O.I. Jalolov, Sh.M. Sharipov

C# dasturlash tilida fayllar bilan ishlash

(uslubiy qo'llanma)

BUXORO 2014

Ushbu uslubiy qo’llanmada C# dasturlash tilining asosiy operatorlari va fayllar ustiba bajariladigan asosiy amallar keltirilgan. Bu uslubiy qo’llanmadan o’quv jarayonida foydalanish mumkin va mustaqil o’rganuvchilar uchun mo`ljallangan.

Mualliflar: Sh.M. Sharipov – BuxDU Amaliy matematika va informa-tika ta’lim yo’nalishi 4-kurs talabasi.

O.I. Jalolov – Buxoro davlat universiteti “Amaliy matematika va axborot texnologiyalari kafedrasi” dotsenti.

Taqrizchilar:

M. Nurullayev – BuxMTI Informatika va axborot texnologiyalari kafedrasi katta o`qituvchisi.

X. Xayatov – BuxDU Amaliy matematika va axborot texno-logiyalari kafedrasi katta o’qituvchisi

Ushbu uslubiy qo`llanma Buxoro davlat universitetining Amaliy matematika va axborot texnologiyalari kafedrasining 2014 – yil __ – _____dagi yig`ilishiga hamda Fizika – matematika fakultetining 2014- yil __ - _____dagi __- sonli Kengash yig`ilishida ko`rilgan va nashrga tavsiya qilingan.

KIRISH

XX asrning 80 - yillaridan oldin ishlab chiqilgan kompyuterlar uchun katta dasturiy sistemalarni ishlab chiqish juda ham mushkul vazifa edi. Buning eng katta sababi shu davrga xos bo'lgan kompyuterlarning imoniyatlarining chegaralanganidadir. Dasturiy komplekslarni ishlab chiqishda asosiy cheklanishlar kompyuter tezkor xotirasining sig'imi, ma'lumotlarni ikkilamchi xotira qurilmalaridan (magnit lentalar, barabanlar va x.k.) o'qish tezligi, prostessorning ishlash tezligi (ularning takt chastotalari bir necha yuz mikrosekund bo'lgan) bilan bog'liq. Bu davrdagi kompyuterlar xalq ho'jaligining hisob-kitob bilan bog'liq bo'lgan masalalarini yechish uchun mo'ljallangan edi. Kompyuterlarda boshqa xarakterdagi masalalarning yechishning iloji yo'q edi. Dasturchilar uchun shu kompyuterlarda yechilayotgan masala uchun berilgan ma'lumotlarni hisobga olgan holda masalaning yechish algoritmini ishlab chiqish birinchi o'rindagi vazifa hisoblangan. Bu borada N. Virtning mashhur ***ma'lumotlar+algoritm=dastur*** formulasini yodga olish yetarli.

80-yillardan keyin ishlab chiqilgan kompyuterlarning imkoniyatlari kengayib, ishlab chiqarish narxi keskin pasaygani tufayli ulardan foydalanish samarasi ortib bordi. Natijada, dastlab matn va grafiklar uchun muharrirlari ishlab chiqildi, keyinchalik multimedia qurilmasi yordamida tovushli ma'lumotlarni qayta ishlash imkoniyatlari ham paydo bo'ldi. Kompyuter tasviriy imkoniyatlarining kengayishi foydalanuvchilar uchun qulay bo'lgan grafik muloqot interfeysini yaratilishiga sabab bo'ldi. Natijada yuqori murakkablikdagi katta amaliy dasturlarni yaratish foydali hamda zarur bo'lib qoldi. Dasturiy vositalarni ishlab chiqish uchun qurol sifatida yuqori bosqichli algoritmik tillardan foydalanildi. Bu dasturiy vositalar dasturchi va dasturchilar guruhi imkoniyatlarini kengaytirib, dasturiy mahsulotlarning murakkablik darajasining ortishiga ham sabab bo'ldi.

Hozirgi kunda C# dasturlash tili yuqori bosqichli dasturlash tillari ichida eng samarali dasturlash tillaridan hisoblanadi. C# dasturlash tilida dastur tuzish uchun Visual Studio .NET muhitidan foydalanamiz. C# dasturlash tili obektga mo'ljallangan dasturlash tili hisoblanadi.

Ushbu qo'llanmada C# dasturlash tili va uning operatorlari haqida ma'lumotlar keltirilgan. Bilamizki ma'lumotlarni oqish va yozish turli xil oqimlar orqali amalga oshiriladi. Ma'lumotlarni fayl oqimlari orqali o'qish va yozish juda katta ahamiyatga ega. Sababi agar bir ma'lumotlarni ekran orqali o'qisak va yozsak unda har safar dasturni ishga tushurganda ma'lumotlarni qaytadan kiritishga to'g'ri keladi bu esa ancha qiyinchiliklar tug'diradi. Bundan tashqali natijalarni tahlil qilishda, ularni solishtirishda ham fayl oqimi bilan ishlash juda ko'p qulayliklarni olib keladi. Birinchi bobda konsol rejimda ma'lumotlarni ekran orqali kiritish va chiqarish amallari ko'rib chiqilgan. C# tilining sintaksisi, ma'lumotlar tiplari haqida ma'lumotlar keltirilgan. Har bir operatorning funksional imkoniyatlari misollar orqali tushuntirib berilgan. C# tilida ifoda, intruksiya va operatorlar haqida ma'lumotlar keltirilgan. Tarmoqlash, takrorlash operatorlari va ularni har xil variantlari misollar orqali tushuntirib berilgan. Bundan tashqari har bir operatorning funksional imkoniyatlari misollar orqali tushuntirib berilgan.obyektga mo'ljallangan dasturlash tillarining asosini sinf tashkil qiladi.

Qo'llanmaning ikkinchi bobi Microsoft Visual Studio 2010 muhitida C# dasturlash tili yordamida fayllar bilan ishlash, ular ustida bajariladigan amallarga bag'ishlangan. Bunda fayl va kataloglar ro'yxati ustida bajariladigan barcha metodlar o'rganib chiqilgan. Bunda fayl va kataloglar ustida bajariladigan metodlar o'rganilib misollar orqali tushuntirib berilgan. Ma'lumotlarni faylda yozish va o'qish oqimlari bo'yicha mavjud metodlar o'rganilib misollar orqali tushuntirib berilgan. Har bir operatorning funksional imkoniyatlari misollar orqali tushuntirib berilgan.

