int j = 0; j < 100; j++ ) {
if (j%2 = = 0)
continue;
Console.WriteLine("{0}", j);}
```

Bu misol ekranga barcha 100 gacha bo'lgan toq sonlarni chiqarib beradi.Agarda tekshirish paytida juft son kelib qolsa, u holda hech qanday amal bajarilmaydi va takrorlanish birta keyinga o'tkaziladi.

Cheksiz takrorlanish yaratish.

Cheksiz takrorlanish yaratish uchun shart ifodasini doimo true bo'ladigan qilib tanlash zarur. Bunda har doim shart bajariladi va cheksiz takrorlanish paydo bo'ladi.

Masalan,

```
using System;
namespace C_Sharp_Programming {
class Cycles {
public static void Main() {
String Name;
```

```
while (true){
Console .Write ("Ismingizni kiriting ");
Name = Console.ReadLine();
Console.WriteLine("Salom {0}", name);}
}}}
```

Bu misolda while operatori oldidagi ifoda o’rniga (shart ifodasida) true qiymati berilgan va u cheksiz takrorlanadi. Bunday takrorlanish faqatgina while operatori yordamida emas barcha takrorlanish operatorlari yordamida qilinishi mumkin.

Masalan, for takrorlanish operatori yordamida :

```
using System;
namespace C_Sharp_Progr aiming {
class Cycles {
public static void Main() {
string Name;
for (;true;)
{ Console.Write("Ismingizni kiriting " ) ;
  Name = Console.ReadLine ();
  if (Name = = "")
  break;
 Console.WriteLine("Salom ( 0 } ", Name);}
}}}
```

E’tibor bering, ichki if shart operatoridan so’ng break amali qo’llanilgan, bu amal foydalanuvchi ismini kiritganidan so’ng ENTER tugmasi bosishi bilan dastur ishini yakunlaydi va ekranga “Salom foydalanuvchi 1 ” yozuvini chiqarib

beradi.for operatori instruksiyasida shartlar o’rniga bo’sh “; ;” qo’llanilgan, bu cheksiz takrorlanish yaratish imkonini beradi.

2.4. Satrlar

Satr tipi va ular ustida amallar

C# o’zida shu operasiya uchun standart qisqartmani tashkil qiladi. Yozuv literal sifatida ko’rinishga ega bo’lib juft qo’shtirnoqlar bilan yakunlanadi.Quyidagi fragment kodi avvalgi fragmentlarning ekvivalentiga teng, unda satr char turning massivi bilan inisiallashtiradi.

```
String s = "abc";
Console.write();
```

String ob’ekti bilan umumiy metodlardan birining qo’llanilishi – length metodi bo’lib u satrdagi belgilar sonini qaytaradi. Keyingi fragment uch sonini tasvirlaydi,chunki qo’llaniladigan satrda uchta belgi mavjud.

```
String s = "abc";
Console.write();
```

C# har bir literalli satr uchun String sinfli taqdimot yaratiladi,endi siz bu sinfning metodini literalli – satr bilan chaqirishingiz mumkin.Keyingi misol ham uch sonini tasvirlaydi.

```
Console.write(“abc”+abc);
```

Satrlarni qo’shish.

```
String s ="He is" + age + " years old.";
```

Satrida + operatori yordamida uch satr bir satrga umumiylashtirilgan. Metodlarning ekvivalentini topgandan ko’ra uni o’qish va tushinish ancha oson.

```
String s = Console.write("He is ").append(age);
s.append(" years old.").toString();
```

Aniqlashtirish bo’yicha String sinfining har bir ob’ekti o’zgarishi mumkin emas.Qatordagi belgilarni almashtirish va yangi belgilar qo’yish mumkin emas. Bir satrning oxiriga yana birini qo’yish mumkin ham emas.

Operatorlar bajarilishining ketma – ketligi.

Yana oxirgi misolimizga murojaat etamiz.

String s = "He is " + age + " years old.";

age string bo’lmagan holda,peremen int turiga mansub bo’lsa bu satrning kodi translyator magiyasidan ko’proq yakunlanadi.

Keyingi misolni ko’rib chiqamiz:

String s = "four: " + 2 + 2;

Birinchi o’rinda butun sonlarni qo’yilishini xohlasangiz unda qavslardan foydalanish zarur:

String s = "four: " + (2 + 2);

Satrlarni o’zgartirish.

String sinfining to String metodi yoki shaxsiy realizasiyasi mavjud.

```
class Point {
int x, y;
Point(int x, int y) {
this.x = x;
this.y = y;  }
public String toString() {
return "Point[" + x + ", " + y + "]";
} }
class toStringDemo {
public static void main(String args[]) {
Point p = new Point(10, 20);
Console.write ("p = " + p);
} }
```

Misoldan olingan natija:

p = Point[10, 20]

Tenglashtirish.

Agar ikki satrning bir xilligini bilmoqchi bo’lsangiz unda siz String sinfining equals metodidan foydalanishingiz mumkin.Bu metodning alternative formasi equalsIgnoreCase deb nomlanadi.

Keyingi misolda ikki metodning qo’llanilishi illyustrasiyasi keltirilgan.