I-BOB. C# tilining sintaksisi va asosiy operatorlari.

1.1. Consol rejimi. C# tilining sintaksisi. Ma’lumotlar tiplari.

Visual Studio.NET sistemasida dasturni kompilyatsiya qilish va bajarishning bir necha usuli bor. Ko’p hollarda dasturchilar dasturni alohida kompilyatsiya qilib bir nechta klavishalar kombinatsiyalari orqali ishlatishga o’rganishgan.

<Ctrl>+<Shift>+<B> tugmalarini bosish orqali yoki menyuning Build->Build Solution qismini tanlash orqali dasturni kompilyatsiya qilish mumkin. Alternativ variant sifatida instrumentlar panelidagi Build tugmasini bosish ham mumkin.

Dasturni kompilyatsiya qilmasdan ishlatish uchun <Ctrl>+<F5> tugmasini yoki menyuning Debug->Start Without Debugging qismini tanlash yoki panel instrumentlar qismidagi mos tugmani bosish lozim.

```
using System;

class Hello

{

 static void Main(string[] args)

{

 Console.WriteLine(“Hello”);

}

}
```

C# tilida yozilgan dasturni ishlatish uchun

1. Kodni fayllar sistemasida biror nom bilan saqlash lozim (hello.cs)

2. Kommandalar satrida csc /debug hello.cs buyrug'ini bajarish lozim

Ushbu buyrug' bajarilgach, natijaviy .exe kengaytmali fayl hosil bo'ladi. Agar kompilyatsiya jarayonida xatolik yuzaga kelsa, ma'lumot chiqariladi. /debug parametri bajariluvchi faylga maxsus simvollarni joylashtiradi. Natijada exe faylni qayta ishlovchi dasturda taxlil qilinayotganda stekni kuzatib borishlari mumkin.

3. Dasturni ishlatish natijasida, ekranga

Hello yozuvi chiqariladi.

C# dasturlash tilida Consol rejimda dastur tuzish uchun yangi loyiha yaratamiz (**File/New Project/Visual C#/ Console Application**). Ushbu loyihamiz- ning nomini masalan "1-misol" deb nomlaymiz. Bizga C# kodini yozish uchun yangi oyna ochiladi.

Consol rejimida 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 oladi. Odatda kiritish, chiqarish standart Consol-da amalga oshiriladi (agar u bo'lmasa, masalan oynali masalalarda chiqarish amalga oshirilmaydi), lekin kiritish va chiqarish oqimlarini o'zgartirish mumkin. Consol bilan ishlashda asosan 4 metod ishlatiladi: Read, ReadLine, Write, WriteLine, birinchi ikkitasi kiritish, qolgani chiqarish metodlari hisoblanadi.

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

```
int i = Console.Read();

Console.WriteLine(i) ;
```

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 amalga oshmagan bo'lsa) qiymatini qaytaradi. Masalan:

```
string s = Console.ReadLine();

Console.WriteLine("Kiritilgan satr : " + s) ;
```

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

```
Console.Write (I);

Console.Write (0.75<3) ;

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

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

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

Writeline metodining farqi shundaki, u keyingi (yangi) satrdan boshlab o'ziga yuborilgan o'zgaruvchi qiymatini ekranga chiqarib beradi.

Endi ushbu metodlarga misolni kodini to'liq keltiramiz:

```
using System;

namespace _01_misol

{
```

```
class Program
{
static void Main(string[] args)
{
Console.Write("1-misol");
Console.ReadKey();
} } }
```

Bu dastur hozircha hech qanday ish bajarmaydi, u faqat ekranga 1-misol degan yozuvni chiqaradi.

C# dasturlash tilining alfaviti quyidagilardan iborat. Alfavit (yoki lite-rallar 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 xizmat qiladi, ya'ni leksemalarni tuzishda. Leksemalarning 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.

1.1.1-jadval. Kalit so’zlar va nomlar.

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

C# tilida ma’lumotlar tiplari.

C# tili juda tiplashgan til hisoblanadi. Uni ishlatish paytida har bir o’zgaruv-chi 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’l-chamli 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.

Yaratilgan tiplar.

1.1.2-jadval. C# tilida yaratilgan tiplar va ularning o’lchamilari.

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*10^30 8	12 баут, aniqlik — 28 razryadli

Yaratilgan 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 xoxlasangiz, 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 hisoblagichlar 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”.

Massivlar.

C# da massivlar boshqa C dasturlash tillaridagi massivlardan ancha farq qiladi. Buni misollar yordamida ko’rib o’tamiz.

int [] k ; // k – massiv.

K = new int [3] ; // massiv 3 ta int tipiga tegishli elementdan iborat.

K [0] = -5;

K [1] = 4 ;

K [2] = 1; // massiv elementlarini e’lon qilamiz.

// massivning uchinchi elementini chiqaramiz

Console.Writeline(k[2]+””);

Yuqoridagilardan ko’rinib turibdiki, massiv quyidagicha e’lon qilinadi :

Int [] k;

Quyidagisi esa xato hisoblanadi :

int *k*[]; //xato!

int k [] ; //xato !

Ko’p o’lchovli massivlar.

Massivlarning ko’p o’lchovli e’lon qilish uchun faqatgina “,” belgisini n marotaba (n o’lchovli uchun), [] lar sonini n marotaba (n darajali) yozish kerak.

Masalan, 2 o’lchovli massiv e’lon qilish uchun :

Int [,] k;

deb e’lon qilish yetarli. 2 darajali massiv uchun

Int [] [] k;

deb e’lon qilish yetarli.

1.2. C# tilida ifoda, instruksiya 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

O'tish operatorlari.

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

1. 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 instrusiyaga 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:

```
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 a ga kichigini b ga 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 Program
{static void Main(string[] args)
{int n1 = 5;
int n2 = 0;
if ( (n1 = = 5) && (n2 = = 5))
Console.WriteLine(" Salom");
else
Console.WriteLine("Yoshlar");
If ((n1 = = 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, switchda 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 (myValue)

case 10:

Console.WriteLine("myValue teng 10") ;

break;

case 20:

Console.WriteLine("myValue teng 20");

break;

case 30: Console.WriteLine("myValue teng 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.

C va **C++** tillarida keyingi **case** amaliga avtomatik o'tishingiz mumkin, agarda oldingi **case** amalining oxirida **break** yoki goto operatorlari yozilmagan bo'lsa. Shunday qilib, **C** va **C++** da quyidagicha yozish mumkin :

Case : statement 1 ;

Case : statement 2;

Break ;

Bu misolda **statement 1** dan so'ng **statement 2** ga avtomatik tarzda o'tiladi (**C++** da). C# da bu dastur ishlamaydi, chunki **C#** tili sintaksisida **case1** dan **case2** ga ikki xil vaziyatda o'tish mumkin : agarda birinchi amal bo'sh bo'lsa (**case** dan so'ng hech qanday qiymat tekshiriladi) yoki **break**, goto operatorlari yordamida. 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 User1");
break;
case 1 :
Console.WriteLine("Salom User2");
break;
case 2:
Console.WriteLine("Salom User3");
break;
default:
Console.WriteLINE("Salom yangi foydalanuvchi");
break;
}
```

Quyida iqtisodiy masalani yechish usuli berilgan :

```
using System;
namespace C_Sharp_Programing
{
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 "user1":

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. Labelga o’tish bajariladi.

Masalan:

```
using System;

public class Labels
```

```
{
public static int Main( )
{
int i = C;
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;
public class Labels
{
```

```
public static int Main( )

{

int i = 0;

while(i < 10)

i++;

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

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) ;

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] ; [i ni ortirish])

{

instruksiya

}

Yuqoridagi misolni for takrorlanish operatori bilan yechamiz :

using System;

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;
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_Programing
{
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_Programing
{
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.

1.3. Sinflar, metodlar, xususiyatlar.

Sinflar. Sinflar – har bir obyektga yo’nalritilgan dasturlash tilining yuragi hisoblanadi. Shuni ta’kidlab o’tish lozimki, sinflar usullar va ma’lumotlar uchun *kapsula*(yig’imi) vazifasini bajarib, ularni qayta ishlanishida asos hisoblanadi.

Sinflarni aniqlash.

Agarda siz **C++** yoki **Java** dasturlash tillari bilan tanish bo’lsangiz, siz uchun **C#** tilida sinflarni aniqlashning sintaksisi qiyinchilik tug’dirmaydi. Yaratayotgan sinfingiz oldiga class kalit so’zini qo’yib, **{ }** orasiga shu sinf elementlarini (a’zolarini) yozishingiz kerak.

```
class Uchburchak
{
private long a,b,c;
}
```

Bu oddiy sinf bo’lib, unda faqat **a, b, c** o’zgaruvchisi berilgan.

Sinflar ishlatilishi.

Faraz qiling, biror firma ishchisi haqida ma’lumot kirityapsiz. Bunda ishchining har xil xususiyatlarini (yoshi, familiyasi, ismi, tug’ilgan yili va hokazo) kiritish zarur bo’ladi. Buning uchun har bir ma’lumotni alohida yozish va keyingi ishchining ma’lumotlarini kiritish uchun joriy ishchining ma’lumotlarini o’chirish kerak bo’ladi. Sinf esa barcha muammolarni hal etadi.

Masalan: shu ishchining (barcha ishchilarning) xususiyatlari o’zgaruvchi sifatida sinf elementiga qo’shiladi

```
using System;

namespace test

{

//sinf boshi

class Worker

public int age=0;

public string name;

}

//sinf oxiri

class Test

{

static void Main(string[] args)

{

Worker wrkl = new Worker();

wrkl.age=34;

wrkl.name="Karim";

Console .WriteLine ((wrkl .name)+", " +wrkl.age) ;

} }
```

Dasturni ishga tushiring. Ekranga “Karim , 34 ” degan xabar chiqadi. Keling dasturimizni chuqurroq o’rganib chiqamiz.

Birinchi navbatda …

class Worker

public int age=0;

public string name;

qismida biz sinfimizni aniqladik(yaratdik). Sinfimizda ikkita age va name maydonlari(o’zgaruvchilari) berilgan. E’tibor bering, C/C++ dan farqli o’laroq biz C# da ba’zi bir boshlang’ich qiymatlarni berishimiz mumkin. Lekin boshlang’ich qiymatlarni berish shart emas. O’zgaruvchilar oldin esa **public** kalit so’zini berdik. Bu esa C++ dagi kabi C# da ham bu o’zgaruvchi (funksiya bo’lishi ham mumkin) ning sinfdan tashqarida ham ishlatilishi mumkinligini bildiradi. Agarda biz o’zgaruvchidan oldin biror bir kalit so’z ishlatmasak, u holda bu o’zgaruvchi **private** kalit so’zini avtomatik tarzda o’zlashtiradi va sinfdan tashqarida ishlamaydi. Bu haqda keyinroq yana chuqurroq tanishib o’tamiz.

Keyingi satrda esa

…

Worker wrkl = new Worker();

…

sinf nusxasini kuche (xotira qismi) ga kiritib, unga murojaat yubordik.

Keyingi satrda esa

…

wrkl.age=34;

wrkl.name="Sharipov";

Console .WriteLine ((wrkl .name)+", "+t+wrkl.age) ;

...

sinfimizni ishga tushirdik va ba’zi qiymatlarni o’zlashtirdik.

Sinf tarkibi.

Common Type System da aniqlangan tiplar sinflarda qo’llab quvvatlanadi va quyidagi ko’rinishda bo’ladi :

Maydon. Bu o’zgaruvchi o’zida bir necha qiymatni saqlaydi. Obyektga yo’naltirilgan dasturlash tillarida bu o’zgaruvchilar obyekt ma’lumotlari deb ham yuritiladi. Maydonga bir necha modifikator qo’llash mumkin (uni qanday ishlatishingizdan qat’iy nazar). Bu modifikatorlarga **readonly**, **static va const** lar kiradi.

Metod(usul). Bu real (amaldagi) kod bo’lib, u obyekt ma’lumotlariga (maydonga ham) ta’sir ko’rsatadi. Hozir biz bu metod tushunchasi bilan obyekt ma’lumotlarini aniqlashga harakat qilamiz.

Xususiyat. Ba’zida xususiyatlarni aqlli maydon deb atashadi(smart fields). Chunki, xususiyatlar aslida metod, lekin ular sinf foydalanuvchilari uchun maydon bo’lib ko’rinadi.

O’zgarmaslar. Bu shunday o’zgaruvchi maydonki, uning qiymatini hech qachon o’zgartirib bo’lmaydi.

Indeksatorlar. Agar xususiyatlar aqlli maydonlar hisoblansa, unda indeksatorlar aqlli massivlardir. Chunki ular obyektlarni get va set metod-aksessorlari bilan indeksatsiya qilishda qo’llaniladi.

Xodisalar. Xodisalar – Microsoft Windowsning ajralmas qismi bo'lib, sichqoncha qimirlashi, oynalar yopilishi, ochilishi va hokazolarda qo'llaniladi.

Main metodi.

C# da tuzilgan har qanday dasturning hech bo'lmaganda birta sinfida Main metodi bo'lshi shart va unga public yoki static modifikatorlari qo'yilgan bo'lishi kerak. Komplyator uchun Main metodining qaysi sinfda yozilganligi ahamiyatsiz, bu metod yozilgan sinf komplyatsiya tartibiga hech qanday ta'sir ko'rsatmaydi. C# tili shunday aqlliki, o'zi avtomatik tarzda Main metodini izlab topadi va u dastur natijasining ekranga chiqishini ta'minlaydi(dastur bajarilishini ham). Shunday qilib siz Main metodini istalgan sinfga joylashtirishingiz mumkin, lekin buning uchun alohida sinf yaratish tavsiya etiladi. Masalan:

```
class MyClass
private in MyClassId;
{
class AppClass
{
static public void Main()
MyClass myObj =new MyClass( );}  }
```

Ruxsat modifikatorlari.

1.3.1-jadval. Ruxsat modifikatorlari.

Ruxsat modifikatori	Mazmuni, vazifasi
Public	Elementni sinfdan tashqarida va ichki sinflarda ishlatilish mumkin

Protected	Elementni sinfdan tashqarida ishlatish mumkin emas, u faqat ichki sinflarda ishlatilishi mumkin
Private	Element faqat shu sinfda ishlatiladi, u ichki sinflarda ham ishlamaydi
Internal	Element faqat shu komplyatsiya qilinuvchi dastur uchun ishlaydi, bu public va protected modifikatorlarining birlashmasidir

Buyruq satrining elementlari.

Siz Main metodini satr massivini qabul qiluvchi qilib e'lon qilsangiz, buyruq satriga murojaat qilishingiz mumkin. Masalan:

```
using System;

class C ommandLineApp

{

public static void Main(string[] args)

{

foreach (string arg in args]

{

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

} }
```

Qaytarilgan qiymat.

Ko’pincha Main metodi quyidagicha aniqlanadi:

```
class SomeClass
{
public static void Main()
…
}
…
}
```

Lekin siz Main metodini int tipidagi qiymatni qaytaruvchi qilib e’lon qilishingiz mumkin:

```
public static int Main() {
 return 0;
 }
```

Bir necha Main metodlari.

C# mualliflari tomonidan shunday mexanizm qo’yilganki, bu mexanizm bir nechta Main metodini aniqlay oladi. Bu nima uchun kerak dersiz. Sabablardan biri bu – sizning dasturingizga test kodini kiritishdir. Keyin almashtirish kaliti bilan **/main;<Sinf_nomi>,** kerakli sinf nomi beriladi va komplyator uni ishga tushiradi. Masalan:

```
using System;
class Main1 {
```

public static void Main()

{

Console.WriteLine("Main1");

class Main2

{

public static void Main()

{

Console.WriteLine("Main2");

}

Bu dasturni Main1.Main metodi natija chiqaruvchi nuqta sifatida ishlashi uchun quyidagi kodni: *Csc MultipleMain.es/main:Main1* berish zarur. Agar alamashtirish kalit so'zi /main.Main2 deb yozilsa, Main2.Main metodi ishga tushadi.

Sinflarni inisializatsiya qilish va konstruktorlar.

Obyektga yo'naltirilgan dasturlash tillarining enga katta yutug'i (C/C++/C#/Java/J#/VB)- shundan iboratki, siz o'zingiz mustaqil ravishda maxsus metodlarni aniqlashingiz mumkin. Bu metodlar sinflarning nusxasi olinganda chaqirish uchun qo'llaniladi. Bu metodlar konstruktorlar deb ataladi. Qachon foydalanuvchi obyekt nusxasini chaqirganda, uning konstruktori chaqiriladi va u foydalanuvchining shu obyekt ustida boshqa amal bajargunicha unga boshqaruvni qaytarishi kerak. Lekin konstruktor nomini qanday qo'yish kerak? Bu muammoni C# mualliflari quyidagicha hal etishdi : C# konstruktorlarining nomi sinf nomi bilan nir xil bo'lishi kerak. Masalan:

using System;

```
class Constructor1App{

ConstructorlApp()

{

Console.WriteLine("a konstruktor") ;

}

public static void Main()

{

ConstructorlApp app = new ConstructorlApp();

} }
```

Konstruktorlar qiymat qaytarmaydi. Agar siz buni amalga oshirmoqchi bo'lsangiz, komplyator o'zgaruvchining sinf nomi bilan bir xil nomga ekanligini xato sifatida xabar qiladi.

Sinfning statistik elementlari.

Siz sinf elementini yo statistik (statistic member), yo nusxa(instance member) qilib aniqlashingiz mumkin. Bu shuni anglatadiki, har bir sinf nusxasi uchun har bir element nusxalanadi. Agar siz elementni statistic deb e'lon qilsangiz undan faqat birta nusxa olinadi. Statistic element dastur joriy sinfni ishga tushirganidan boshlab to dastur yakunlanganicha "yashaydi". Masalan:

```
using System;