```
class equalDemo {
public static void main(String args[]) {
String s1 = "Hello";
String s2 = "Hello";
String s3 = "Good-bye";
String s4 = "HELLO";
Console.Write(s1 + " equals " + s2 + " -> " + s1.equals(s2));
 Console.Write (s1 + " equals " + s3 + " -> " + s1.equals(s3));
Console.Write (s1 + " equals " + s4 + " -> " + s1.equals(s4));
Console.Write (s1 + " equalsIgnoreCase " + s4 + " -> " +
s1.equalsIgnoreCase(s4));
} }
```

Natija:

```
Hello equals Hello -> true
Hello equals Good-bye -> false
Hello equals HELLO -> false
Hello equalsIgnoreCase HELLO -> true
```

Nusxalashda satrlarning modifikatsiyasi(o’zgarishi)

String sinfining o’zgartirib bo’lmasligi sababli,har safar satrni modifikatsiyalamoqchi bo’lsangiz uni yoki StringBuilder nusxalashingiz,yoki string sinfining tavsiflanadigan,satrga o’zgartirish kiritib,uni yangi nusxasini yaratadigan metodidan birini ishlatishingiz lozim.

Substring.

Subtring metodi yordamida Stringidan qism satrni ajratib olishingiz mumkin. Bu metod orginal(asl) satrdan chaqiruv chog’ida,ko’rsatilgan indekslar diapazonidagi belgilarning yangi nusxasini yaratadi.Kerakli qism satrning birinchi – simvoli indekslarini ko’rsatish mumkin.Bunda yangi satrga birinchi ko’rsatilgan belgidan boshlab to oxirgi indeks bilan ko’rsatilgan simvolgacha(lekin uning o’zi emas) bo’lgan barcha simvollar(ya’ni belgilar) nusxalanadi.

"Hello World".substring(6) -> "World"

"Hello World".substring(3,8) -> "lo Wo"

Concat

Satrlarning qo’shilishi yoki konkatenatsiyasi concat metodi yordamida bajariladi.Bu metod String sinfining yangi ini yaratadi,unga boshlang’ich satrini butunlay nusxalaydi va oxiridan metod parametri sifatida ko’rsatilgan satrini qo’shadi.

"Hello".concat(" World") -> "Hello World"

Replace

Replace metodiga parametr sifatida ikkita belgi sifatida uzatiladi.Birinchi belgi bilan mos tushuvchi barcha belgilar satrning yangi nusxasida ikkinchi belgi bilan almashtiriladi.

"Hello".replace('l' , 'w') -> "Hewwo"

toLowerCase va toUpperCase

Bu metodlar juftligi mos ravishda boshlang’ich satrning barcha belgilarini kichik yoki katta registrga(kichik yoki bosh harflar) almashtiradi.

"Hello".toLowerCase() -> "hello"

"Hello".toUpperCase() -> "HELLO"

Trim

Trim metodi boshlang’ich satrning boshi va oxirida kelgan bo’sh joy belgilarni yo’qotadi.

“Hello World “.trim() -> "Hello World"

valueOf

Agar siz biror berilganlar tipi bilan ishlasangiz va shu tipning qiymatini o’qilishi qulay shaklga keltirmoqchi bo’lsangiz,dastlab undagi qiymatini satr ko’rinishiga o’tkazishingiz kerak.Buning uchun valueOf metodi mavjud.Bunday static metod C#da mavjud bo’lgan ixtiyoriy berilganlar tipi uchun aniqlangan(barcha shunday metodlar o’zaro moslashgan,ya’ni bitta nomdan foydalanadilar.) Shu sababli ixtiyoriy tip qiymatini satrga aylantirish qiyinchilik tug’dirmaydi.

StringBuilder

StringBuilder – String sinfining egizagi bo’lib, satrlar bilan ishlashda talab etiladigan narsalardan ko’pini taqdim etadi. String sinflari tayinlangan (fiksirlangan) uzunlikdagi belgilar ketma-ketligi bo’lib, ularni o’zgartirib bo’lmaydi. StringBuilder tipidagilar esa shunday belgilar ketma-ketligi, ularni kengaytirish yoki modifikatsiyalash mumkin. C#da ikkala sinf ham keng qo’llaniladi, biroq ko’pchilik dasturchilar faqat String tiplari bilan, “+” operatorini qo’llagan holda ishlashni ma’qul ko’radilar. Bunda C# StringBuilder bilan barcha kerakli amallarni o’zi “sahna ortidan” bajaradi.

Konstruktorlar.

StringBuilderini paranametrlarsiz hosil qilish mumkin, bunda unda o’n oltita belgini saqlash uchun joy ajratiladi, biroq satr uzunligini o’zgartirib bo’lmaydi. Siz yana Konstruktorga butun son uzatishingiz va buferning talab etilgan o’lchami oshkor holda berishingiz mumkin. Va nihoyat, Konstruktorga satr uzatishigiz mumkin, bunda u ga nusxalanadi, qo’shimcha ravishda yana o’n oltita belgi uchun joy ajratiladi. String Builder ning joriy uzunligini length metodini chaqirib aniqlash munkin. StringBufferida satr uchun ajratib qo’yilgan barcha joyni aniqlash uchun esa capacity metodini ishlatish kerak. Quyidagi misolni bu holda izohlab beradi:

```
class String Builder Demo {
public static void main(String args[]) {
StringBuilder sb = new StringBuilder("Hello");
Console.Write ("buffer = " + sb);
Console.Write ("length = " + sb.length());
Console.Write ("capacity = " + sb.capacity());
} }
```

Bu dasturning natijasidan ko’rinib turibdiki StringBufferida satrlar bilan ishlash uchun qo’shimcha joy ajratilgan.

buffer = Hello

length = 5

capacity = 21

ensureCapacity

Agar siz StringBuilderini yaratib bo’lganingizdan so’ng unda ma’lum miqdordagi belgilar uchun joy ajratib qo’ymoqchi bo’lsangiz, buffer o’lchamini o’rnatish uchun ensureCapacity metodidan foydalanishingiz kerak. Ayniqsa, buferga ko’pgina kichikqroq satrlarni qo’shishga to’g’ri kelishini oldindan bilsangiz, bu metodni qo’llash juda foydalidir.

setLength

Agar siz nogahon bufferdagi satr uzunligini oshkor holda o’rnatishingizga to’g’ri kelsa, setLength metodini ishlatishingiz mumkin.Agar sizdagi satr uzunligining kattaroq sonni uzatsangiz bu metod yangi kengaygan satr oxirini nol(0) kodli belgi bilan to’ldiradi sal keyinroq keltiriladigan setCharDemo dasturida setLength metodi buferni qisqartirish uchun ishlatiladi.

charAt va setCharAt

String Bufferidan bitta belgini charAt metodi yordamida ajratib olinadi.Boshqa metod setCharAt satrning berilgan o’rniga(pozitsiyasiga) kerakli belgini yozishga imkon beradi.Bu metodlarning ishlatilishi misol bilan keltirilgan:

```
class setCharAtDemo {
public static void main(String args[]) {
StringBuilder sb = new StringBuilder Hello");
Console.Write ("buffer before = " + sb);
Console.Write ("charAt(1) before = " + sb.charAt(1));
sb.setCharAt(1, 'i');
sb.setLength(2);
Console.Write ("buffer after = " + sb);
Console.Write ("charAt(1) after = " + sb.charAt(1));
} }
```

Ushbu dastur ishga tushirilganda olinadigan natija:

buffer before = Hello

charAt(1) before = e

buffer after = Hi

charAt(1) after = i

append

StringBuilder sinfining append metodi odatda satrli ifodalarda “+” operatori qo’llanilganda oshkormas holda chaqiriladi.Har bir parametr uchun String.valueOf metodi chaqiriladi va uning natijasi joriy StringBuilderiga qo’shiladi.Buning ustiga har safar append metodi chaqirilganda u o’zi bilan birga chaqirilgan StringBufferiga ko’rsatgichni qaytaradi.Bu esa metodni ketma-ket chaqirishlar zanjirini hosil qilishga imkon beradi.Quyidagi misolda shu xossasi ko’rsatilgan.

```
class appendDemo {
public static void main(String args[]) {
String s;
int a = 42;
StringBuilder sb = new StringBuilder (40);
s = sb.append("a = ").append(a).append("!").toString();
Console.Write (s);
} }
```

Misol natijasi:

a = 42!

Insert

Insert metodi append metodi bilan shu jihatlar bir xilki,har bir tip uchun ushbu metodni qo’llash mumkin. Biroq,appenddan farqli ravishda u String.valueOf metodi tomonidan qaytariladigan belgilarni StringBuilderi oxiriga qo’shmaydi balki uni birinchi parametr bilan beriladigan buferning muayyan joyiga qo’yish mumkin.Navbatdagi misolda "there" satri "hello" va "world!" orasiga qo’yiladi.