class InstCount{

public InstCount IS

}
```

instanceCount++;

static public int instanceCount;

//instanceCount = 0;

class AppClass {

public static void Main() {

Console.WriteLine(InstCount.instanceCount)

InstCount ic1 = new InstCount ();

Console.WriteLine(InstCount.instanceCount);

InstCount ic2 = new InstCount ();

Console.WriteLine(InstCount.instanceCount) ; }

O’zgarmaslar va o’zgarmas maydon tushunchasi.

O’zgarmas – o’z nomi bilan o’zgarmas bo’lib, u dastur ishlashi davomida hech qachon o’zgarmaydigan maydondir. Biror nimani const etib belgilaganda so’ng ikkita qoidani esda tutish lozim. Birinchi – o’zgarmas dastur davomida dasturchi yoki komplyator (odatiy parametrlar yuklansa) tomonidan kiritiluvchi elementdir. Ikkinchi-o’zgarmas nomi xuddi literal ko’rinishida yozilishi kerak. Maydonni o’zgarmas deb e’lon qilish uchun element oldiga const kalit so’zni yozishingiz yetarli. Masalan:

using System;

class MagicNumbers

public const double pi = 3.1415;

public const int g = 10 ;

class ConstApp {

public static void Main() {

Console.WriteLine("pi = {0}, g = {0}",

MagicNurrbers . pi, MagicKumbers . g) ; } }

O’zgarmas maydon – const deb e‘lon qilingan madon shuni anglatadiki, dasturchi unga doimiy qiymatni o’zlashtirmoqchi bo’lyapti, bu plyus.Lekin u komplyatsiya paytida bu maydonning qiymati aniq bo’lgandagina ishlaydi.

Xulosa. Hozirgi kunda C# dasturlash tili yuqori bosqichli dasturlash tillari ichida eng samarali dasturlash tillaridan hisoblanadi. C# dasturlash tilida dastur tuzish uchun Visual Studio .NET muhitidan foydalanamiz. C# dasturlash tili obektga mo’ljallangan dasturlash tili hisoblanadi. BMIning birinchi bobida C# dasturlash tili va uning operatorlari haqida ma’lumotlar keltirilgan. Ma’lumotlarni kiritish va chiqarish turli xil oqimlar orqali amalga oshiriladi. Bu bobda konsol rejimda ma’lumotlarni ekran orqali kiritish va chiqarish amallari ko’rib chiqilgan. Birinchi bob 3 ta bo’limdan iborat. Birinchi bo’limda C# tilining sintaksisi, ma’lumotlar tiplari haqida ma’lumotlar keltirilgan. Har bir operatorning funksional imkoniyatlari misollar orqali tushuntirib berilgan. Ikkinchi bo’limda C# tilida ifoda, intruksiya va operatorlar haqida ma’lumotlar keltirilgan. Bu bo’limda tarmoqlash, takrorlash operatorlari va ularni har xil variantlari misollar orqali tushuntirib berilgan. Bundan tashqari har bir operatorning funksional imkoniyatlari misollar orqali tushuntirib berilgan. Uchinchi bo’limda sinflar, metodlar, xususiyatlar haqida ma’lumotlar keltirilgan. Bu bo’limda C# dasturlash tilida sinf qanday yaratiladi va obyektga mo’ljallangan dasturlash tilining asosiy xususiyatlari keltrilgan. Har bir operatorning funksional imkoniyatlari misollar orqali tushuntirib berilgan.

II.BOB. C# DASTURLASH TILIDA FAYLLAR BILAN ISHLASH.

2.1. Fayl va kataloglar ro’yxati.

Bizga ma`lumki, fayl deb xotiraning nomlangan sohasiga aytiladi. Faylda turli ma'lumotlar saqlanadi. Har bir fayl bilan fayl ko`rsatkichi degan tushuncha biriktirilgan. Fayl bir necha elementlardan iborat bo`lib, foydalanuvchi faqat faylning ko`rsatkichi ko`rsatayotgan ma'lumotga murojaat qilishi mumkin. Demak, fizik jihatdan biz faqat ketma-ket fayllarga egamiz. Ya'ni biz oldin birinchi, keyin ikkinchi, uchinchi va h.k. ma'lumotlarni o`qishimiz mumkin. Fayl o`z nomiga ega. Masalan, d:\ malumot.txt. C# tili dasturiy vositalari yordamida, ya'ni dasturda ham fayllarni tashkil qilish va undagi ma'lumotlarni qayta ishlash mumkin. Shu paytga qadar, C# dasturlash tilida bir necha o`zgaruvchilarning toifalari bilan ishlab keldik. Bular skalyar, oddiy va murakkab tarkiblashgan toifalardir. Bu toifadagi ma'lumotlar yordamida masalalarni yechishda boshlang`ich ma'lumotlar klaviaturadan operativ xotiraga kiritiladi va natija ekranga chiqariladi. Ulardan boshqa dasturlarda foydalanib bo`lmaydi, chunki ular tizimidan chiqilgandan so`ng ma’lumotlar hech qayerda saqlanmaydi. Bu ma'lumotlarni xotirada saqlash uchun C# dasturlash tilida ma'lumotlarning faylli toifasi belgilangan. Fayl toifasi alohida o`rin egallaydi. Fayl toifasi bilan ishlashda ma'lum tushunchalarni o`zlashtirish talab qilinadi. Birinchidan, fayllar toifasi nega va qachon qo`llaniladi? Maqsad nima? Zaruriyat nimadan kelib chiqyapti? Ikkinchidan, boshqa toifalardan nega katta farqi bor? Bu savollarga faqat foydalanuvchining nuqtai nazaridan qaragan holda javob bera olamiz: 1. Juda ko`p o`zgaruvchilardan foydalanganda ularning qiymatlarini har doim klaviaturadan kiritishda ma'lum noqulayliklarga duch kelamiz. Bunga katta massivlar misol bo`la oladi. 2. Shunday masalalar uchraydiki, oldindan kattaliklarning qiymatlar soni noma'lum bo`ladi (masalan, natijalar), bu kattaliklarni faylga yozish maqsadga muvofiq. 3.Hech qanday toifalar

tashqi qurilmalarga murojaat qilib, ular bilan ishlashga imkon yaratmaydi (dasturiy til muhitida). Va nihoyat, boshqa toifalardan fayl toifasi farqliligi shundaki, u boshqa toifalar tarkibiga kira olmaydi. Fayllarning turlari. Fayllar uchun mo`ljallangan umumiy protsedura va funktsiyalar Faylda saqlanayotgan ma'lumotlar turiga ko`ra, turlarga bo`linadi: 1) toifalashmagan; 2) toifalashgan; 3) matnli. Toifalashgan fayllar bir xil toifali elementlardan tashkil topadi. Ularni faqat ma'lum qurilmalarda uzatish mumkin, lekin ekranda o`qish mumkin emas. Faylning elementlari mashina kodlarida yoziladi va saqlanadi. Toifalashmagan fayllarda turli toifadagi ma'lumotlarni saqlash mumkin. Ular ham mashina kodlari bilan yozilgan bo`lib baytlar to`plamini tashkil qiladi. Matnli fayllar ASCII kodlardan tashkil topgan va qatorlarga ajratilgan bo`ladi. Matnli fayllarda nafaqat faylning yakunida fayl oxiri belgisi, balki har qatorning yakunida maxsus qator oxiri belgisi qo`yiladi. Fayl turidagi o`zgaruvchi fayl o`zgaruvchisi deyiladi, u faylning mantiqiy nomini belgilaydi va u mantiqiy fayl bilan tashqi (fizik) fayl o`rtasida «vositachi» vazifasini o`ynaydi. Fayl turi uchun arifmetik amallar belgilanmagan. Xatto fayllarni solishtirish va bir faylning qiymatini ikkinchi faylga o`zlashtirish amallari ham aniqlanmagan. Har bir turdagi fayllar ustida, umuman olganda, quyidagi amallarni bajarish mumkin va bu amallar uchun maxsus protsedura va funktsiyalar ishlatiladi. Hozir biz katalog va fayllar ro’yxati ustida bajariladigan amallarni qarab chiqamiz.

1.Kompyuterda mavjud mantiqiy disklarni aniqlash uchun GetLogicalDrives() metodidan foydalanamiz. Quyida kompyuterda mavjud barcha mantiqiy disklarni ro’yxatini chiqaruvhi dastur keltirilgan.

```
class Program {

static void Main(string[] args)  {

string[] LogicalDrives = Environment.GetLogicalDrives();
```

```
foreach (string a in LogicalDrives) {
Console.WriteLine(a);
}
Console.ReadKey();
} }
```

Dastur natijasi:

```
C:\
D:\
E:\
F:\
```

Keyingi misolda yuzaga keladigan xatoliklarni oldini olish uchun System.Security.SecurityException metodidan foydalanamiz.

```
class Program {
 [STAThread]
static void Main(string[] args) {
GetLogicalDrives();
Console.ReadLine();
}
static void GetLogicalDrives()  {
try{
string[] a = System.IO.Directory.GetLogicalDrives();
```

```
foreach (string b in a)  {
System.Console.WriteLine(b); }
}
catch(System.IO.IOException)
{
System.Console.WriteLine("xato");
}
catch (System.Security.SecurityException)
{
System.Console.WriteLine("xato 1");  }
Console.ReadKey();
}  }
```

2. Diskdagi kataloglar ro'yxatini chiqarish uchun System.IO. Directory sinfining GetDirectories() metodidan foydalanamiz. Quyida uning dasturi keltirilgan.

```
class Program  {
static void Main(string[] args)
{
try  {
string[] a = Directory.GetDirectories(@"d:\\");
Console.WriteLine("hamma papkalar :{0}.", a.Length);
foreach (string b in a) {
```

```
Console.WriteLine(b);
} }
catch (Exception e)
{
Console.WriteLine("xato: {0}", e.ToString());
}
Console.ReadKey();
} }
```

Dastur natijasi: D diskdagi barcha kataloglarni ro’yxatini chiqaradi.

3. Endi maska orqali diskdagi kataloglar ro’yxatini chiqarishni ko’rib chiqamiz. Ya’ni quyida d diskdagi c harfi bilan boshlanuvchi barcha kataloglar ro’yxatini chiqaruvchi dastur keltirilgan.

```
class Program  {
static void Main(string[] args) {
try{
string[] a = Directory.GetDirectories(@"d:\\", "c*");
Console.WriteLine("barcha c harfi bilan boshlangan papkalar: {0}.", a.Length);
foreach (string b in a) {
Console.WriteLine(b); }     }
catch (Exception e)
{  Console.WriteLine("Xato: {0}", e.ToString());  }
```

```
Console.ReadKey();

} }
```

4. Berilgan katalogdagi fayllar ro'yxatini chiqarish uchun System.IO. Directory sinfining GetFiles() metodidan foydalanamiz. Quyida uning dasturi keltirilgan.

```
class Program {

static void Main(string[] args)  {

try {

string[] a = System.IO.Directory.GetFileSystemEntries(@"d:\\TP71\\");

Console.WriteLine("d:\\TP71 katalogidagi barcha fayllar {0}", a.Length);

foreach (string b in a)    {

Console.WriteLine(b);

}  }

catch (Exception e)

{

Console.WriteLine("Xato: {0}", e.ToString());

}

Console.ReadKey();

}   }
```

Dastur natijasi d:\\TP71 papkadagi barcha fayllarni ro'yxatini chiqaradi.

5. Endi maska orqali katalogdagi fayllar ro’yxatini chiqarishni ko’rib chiqamiz. Ya’ni quyida d:\\Hujjatlar katalogdagi barcha *.doc fayllarni ro’yxatini chiqarishni dasturi keltirilgan.

```
class Program  {
static void Main(string[] args)     {
try  {
string[] a = Directory.GetFiles(@"d:\\Hujjatlar\\", "*.doc");
Console.WriteLine("d:\Hujjatlar\ papkasidagi 'doc' kengaytmali fayllar {0} ta",
a.Length);
foreach (string b in a) {
Console.WriteLine(b);
}  }
catch (Exception e)
{
Console.WriteLine("Xato: {0}", e.ToString());
}
Console.ReadKey();    }  }
```

Dastur natijasi d:\\Hujjatlar\ papkasidagi *.doc kengaytmali barcha fayllar ro’y-xati chiqadi.

6. Endi maska orqali katalogdagi fayllar ro’yxatini chiqarishni qulayrog’ini ko’rib chiqamiz. Ya’ni quyida d:\Hujjatlar katalogidagi barcha a*.txt fayllarni ro’yxatini chiqarishni dasturi keltirilgan.

```
class Program  {

static void Main(string[] args)     {

try  {

string[] a = Directory.GetFiles(@"d:\\Hujjatlar\\", "a*.txt");

Console.WriteLine("d:\\Hujjatlar\\ papkasidagi a bilan boshlanuvchi 'txt' kengayt-mali fayllar {0} ta", a.Length);

foreach (string b in a)     {

Console.WriteLine(b);

}     }

catch (Exception e)

{

Console.WriteLine("Xato: {0}", e.ToString());

}

Console.ReadKey();    }  }
```

Dastur natijasi d:\Hujjatlar\ papkasidagi a bilan boshlanuvchi 'txt' kengaytmali barcha fayllar ro’yxati chiqadi.

Yana bir misolni ko’rib chiqaylik, ya’ni faylni hajmi va yaratilgan vaqtini ham chiqarsin. Quyidagi dasturda bu amallar ko’rib chiqilgan.