```
class insertDemo {
public static void main(String args[]) {
StringBuilder sb = new StringBuilder ("hello world !");
```

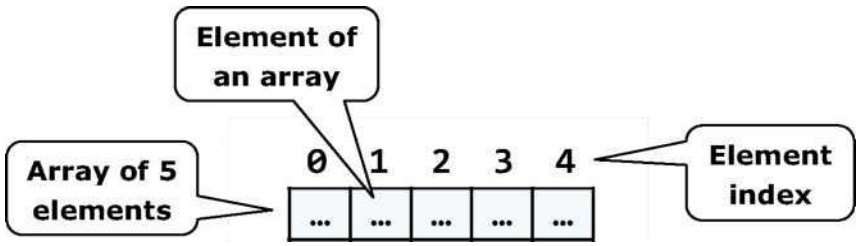
```
sb.insert(6,"there ");
Console.Write (sb); } }
```

Ushbu dastur ishga tushirilsa quyidagi satr chop etiladi:

hello there world!

2.5. Massivlar

Massivlar dasturlash tillari uchun eng muhim hisoblanadi . Ular biz elementlar deb ataydigan o‘zgaruvchilarning massividan iborat.



2.5.1 – rasm. Massivning ko`rinishi

Massivlarning bir qator elementlari C# 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.

Massivlarni e’lon qilishi uchun xotiradan joy ajratish

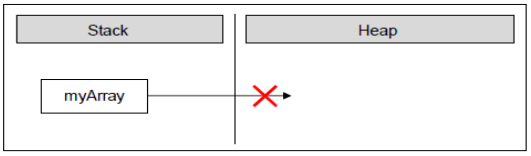
Biz massivni quyidagicha e’lon qilamiz:

```
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.



2.5.2 – rasm. Massivni xotira ajratilmagan paytida

Dasturning amalga oshirish jarayonida myArray o‘zgaruvchisi kiritiladi va unga qiymati nolga teng(qiymatga ega emas) bo‘ladi.

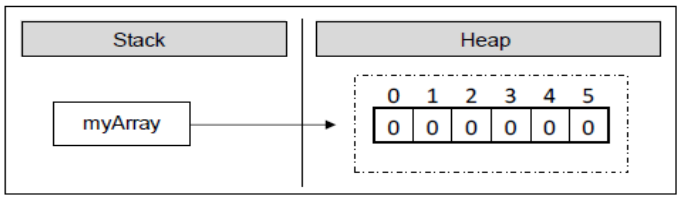
Massiv yaratish - "new" opreatori

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:



2.5.3 – rasm. Massivni stekdagi ko`rinishi

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.

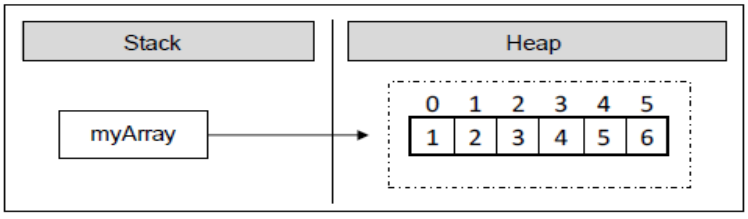
Massiv ishga tushishi va mavjud qiymatlar

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.



2.5.3 – rasm. Massivni stekdagi ko`rinishi

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.

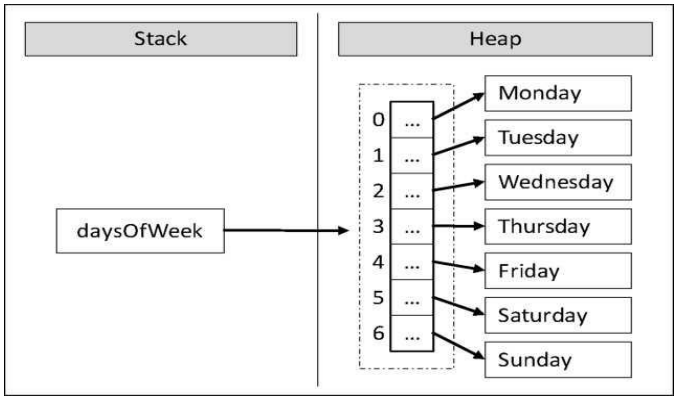
Massivni e’lon qilish va initsializatsiya qilishga –misol

Bu erdagi 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:



2.5.4 – rasm. Massivni stekdagi ko`rinishi

Massivning chegarasi

Massiv elementlari 0 dan boshlab indekslanadi. Birinchi element 0 bilan, 2-element 1 bilan va shu tartibda n-element n-1bilan indekslanadi.

Massiv elementlarini kiritish

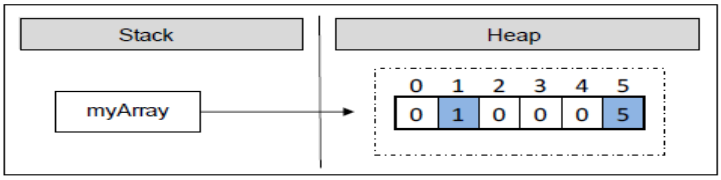
Biz massiv elementlarini ularni indekslaridan foydalangan holda kiritamiz. Har bir elementning indeksi massivning nomidan keyin va to‘rtburchak qavs ichiga joylashadi. Berilgan massivning elementlariga o‘qish va yozish uchun kira olasiz. Massiv elementlarini kiritish doir misol:

```
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.



2.5.5 – rasm. Massivni stekdagi ko`rinishi

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;
}
```

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.

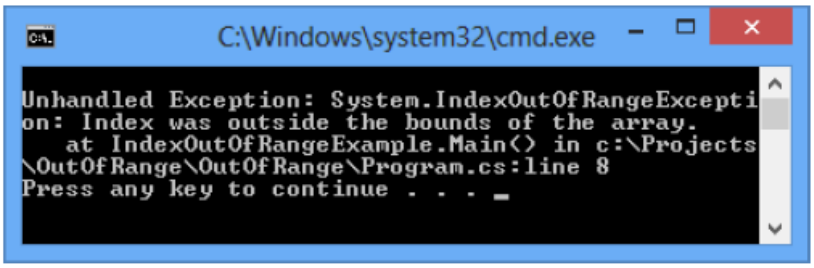
Quyidagi misolda biz massivda mavjud bo‘lmagan elementni kiritamiz.

IndexOutOfRangeExample.cs

```
class IndexOutOfRangeExample
{
   static void Main()
   {
      int[] myArray = { 1, 2, 3, 4, 5, 6 };
      Console.WriteLine(myArray[6]);
   }
}
```

2.5.6 – rasm. Massivni chegara xatoligi

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.



2.5.7 – rasm. Massivni chegara xatoligi

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 xil bo‘ladi.

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
```

2.5.8 – rasm. Massivni teskari tartiblash

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.

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.

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

```
int n = int.Parse(Console.ReadLine());
int[] array = new int[n];
```

sifatida foydalanamiz.

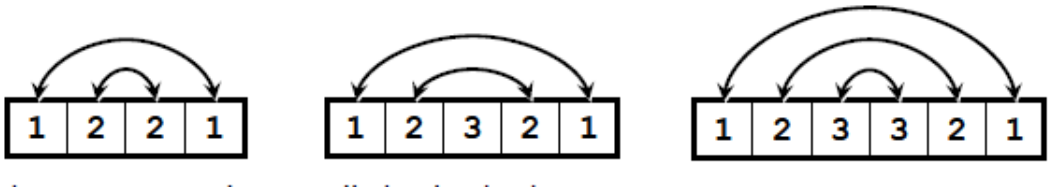
Biz yana massiv ustida takrorlash uchun sikl operatorlaridan foydalanamiz. Har bir takrorlash davomida biz konsoldan o‘qib olingan joriy elementni massivga yuklaymiz. Massivning barcha elementlarini o‘qib olish uchun sikl **n** marta takrorlanadi .

```
for (int i = 0; i < n; i++)
{
   array[i] = int.Parse(Console.ReadLine());
}
```

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.



2.5.9 – rasm. Massivni chegara xatoligi

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.

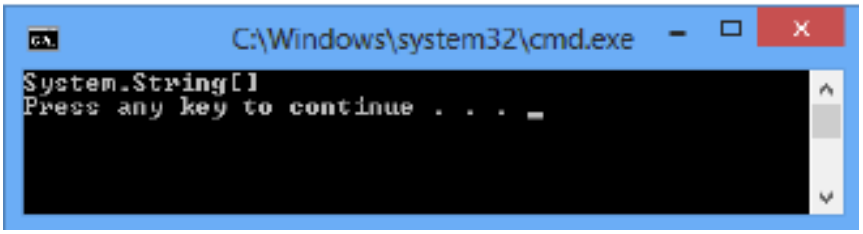
Konsolga massivni chop etish

Ko‘p hollarda 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.



2.5.10 – rasm. Massivni chop et

```
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** sikli yordamida birma – bir massiv elementlarini chop etish. 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
```

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.

"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 eg‘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

```
string[] capitals =
  { "Sofia", "Washington", "London", "Paris" };

foreach (string capital in capitals)
{
  Console.WriteLine(capital);
}
```

Quyidagi misolda massiv ketma – ketligidan foydalanish **foreach** sikli orqali 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

```
Sofia
Washington
London
Paris
```

bajarilgandan so‘ng quyidagi natijaga erishiladi.

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)

Ko‘p o‘lchamli massiv va matritsa

C# dagi har 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

```
int[,] twoDimensionalArray;
```

qilish keltirilgan:

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.

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

```
int[,] intMatrix;
float[,] floatMatrix;
string[,,] strCube;
```

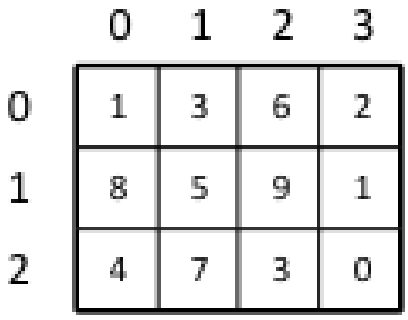
o‘lchamidan bitta kam vergul bilan ajratiladi.

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:



2.5.11 – rasm. Massivni xotiradagi ko`rinishi

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.

Ikki-o‘lchamli massivlarni initsializatsiya qilish