```
class Program {

static void Main(string[] args) {

try{
```

```
DirectoryInfo a = new DirectoryInfo("d:\\Cpp_misollar\\");
foreach (FileInfo b in a.GetFiles("*.cpp")){
string name = b.FullName;              //faylning nomi
long size = b.Length;                  //o'lchami
DateTime creationTime = b.CreationTime; //yaratilgan vaqti
Console.WriteLine();
Console.WriteLine( " "+size + "->" +creationTime +" ->"+ name);}  }
catch (Exception e){
Console.WriteLine("Xato: {0}", e.ToString());}
Console.ReadKey(); } }
```

Dastur natijasi quyidagicha bo’ladi. Bunda birinchi fayl hajmi, yaratilgan vaqti va nomi
chiqadi.

```
297->04.04.2014 15:15:45 ->d:\Cpp_misollar\My001.cpp
236->04.04.2014 15:15:46 ->d:\Cpp_misollar\My002.cpp
246->04.04.2014 15:15:46 ->d:\Cpp_misollar\My003.cpp
273->04.04.2014 15:15:46 ->d:\Cpp_misollar\My004.cpp
540->04.04.2014 15:15:46 ->d:\Cpp_misollar\My005.cpp
242->04.04.2014 15:15:46 ->d:\Cpp_misollar\My006.cpp
393->04.04.2014 15:15:46 ->d:\Cpp_misollar\My007.cpp
1603->04.04.2014 15:15:46 ->d:\Cpp_misollar\MyString.cpp
173->04.04.2014 15:15:46 ->d:\Cpp_misollar\rekursiyafibo.cpp
286->04.04.2014 15:15:46 ->d:\Cpp_misollar\satr11.cpp
364->04.04.2014 15:15:46 ->d:\Cpp_misollar\satr222.cpp
380->04.04.2014 15:15:46 ->d:\Cpp_misollar\str1.cpp
869->04.04.2014 15:15:47 ->d:\Cpp_misollar\StrToInt.cpp
```

2.2. Fayl va kataloglar ustida amallar.

Bu bo'limda biz fayl va kataloglar ustida bajariladigan asosiy amallarni ko'rib chiqamiz. Bu uchun biz C# dasturlash tilida qaysi kutubxonadan foydalanamiz, qaysi sinflardan foydalanamiz va qaysi metodlardan foydalanamiz? har biriga alohida to'xtalib o'tamiz. C# dasturlash tilida fayl va kataloglar ustida amallar bajarish uchun juda ko'p sinflar yaratilgan va bu sinflarda fayl va kataloglar ustida amallar bajarish uchun juda ko'p metodlar mavjud. Mana shu metodlardan qanday qilib foydalanish jarayonini misollar orqali qarab chiqamiz.

Kataloglar ustida bajariladigan asosiy amallar System.IO.Directory sinfining metodlari orqali amalga oshiriladi.

1. DirectoryInfo CreateDirectory(string nomi) – yangi katalog yaratish.
2. void Move(string eski nom, string yangi nom)- katalog nomini o'zgartirish yoki katalogni ko'chirish.
3. void Delete(string nom, bool x)- katalogni o'chirish, agar x parametrning qiymati true bo'lsa bu metod katalog ichidagi fayllar bilan birgalikda o'chiradi.
4. bool Exists(stirng nom) – bu metod chin qiymat qaytaradi agar papka mavjud bo'lsa, aks holda yolg'on qiymat qaytaradi.

Endi yuqorida ko'rib o'tilgan metodlarni misollar orqali qarab chiqamiz.

1-misol. D diskda Salom nomli yangi papka yaratish.

```
class Program {

static void Main(string[] args)

{

//d: diskda salom nomli papka yaratadi
```

```
System.IO.Directory.CreateDirectory(@"d:\\salom");
Console.WriteLine("salom");
Console.ReadKey();
} }
```

2- misol. d:\aka papkasini c:\aka joyga ko’shirish.

```
class Program {
static void Main(string[] args)
{
System.IO.Directory.Move(@"d:\\aka","c:\\aka");
Console.WriteLine("dastur ishladi");
Console.ReadKey();
} }
```

3- misol. d:\aka papkasini o’chirish.

```
class Program {
static void Main(string[] args)
{
bool a = System.IO.Directory.Exists(@"d:\\aka");
if (a) {
System.IO.Directory.Delete(@"d:\\aka");
Console.WriteLine("d: diskdagi aka papkasi o'chirildi");
```

```
}
else Console.WriteLine("aka papkasi yo'q");
Console.ReadKey();
}  }
```

4-misol. d:\aka papkasini bor yo yo’qligini tekshirish.

```
class Program {
static void Main(string[] args){
bool a=System.IO.Directory.Exists(@"d:\\aka");
if(a) Console.WriteLine("d: diskda aka papkasi bor");
else Console.WriteLine("d: diskda aka papkasi yo'q");
Console.ReadKey();
}  }
```

5- misol. d:\aka papkasini nomini o’zgartirish

Buning uchun ham papkani ko’chirish amalidan foydalanamiz.

```
class Program  {
static void Main(string[] args)
{
System.IO.Directory.Move(@"d:\\aka","d:\\aka1");
Console.WriteLine("dastur ishladi");
Console.ReadKey();
```

```
} }
```

Fayllarlar ustida bajariladigan asosiy amallar System.IO.FileInfo sinfining metodlari orqali amalga oshiriladi.

1. bool Exists – fayl bor yoki yo’qligini tekshiradi.
2. FileInfo CopyTo(string fayl nomi) – fayldan nusxa olish.
3. FileStream Creat() – yangi fayl yaratish.
4. StreamWriter CreatText() – matnli fayl yaratish.
5. void Delete() – faylni o’chirish.
6. void MoveTo(string fayl nomi)- faylni ko’chirish.

Endi yuqorida ko’rib o’tilgan metodlarni misollar orqali qarab chiqamiz.

1-misol. Fayl bor yoki yo’qligini tekshirish.

```
class Program {

static void Main(string[] args)

{

System.IO.FileInfo a = new System.IO.FileInfo(@"d:\\БМИ.doc");

bool x = a.Exists;

if (x) Console.WriteLine("d: diskda БМИ.doc fayli bor");

else Console.WriteLine("d: diskda БМИ.doc fayli yo'q");

Console.ReadKey();

} }
```

2-misol. Fayldan nusxa olish.