```
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)
```

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 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.

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[0, 0]  matrix[0, 1]  matrix[0, 2]  matrix[0, 3]
matrix[1, 0]  matrix[1, 1]  matrix[1, 2]  matrix[1, 3]
```

matrix nomli massivda 8 ta elementi mavjud bo‘lib, 2 qator va 4 ta ustundan 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]
```

Ko‘p o‘lchamli massivlaring uzunligi

```
int[,] matrix =
{
   {1, 2, 3, 4},
   {5, 6, 7, 8},
};
```

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.

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.

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 ta’minlab beradi. Har bir jarayon ketma –ketligida bu ikki indeksdan foydalangan holda elementlar konsolga chop etiladi (qator va ustun bo‘yicha).

2.6. Funktsiyalar. Fayllar bilan ishlash.

C# da dasturlashning asosiy bloklaridan biri funksiyalardir. Funksiyalarning foydasi shundaki, katta Masala, bir necha kichik bo'laklarga bo'linib, har biriga alohida funksiya yozilganda, Masala, yechish algoritmi ancha soddalashadi. Bunda dasturchi yozgan funksiyalar C# ning standart kutubhonasi va boshqa firmalar yozgan kutubhonalar ichidagi funksiyalar bilan birlashtiriladi. Bu esa ishni osonlashtiradi. Ko'p holda dasturda takroran bejariladigan amalni funksiya sifatida yozish va kerakli joyda ushbu funksiyani chaqirish mumkin. Funksiyani

programma tanasida ishlatish uchun u chaqiriladi, yani uning ismi yoziladi va unga kerakli argumentlar beriladi. () qavslar ushbu funksiya chaqirig'ini ifodalaydi.

Masalan:

foo();

k = square(l);

Demak, agar funksiya argumentlar olsa, ular () qavs ichida yoziladi. Argumentsiz funksiyadan keyin esa () qavslarning o'zi qo'yiladi.

Fayl bu ma’lumotlarning fundamental strukturalaridan biri. Kompyuterlarning dastur bilan ishlashi, tashqi qurilmalar bilan aloqasi fayl strukturasiga soslangandir.

Ma’lumotlarni faylda yozish va o’qish

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 – kattik yoki yumshok magnit disklaridagi baytlarning oddiy ketma-ketligidan iborat.

Mantiqiy fayl strukturasi

Mantiqiy fayl strukturasi – bu, faylni dasturda qabul qilish usulidir. Obrazli qilib aytsak, u biz faylning fizik strukturasiga qarashimiz mumkin bo’lgan «shablon»dir.

Ana shunday shablonlarga misol ko’raylik:

Belgi	***Belgi***	…	***Belgi***	eof

Bu yerda har bir belgi faylning bir yozuvi, eof esa faylning oxirini bildiradi. Yunalish tugmasi belgisini marker deb ataylik, markerni dastur ko'rsatmasi bilan faylning ixtiyoriy yozuviga keltirib quyish va o'sha yozuvni o'qish mumkin. Bunda har bir o'qishda faqat bitta belgi uqiladi.

Yuqorida fayl yozuvi sifatida belgili tipni oldik, endi fayl yozuvi sifatida strukturani olamiz:

Bu yerda bizda strukturada quyidagicha malumotlar bor desak

Mirzayev	Buxoro,15	1974
Tolib	P.Neruda,12	1985
MrX	Nakshband,45	1980
Oka	Galaba,13	1976

Har bir yozuvda 3 ta maydon bor va yozuvlar soni esa to'rtta.

Agar biz fayldan 1-chi yozuvni o'qishni buyursak, unda bir o'qishda 3 ta o'zgaruvchida Mirzayev, Buxoro 15, 1974 ma'lumotlari birdaniga o'qiladi. Buni qanday amalga oshirishni quyida ko'ramiz.

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.

Fayl tipidagi o’zgaruvchi fayl o’zgaruvchisi deyiladi, u faylning mantiqiy nomini belgilaydi va u mantiqiy fayl bilan tashqi(fizik) fayl o’rtasida «vositachi» vazifasini o’taydi.

Yuqoridagi barcha turdagi fayllar ustida, umuman olganda quyidagi amallarni bajarish mumkin va bu amallar uchun quyidagi maxsus metodlar ishlatiladi:

Fayllarda ma’lumotlarni yozish va o’qishni misollar orqali qarab chiqamiz.

1- misol. uqish.txt fayldan ikkita sonni o’qib, ularni yig’indisini yozish.txt fayliga yozish dasturini ko’rib chiqamiz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
namespace ConsoleApplication1 {
   class Program    {
      static void Main(string[] args) {
         StreamWriter yozish = new StreamWriter("d:\\yozish.txt");
         StreamReader uqish = new StreamReader("d:\\uqish.txt");
         yozish.WriteLine(int.Parse(uqish.ReadLine()) +
          int.Parse(uqish.ReadLine()));
         yozish.Close();
          uqish.Close();
      }    }
```

Dastur natijasi: dasturni ishlatishdan oldin uqish.txt faylda 2 ta son kiritish kerak, so’ng dastur ikkita sonni o’qib, ularni yig’indisini yozish.txt fayliga yozadi. Natijani ko’rish uchun yozish.txt faylni ochib ko’rish kerak

2-misol. fayl davomidan ma’lumotlarni yozish.

```
class Program {
    static void Main(string[] args){
      /*textFile.txt faylining yangi satridan "yangi satr" so'zini qo'shadi
      agar bu fayl bo'lmasa o'zi yaratib "yangi satr" so'zini qo'shadi*/
      StreamWriter a;
      a = File.AppendText("d:\\textFile.txt");
      a.WriteLine("yangi satr");
      a.Close();
    }   }
```

3- misol. fayldan ma’lumotlarni bitta bitta belgi orqali oqish va chiqarish.

```
class Program   {
    [STAThread]
    static void Main(string[] args) {
      StreamReader uqish;
      try   {   uqish = new StreamReader("d:\\textFile.txt");     }
      catch  {   Console.WriteLine("Faylni ochishda xatolik bor");
        Console.ReadKey();    return;   }
      int ch;
      while ((ch = uqish.Read()) != (-1))
      {    Console.WriteLine((char)ch);   }
      Console.ReadKey();       }   }
```

4- misol. Faylda ma’lumotlarni binar yozish va o’qizh.

```
class Program   {
    static void Main(string[] args) {
      FileStream a = new FileStream("d:\\a.txt", FileMode.CreateNew);
      BinaryWriter w = new BinaryWriter(a);
      //faylga ma'lumotlarni yozamiz
      for (int i = 1; i < 20; i++) {
        w.Write( (int) i);   }
      //faylni yopish
```

```
        w.Close();
        a = new FileStream("d:\\a.txt", FileMode.Open, FileAccess.Read);
        BinaryReader r = new BinaryReader(a);
// fayldan ma’lumotlarni o’qiymiz
        for (int i = 1; i < 20; i++) {
           Console.WriteLine(r.ReadInt32());  }
        r.Close();
           Console.ReadKey();
     }   }
```

5- misol. fayldan ma’lumotlarni satrma - satr o’qish.

```
class Program {
     static void Main(string[] args){
        FileStream a = new FileStream("d:\\textFile.txt", FileMode.Open,
FileAccess.Read);
        StreamReader b = new StreamReader(a);
        string s;
        while ((s = b.ReadLine()) != null){
           Console.WriteLine(s);  }
        b.Close();
        Console.ReadKey();
     }   }
```

Xulosa

Ikkinchi bob, C# tilida strukturali dasturlash deb nomlangan. Bu bobning oltita bo`limi mavjud. Bobimizning birinchi bo`limi C# tili sintaksisi, standart tiplar deb nomlangan bo`lib, unda dasturlash tilida yoziladigan qoida (sintaksis) haqida ma`lumot keltirilgan. Bazaviy (standart) tiplar haqida ham. Bobning ikkinchi bo`limi esa, ifoda, intruksiya va operatorlar deb nomlangan. Bu bo`limda ifodalar va ularning turlari keltirilgan. Inkrement, decrement va h.k. Uchinchi bo`limimiz, boshqaruv operatorlari deb nomlangan. Bu bo`limda goto, if, if..else, while, for, do..while, break,continue, switch operatorlarining vazifalari haqida ma`lumotlar keltirilgan. To`rtinchi bo`limimiz satrlar bo`lib, unda matn ustida

ishlaydigan jarayonlar yoritilgan. Beshinchi bo`limimiz massivlar bo`lib, unda massiv elementlari haqida ma`lumot keltirilgan. Oltinchi bo`limimiz funktsiyalar. Fayllar bilan ishlash bo`lib, unda mustaqil ishlaydigan funktsiyalar yaratish, fayldagi ma`lumotlarni o`qish va yozish haqida ma`lumotlar berilgan.

XOTIMA

Bitiruv malakaviy ishning birinchi bobi “Dasturlash tillari” bo`lib unda uchta bo`lim mavjud. Birinchi bo`limi, Dasturlash tillarining evolyutsiyasi bo`lib, unda hozirgi vaqtga qadar yaratilgan tillar haqida ma`lumot keltirilgan. Bobning ikkinchi bo`limida esa, .Net platforma. Bu bo`limda Visual Studio muhitida ishlashda platforma tushunchasi, bir tilnda yozilgan ikkinchi tilga o`tkazila olishi haqida ma`lumot keltirilgan. Platformadagi buyruqlar barcha tillarga ishlata olishi haqida ham ma`lumot keltirilgan. Bobning uchinchi bo`limi, Visual Studio muhiti va unda ishlash deb nomlangan. Bo`limda muhit bilan tanishish, muhitdagi imkoniyatlar haqida ma`lumotlar keltirilgan.

Ikkinchi bob, C# tilida strukturali dasturlash deb nomlangan. Bu bobning oltita bo`limi mavjud. Bobimizning birinchi bo`limi C# tili sintaksisi, standart tiplar deb nomlangan bo`lib, unda dasturlash tilida yoziladigan qoida (sintaksis) haqida ma`lumot keltirilgan. Bazaviy (standart) tiplar haqida ham. Bobning ikkinchi bo`limi esa, ifoda, intruksiya  va operatorlar deb nomlangan. Bu bo`limda ifodalar va ularning turlari keltirilgan. Inkrement, decrement va h.k. Uchinchi bo`limimiz, boshqaruv operatorlari deb nomlangan. Bu bo`limda goto, if, if..else, while, for, do..while, break,continue, switch operatorlarining vazifalari haqida ma`lumotlar keltirilgan. To`rtinchi bo`limimiz  satrlar bo`lib, unda matn ustida ishlaydigan jarayonlar yoritilgan. Beshinchi bo`limimiz  massivlar bo`lib, unda massiv elementlari haqida ma`lumot keltirilgan. Oltinchi bo`limimiz funktsiyalar. Fayllar bilan ishlash bo`lib, unda mustaqil ishlaydigan funktsiyalar yaratish, fayldagi ma`lumotlarni o`qish va yozish haqida ma`lumotlar berilgan.

FOYDALANILGAN ADABIYOTLAR

1. O‘zbekiston Respublikasi Prezidentining “Oliy ta’lim tizimini yanada rivojlantirish chora-tadbirlari to‘g‘risida”gi qarori. 2017 yil 20 aprel. Toshkent.
2. Robert W. Sebesta, Concepts of Programming Languages, John Wiley & Sons, USA 2015.
3. Fundamentals of Computer Programming With C# (The Bulgarian C# Programming Book). Svetlin Nakov & Co., 2013
4. Andrew Troelsen. Pro C# 5.0 and the .NET 4.5 Framework Sixth Edition 2012 apress.
5. Вагнер Билл. Эффективное программирование на C#. 50 способов улучшения кода. – 3-е изд., перераб. и доп. - М.: ЛОРИ, 2017. - 224 с.
6. Троелсен Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5. – 6-е изд., перераб. и доп. - М.: Вильямс, 2015. – 1312 с.
7. Пахомов Б. И. C# для начинающих. — СПб.: БХВ-Петербург,2014. — 432 с.: ил.
8. Зиборов В.В. Visual C# 2012 на примерах. - М.: БХВ-Петербург, 2013. - 480 с.: ил.
9. Ишкова Элеонора Алексеевна. Самоучитель C#. Начала программирования. – 2-е изд., перераб. и доп. - М.: Наука и техника, 2013. - 496 с
10. Шилдт Гилберт. C# 4.0: полное руководство.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2011. – 1056 с.: ил. - Парал. титл. англ.
11. Фленов М. Е. Библия C#. – 2-е изд., перераб. и доп. –СПб.: БХВ-Петербург. 2011. -560 с.: ил. + CD-ROM
12. Зиборов Виктор Visual C# 2010 на примерах. - М.: "БХВ-Петербург", 2011. - 432 с.: ил.
13. Подбельский В. В. Язык C#. Базовый курс. - М.: Финансы и статистика, Инфра-М, 2011. - 384 с.
14. Троелсен Эндрю. Язык программирования C# 2008 и платформа .NET 3.5. – 4-е изд., перераб. и доп. - М.: Вильямс, 2010. - 1344 с.
15. Неш Трей. C# 2010: ускоренный курс для профессионалов. : Пер. с англ. – М.: ООО “И.Д.. Вильямс”, 2010. – 592 с.: ил. – Парал. титл. англ.
16. Шилдт Гилберт. C# 3.0: полное руководство.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2010. – 992 с.: ил. - Парал. титл. англ.
17. Агуров Павел. C#. Сборник рецептов. - М.: "БХВ-Петербург", 2012. - 432 с.
18. Джозеф Албахари , Бен Албахари. C# 3.0. Справочник. - М.: БХВ-Петербург, 2012. - 944 с.
19. Джозеф Албахари , Бен Албахари. C# 3.0. Справочник. - М.: БХВ-Петербург, 2013. - 944 с.
20. Ватсон Бен. C# 4.0 на примерах. –СПб.: БХВ-Петербург, 2011. 608 с.
21. Николай Секунов. Самоучитель C#. СПб: БХВ-Петербург, 2001. - 576 с.: ил.