```
class Program {

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

```
{
//salom.txt faylini d: diskdan c: diskga nusxalaydi
System.IO.FileInfo a = new System.IO.FileInfo(@"d:\\salom.txt");
a.CopyTo(@"c:\\salom.txt");
Console.WriteLine("salom.txt faylini d: diskdan c: diskga nusxaladi");
Console.ReadKey();
} }
```

3-misol. Yangi fayl yaratish.

```
class Program {
static void Main(string[] args)
{
//d diskda salom.doc faylini yaratadi
FileStream a = File.Create(@"d:\\salom.doc");
Console.WriteLine("d diskda salom.doc faylini yaratdi");
Console.ReadKey();
}  }
```

4- misol. Matnli fayl yaratish.

```
class Program  {
static void Main(string[] args)
{
```

```
//"d diskda salom.txt matnli fayl yaratiladi"
StreamWriter a = File.CreateText("d:\\salom.txt");
a.WriteLine("salom aka");  //"salom aka" matnni salom.txt fayliga yozadi
a.Close();                          //yozishni tugatish
Console.WriteLine("d diskda salom.txt matnli fayl yaratildi");
Console.ReadKey();
}    }
```

5- misol. Berilgan faylni o’chirish.

```
class Program {
static void Main(string[] args)
{
//d diskdagi ko'rsatilgan faylni o'chiradi
FileInfo a = new FileInfo("d:\\salom.doc");
a.Delete();
Console.WriteLine("fayl o'chirildi");
Console.ReadKey();
}  }
```

6-misol. Berilgan faylni ko’chirish.

```
class Program {
static void Main(string[] args) {
```

```
System.IO.FileInfo X = new System.IO.FileInfo(@"d:\\kitoblar\\Milliy
g'oya\\12.docx");

X.MoveTo("d:\\14.docx");

Console.ReadKey();

}  }
```

2.3. Ma’lumotlarni faylda yozish va o’qish.

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

Fayllar quyidagi masalalarni yechishga asoslangandir:

1. Qiymatlarni boshqa dasturlar foydalanishi uchun saqlab qo’yish;
2. Dasturning kiritish-chiqarish tashqi qurilmalari bilan aloqasini tashkil qilish.

Fayl tushunchasining ikki tomoni bor:

1- Tomondan fayl – tashqi xotiraning biror axborotni saqlovchi nomlangan bir qismidir. Bunday tushunchadagi fayl fizik fayl deb ataladi.

2- Tomondan fayl – bu dasturda ishlatiladigan ma’lumotlarning turli strukturalaridan biridir. Bunday tushunchadagi fayl mantiqiy fayl deb ataladi, yani u bizning tasavvurimiz bilan hosil qilinadi.

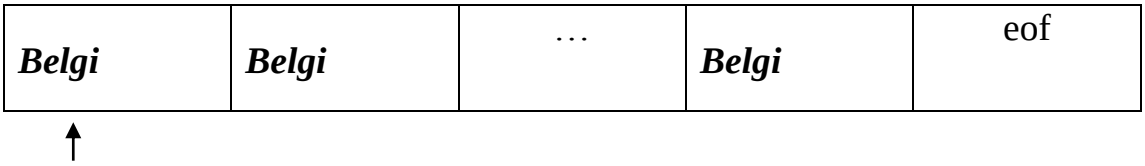
Fizik fayl strukturasi

Fizik fayl strukturasi axborot tashuvchi – qattiq yoki yumshoq magnit disklaridagi baytlarning oddiy ketma-ketligidan iborat.

Mantiqiy fayl strukturasi

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

Ana shunday shablonlarga misol ko'raylik:



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

Yuqorida fayl yozuvi sifatida belgili tipni oldik, endi fayl yozuvi sifatida strukturani olamiz: Bu yerda bizda strukturada quyidagicha malumotlar bor desak

2.3.1-jadval. Fayl strukturasidagi ma'lumotlar ko'rinishi.

Aliev	Buxoro,15	1974
Bozorov	P.Neruda,12	1985
Akramov	Nakshband,45	1980
Karimov	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

Aliev, 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();
}  }
```

6- misol. a.txt fayldagi barcha sonlarni yig’indisini b.txt faylda chiqarish.

```
class Program
```

```
{
static void Main(string[] args)
{
StreamWriter yozish = new StreamWriter("d:\\a.txt");
StreamReader uqish = new StreamReader("d:\\b.txt");
int b, n, k,s=0;
string a;
while ((a = (uqish.ReadLine())) != null)
{
s = s + int.Parse(a);
}
yozish.WriteLine(s+"");
yozish.Close();
Console.ReadKey();
}  }
```

7- misol. a.txt fayldagi eng uzun so’zni b.txt faylda chiqarish.

```
class Program
{
static void Main(string[] args)
{
```

```
StreamWriter yozish = new StreamWriter("d:\\a.txt");

StreamReader uqish = new StreamReader("d:\\b.txt");

int b, n, k,s=0;

string a,x,y; k:=0; x:= "";

while ((a = (uqish.ReadLine())) != null)

{if (a.length()>k) {x = a; k=a.length();}}

yozish.WriteLine(x);

yozish.Close();

Console.ReadKey();

}    }
```

ADABIYOTLAR

1. I.A.Karimov. Yuksak ma’naviyat yengilmas kuch:-Toshkent: O’zbekiston, 2008y. -176 b.
2. I.A.Karimov. O’zbekistonning o’z istiqlol va taraqqiyot yo’li:-Toshkent: O’zbekiston, 1992y. 173-174 b.
3. Трей Неш. C# 2008 усконренный курс для профессионалов: -Москва: Санкт-Петербург, Киев, 2008г. -576с.
4. Павел Агуров. C# Сборник рецептов: -Москва: Санкт-Петербург, 2008г. - 432 с.
5. Дейтел Х, Дейтел П, Листфилд Дж. C# Наиболее полное руководство В Подлиннике: -Москва: Санкт-Петербург, 2006г. -1056 с.
6. Лабор В. В. Создание приложений для Windows: -Москва: Харвест, 2003. - 384 с.
7. Шилдт Герберт. Полный справочник по C#:-Москва: Издательский дом "Вильяме", 2004г. -752 с.
8. Aripov M.M, Imomov T, Irmuhamedov Z.M va boshqalar. Informatika va axborot texnologiyalari: -Toshkent: O’zbekiston, 1-qism. 2002y, 2-qism. 2003y, -168b.
9. Aripov M.M. Informatika va hisoblash texnikasi asoslari: -Tashkent: O’zbekiston,2001y,-180b.

Aripov M.M, Imomov T, Irmuhamedov Z.M va boshqalar. Informatika. Axborot texnologiyalari: -Toshkent: O’zbekiston, 1-qism. 2002y, -188 b.

MUNDARIJA: