O‘ZBEKISTON RESPUBLIKASI

OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

GULISTON DAVLAT UNIVERSITETI

Axborot texnologiyalari kafedrasi

DASTURLASH TEXNOLOGIYALARI

fanidan o‘quv – uslubiy ko‘rsatma

GULISTON-2024

O‘quv– uslubiy ko‘rsatma Guliston davlat universiteti Kengashining 2024 yil “__” _____dagi ___-sonli bayonnomasi bilan tasdiqlangan “Dasturlash texnologiyalari” fanining ishchi o‘quv dasturi asosida tayyorlangan.

Tuzuvchilar : Saidov J.D. - GulDU “Axborot texnologiyalari” kafedrasi katta o‘qituvchisi p.f.b.f.d. (PhD).

Taqrizchi: Z.Negmatullayev. - GulDU “Axborot texnologiyalari” kafedrasi dotsenti, t.f.b.f.d. (PhD).

O‘quv– uslubiy ko‘rsatma Guliston davlat universiteti O‘quv–uslubiy Kengashi tomonidan («___»_____2024 y. dagi, № __ –sonli bayonnoma) nashrga tavsiya etilgan.

MUNDARIJA

So’zboshi

“Dasturlash texnologiyalari” faninig bosh maqsadi talabalarga qo’yilgan ko’p tarmoqli masalalarni yechadigan kompyuter dasturlarini tuzishni o’rgatishdir. Shu maqsadda dasturlash tili asosida berilganlar bazasi bilan ishlovchi, web sayt ko’rinishidagi axborot tizimlarini yaratish o’rgatiladi.

“Dasturlash texnologiyalari” fani umumkasbiy fanlar blogiga kiritilgan kurs xisoblanib, dasturlash asoslari fani o’tilgandan keyin 2-kurs va 3-kursda o’tilishi maqsadga muvofiq. “Dasturlash texnologiyalari” fani yo’nalishning o’quv rejasidagi “Dasturlash asosladi”, “Algoritmlar nazaryasi”, “Analitik giometriya va chiziqli algebra”, “Diskret matematika va matematik mantiq” fanlari bilan uzviy bog’liq. Fan mazmuni yo’nalishning o’quv rejasidagi “Berilganlar bazasini boshqarish tizimlari”, “Matematik tizimlar” fanlarini o’zlashtirishda tayanch xisoblanadi. Maskud dasturga ko’ra ushbu fan doirasida ko’plab amaliy masalalar o’rganiladi, bu mazkur fanni chuqur o’rgangan xar bir bakalavrga olgan bilim va ko’nikmalarni ishlab-chiqarishda, ilmiy-tadqiqot ishlarida, shuningdek, ta’lim tizimida samaraliy foydalanish imkonini beradi.

1-mavzu: C# tilida umumlashtirish.

Reja:

1. Birinchi yuqori darajali dasturlash tillari.
2. Protseduraviy dasturlash tillari tarixi.
3. Sun'iy intelekt tillari.
4. Zamonaviy obyektga yo'naltirilgan va vizual dasturlash tillari.

Tayanch iboralar: visual, sun'iy, daraja, til, ob'ektga yonaltirilgan, prosedura, C#, C++.

Dasturlash tillari rivojining birinchi qadamlari.

Birinchi EHM lar uchun dasturlani dasturchilar mashina kodi tilida yozganlar. Bu juda qiyin va uzoq vaqt talab etadigan jarayon edi. Dastur tuzishni boshlash va ishlatib ko'rish orasida ancha vaqt o'tar edi. Bunday muammolarni yechish faqatgina dasturlash jarayonini rivojlantirish, optimizatsiya qilish orqaligina bajarilishi mumkin edi.

Dasturchilar mehnatini iqtisod qiluvchi bunday "jixoz" o'rnini qism dasturlari egalladi. 1944 yil avgustida releli "Mark-I" mashinasi uchun Greys Xopper (dasturchi ayol, AQSH ning dengiz ofitseri) boshchiligida sin x ni hisoblovchi qism dasturi yozildi.

Greys Xopperdan boshqalar ham bu ishda ortta qolmadilar. 1949 yilda Jon Mouchli (ENIAK EHM si ixtirochilaridan biri) yuqori darajali dasturlash tillarining dastlabkilariga asos bo'lgan Short Code sistemasini ishlab chiqdi. 1951 yilda Greys Xopper birinchi bo'lib A-O kompilyatorini yaratdi.

1. Birinchi yuqori darajali dasturlash tillari.

O'tgan asrning 50-yillarida Greys Xopper boshchiligida yangi dasturlash tili va kompilyatori V-O ni ishlab chiqishga kirishildi. Yangi til dasturlashni ingliz tiliga yaqin tilda yuajarish imkonini berdi. 30 ga yaqin inglizcha so'zlar tanlandi.

1958 yilda V-O sistemasi Flow-Matic nomini va tijoriy ma'lumot-larni qayta ishlashga yo'naltirildi. 1959 yilda COBOL (Common Business Oriented Language – umumiy tijoratga yo'naltirilgan til) tili io'lab chiqildi. Bu til mashinadan mustaqillikka ega bo'lgan yuqori darajali biznesga yo'naltirilgan dasturlash tilidir. Mashinadan mustaqillikka ega bo'lgan dasturlash tillarida yozilgan dasturlar istalgan turdagi EHM da maxsus kompilyatorlar vositasida bajarilaveradi. COBOL tilini yaratishda ham Greys Xopper maslahatchi bo'lgan.

1954 yilda FORTRAN (FORmula TRANslation) tili yaratilayotgani haqidagi xabar chop etildi. Bu dastur IBM kompaniyasining Nyu Yorkdagi shtab kvartirasida yaratildi. Uni a tuzuvchilardan biri Jon Bekus bo'ldi. U BNF(NFB - normalnaya forma Bekusa – Bekusning normal formasi) muallifi bo'lib, bu forma ko'plab dasturlash tillarining sintaksisini izohlashda qo'llaniladi.

Bu vaqtda Yevropa davlatlarida mashhur til ALGOL edi. Xuddi Fortran kabi u ham matematik topshiriqlarga yo'naltirilgan edi. Unda o'sha davrning ilg'or texnologiyasi – tarkibli dasturlash amalda qo'llangan.

Ko'plab dasturlash tillari o'tgan asrning 60-70-yillarida paydo bo'ldi. Uzoq vaqt yashagan tillar sirasiga BASIC tilini kiritish mumkin. Bu dasturlash tili 1964 yilda Jon Kemeni va Tomas Kurs boshchiligida Dartmut universitetida ishlab chiqildi. Mualliflarining fikriga ko'ra bu til sodda, o'rganishga oson va murakkab bo'lmagan hisoblashlarni bajarishga mo'ljallangan. BASIC ko'proq mikro EHM va shaxsiy kompyuterlarda keng tarqaldi. Dastlab bu til tarkib(struktura)li bo'lgani uchun sifatli dasturlashni o'rganishga qiyin bo'ldi. 1985 yilda uning True BASIC ishlab chiqildi. Bu dasturni tuzganlarni fikriga ko'ra bu til PASCAL ko'ra mukammalroqdir. 1991 yilda Visual BASIC ning birinchi versiyasi paydo bo'ldi.

2. Protseduraviy dasturlash tillari tarixi.

Dasturlash tillari tarixida e’tiborga sazovor voqea 1971 yilda PASCAL tilining yaratilishi bo’ldi. Uning muallifi Shvetsariyalik professor Niklaus Virtdir. Virt bu tilni fransuz fizigi va matematigi Blez Paskal sharafiga qo’ydi (Blez Paskal 1642 yili hisoblash mexanizmini ixtiro qilgan). Dachtlab PASCAL o’rganish tili sifatida tuzilgan. Bu tilda dasturlashning yorqin tomonlari ochib berilgan. Amaliyotda keng qo’llanilishi shaxsiy kompbyuterlarda Turbo PASCAL versiyasidan boshlangan.

C (“Si”) dasturlash tili operatsion tizimlarni ishlab chiqish uchun yaratilgan. U UNIX operatsion tizimi bilan bir vaqtda yaratilgan. Ushbu UNIX operatsion tizimi va dasturlash tilining mualliflari amerikalik dasturchilar Dennis Richi va Kennet Tompsonlardir. Dastlab Kennet Tompson UNIX operatsion tizimini FORTRAN tilida yozgan. Keyinchalik S tili ishlab chiqilgandan so’ng, 1973 yilda operatsion tizimning yadrosi yordamchi dasturlar(utilita programmalar)i bilan C tilida qayta yozildi. Bu yuqori darajali tarkibli (strukturniy) dasturlash tilidir. Bugungi kunda bu til nafaqat operatsion tizimlar, balki translyatorlar, tizimli va amaliy dasturlar ishlab chiqishda qo’llaniladi.

3. Sun’iy intelekt tillari.

O’tgan asrning 90-yillarida “Sun’iy intelekt” nomli beshinchi avlod kompyuterlari ishlab chiqarilishi rejalashtirilgandi. Bu ishni asosiy dasturlash tillarida amalga oshirish amri mahol edi, shu sababli loyihada sun’iy intelekt tillari sifatida LISP va PROLOG tillari tanlandi.

LISP dasturlash tili (1956-1959 yillar) asoschisi Jon Makkarti bo’lib, u sun’iy intelektning otasi hisoblanadi. Aynan u birinchi bo’lib “sun’iy intelekt” atamasini ishlatgan. LISP tilida asosiy element rekursiv ajratilgan funksiyalar tushunish bo’lgan. Istalgan algoritm bir nechta rekursiv bilan funksiyalar to’plami vositasida izohlanishi isbotlangan. Ushbu tilning asosiy g’oyalari keyinroq Seymur Peypert boshchiligida Masachusets texnologiyalar institutida 70-yillarda bolalar uchun ishlab chiqilgan LOGO tilida qo’llanildi.

PROLOG tili ham 1972 yilda Fransiyada sun’iy intelekt muammolarini yechish uchun ishlayu chiqildi. PROLOG tili har xil fikrlarni formal ko’rnishda tavsiflash, mantiqni muxokama qilish, kompyuterni berilgan savollarga javob berdirishga imkoniyatli hisoyulanadi.

4. Zamonaviy obyektga yo’naltirilgan va vizual dasturlash tillari.

So’ngi yillarda kompyuterning dasturiy ta’minoti rivojlanishi asosiy yo’nalishlaridan biri bu obyektga yo’naltirilgan dasturlash sohasi bo’ldi. Obektga yo’naltirilgan operatsion tizimlar (Masalan, Windows), amaliy dasturlar va obyektga yo’naltirilgan dasturlash (OYD) tizimlari ham ammviylashdi.

Birinchi OYD elementi Simula-67 (1967 y., Norvegiya) tili bo’ldi. Turbo PASCAL da 5,5 versiyasidan boshlab OYD vositalari paydo bo’ldi. Turbo PASCAL ning rivoji yakuni yakuni sifatida BORLAND filmasi tomonidan DELPHI dasturlash tizimi yaratilishi bo’ldi. Ushbu sistema yordamida tez va oson murakkab bo’lgan grafik interfeysni dasturlash imkoniyati mavjud. 1991 yilda Visual BASIC ning I versiyasidan boshlab bu til to’laligicha obyektga yo’naltirildi (1997 yil).

1985 yilda Bell Labs (AQSH) layuoratoriyasi C++ dasturlash tili yaratilganligini xabarini berdi. Bugungi kunda bu til OYD tillari orasida mashhurdir. Bu til yordamida istalgan mashina uchun – shaxsiydan to superkompyuterlargacha dasturlar yozish mumkin. Bu tilning asoschisi Born Straustrupdir.

OYD tillaridan yana biri 1995 yilda Jeyms Gosling boshchiligida Sun Microsystems kompaniyasida yaratilgan JAVA tilidir. Uni ishlab chiqishda maxsus o’rganish talab qilmaydigan, sodda tilni maqsad qilingan.

JAVA tili maksimal darajada C++ tiliga o’xshash bo’lishi uchun yaratilgan yaratilgan JAVA Internet uchun dasturlar tayyorlashning ideal vositasidir. So’ngi yillarda Microsoft kompaniyasi tomonidan C++davomchisi sifatida C# (Ci sharp) tili yaratildi.

Consol rejimi.

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 tili alfaviti.

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.

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 *Delegate*	*if* *implicit*	*params* *private*	*this* *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.
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*10^308	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!

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.

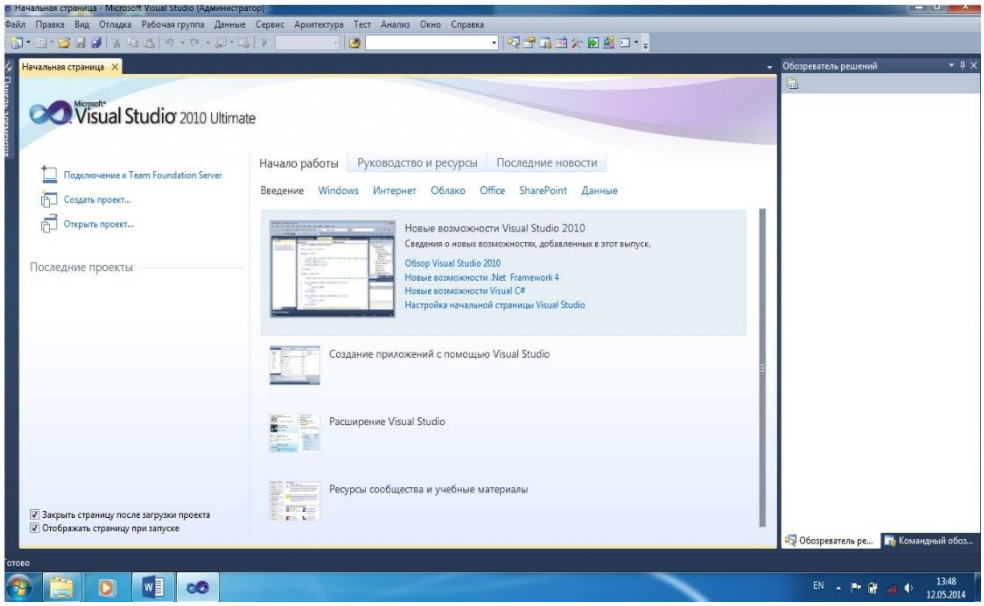
Masalan, 2 o'lchovli massiv e'lon qilish uchun:
Int [,] k; deb e'lon qilish yetarli.

2 darajali massiv uchun *Int [] [] k;* deb e'lon qilish yetarli.

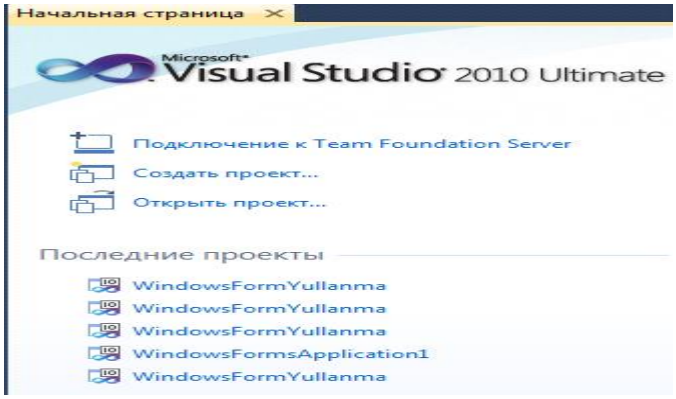
1. Formalar yaratish.

Microsoft Visual Studio haqida

Microsoft korporatsiyasi o'zining oynalar bilan ishlovchi va dunyoda eng ko'p tarqalgan va juda ham ko'p foydalanuvchilar tomonidan ma'qullangan operatsion tizimi Windows uchun ishlab chiqqan Visual Studio (VS) muhiti ham faqatgina oynalar orqaligina ishlaydi. Unda mavjud barcha dastrulash tillari oynalarga bo'lingan va shu oynalar yordamida bir biriga bog'langan. VS yuklash uchun **пуск->program->Microsoft Visual Studio->Microsoft Visual Studio** ketma ketligini bajarish kerak.VS yuklanganda odatda quyidagi ko'rinishda bo'ladi. Birinchi navbatda,VS yuklanganda Start Page muloqot oynasi ochiladi.



MS Visual Studio muloqot oynasi.



Bosh sahifa oynasi.

1. Recent Projects – avval ishlangan loyihalardan 6 tasi ekranda ko'rsatilgan bo'ladi, ularni ustiga sichqonchaning chap tugmasini ikki marotaba bosish orqali ishga tushirish mumkin, shu bo'limning pastki qismida Open: Projects va Create: Projects bandlari bo'lib, open bandi orqali xotirada mavjud bo'lgan loyihani qayta ochish imkonini beradi, create bandi orqali esa yangi loyiha yaratish mumkin.

2. Getting started – loyiha yaratish va shu muhit haqida ma'lumot olish imkonini beradi;

3. Visual Studio Headlines – bu bo'limda sizning shu muhitda ishlayotganingiz uchun uning mualliflari tomonidan bildirilgan minnatdorligi ko'rsatilgan;

4. Visual Studio Developer News – bu asosiy qismda VS ustida qanday o'zgarishlar olib borilayotgani yoki olib borilgani haqida ma'lumot beradi.

Endi biz yangi foydalanuvchi bo'lganimiz uchun bu muloqot oynasini yopib, yangi oyna ochamiz. Buning uchun biz menyular paneli bilan tanishish kerak.

Menyular paneli bilan ishlash.

1. File

- **new** – yangi loyiha yaratish
- **open** – yaratilgan loyihani ochish
- **close** – joriy loyihani yopish
- **close Solution** – barcha loyihalarni yopish
- **save Selected Items** – belgilangan barcha loyiha qismlarini saqlash
- **save Selected Items As** - belgilangan qismlarni barchasini xohlagan tartibda saqlash
- **save All** – barcha dasturlarni saqlash
- **export Template** – dasturni arxivlash
- **page Setup** – sahifa sozlamasini o'zgartirish
- **print** – chop etish
- **recent files** – fayllarni qayta ochish
- **recent Projects** – loyihalarni qayta ochish
- **Visual Studio** dan chiqish

2. Edit

- **undo** – bajarilgan amallarni bir marotaba bekor qilish;
- **redo** – oxirgi bekor qilingan amalni qaytarish;
- **undo Last Global Action** – barcha qilingan amallarni bekor qilish;
- **redo Last Global Action** – bekor qilingan barcha amallarni qaytarish;
- **cut** – belgilangan qismni yoki komponentni qirqib, xotirada saqlab turish;
- **copy** – belgilangan qism yoki komponentaning nusxasini xotirada saqlab turish;

- paste – xotiraga qirqib yoki nusxalab olingan qism yoki komponentni belgilangan yoki ko’rsatilgan joyga qo’yish

- select All – barcha qismlarni belgilab olish;

- Find and Replace – ma’lum qism yoki komponentani butun muhit bo’ylab qidiradi va o’zgartiradi

3. View

- open – Joriy dasturni ochib beradi;

- open With – Joriy dasturni xohlagan dastur yordamida ishlov berish imkoniyatini yaratadi;

- server Explorer – Server oynasi bo’lib, unda asosiy komponentalar joylashadi;

- solution Explorer – Ob’yektlar oynasi;

- class View – Loyiha elementlarini ko’rsatib beradi, uning yordamida yangi yoki qo’shimcha ob’yektlarni qo’shish, olib tashlash, ko’rinishini, tartibibni o’zgartirish mumkin;

- code Definition Window – Ob’yekt kodlarini ko’rsatib beruvchi oyna;

- object Browser – Visual Studiodagi barcha ob’yektlar xususiyatlarini va ularni qo’llash usullari beriladigan oyna;

- Error List – xatoliklar oynasi bo’lib, dastur kompiliyatsiya qilinishidan oldin va keyin undagi xatoliklarni ko’rsatib beradi va unda namoyon bo’lgan har bir xatolikni ustida ikki marotaba sichqoncha chiqillatilsa xatolik bo’lgan joyga kursorni eltib qo’yadi;

- output –Dastur o’z ishlashi davomida qanday fayllarga murojaat qilishi, qanday fayllarni hosil qilishi, qaysi cs yordamida hosil bo’lishi va qanday qiymat qaytarishini ko’rsatuvchi oyna;

- properties Window – xususiyatlar oynasi bo’lib, qaysi ob’yekt faol bo’lsa, shu ob’yektning xususiyatlarini;

- toolbox – komponentalar joylashgan oyna bo’lib, unda VS da ishlatish mumkin bo’lgan barcha ob’yektlar joylashtirilgan;

- Find Result – natijani topish oynasi bo’lib, uning yordamida dastur ishlatilganda berilgan qiymatlarda qanday natija olinishini ko’rsatib beradi;

- toolbars – panellar qo’yish yoki olib tashlash bandi bo’lib, unda barcha turdagi panellarni izlash mumkin;

- full Screen – VS butun ekran hajmida kattalashadi;

4. Refactor

- rename – dasturni qayta nomlash;

- extract Method – metodni qayd etish;

- extract Interface – Interfeysni qayd etish;

- remove Parameters–parametrlarni qayta ko’chirish;

- reorder Parameters – parametrlarni qaytarish.

5. Project

- add Windows Form –Windows formasini qo’shadi;

- add User Control – foydalanuvchi boshqarishining boshqa usullarini qo’shadi;

- add Component – Yangi component qo’shadi (foydala-nuvchi tomonidan tayyorlangan bo’lishi ham mumkin);

- add Class – yangi sinf(ma’lumotlar yoki formalar sinfi)ni qo’shadi;

- add New Item – yangi elementini qo’shadi (ob’yektning);

- add Exiting Item – yangi chiquvchi elementini qo’shadi;

- show All Files – barcha fayllarni namoyish etadi;

- console - Properties – Consolning xususiyatlari;

6. Debug

- **windows** – Breakpoints,Output, Immedite Windows bandlarni o’z ichiga olgan bo’lib, bu oynalar yordamida ishlash imkonini beradi;

- **start Debugging** – Joriy loyihani komplyatsiya qilib, ishga tushiradi;

- **start Without Debugging** – Dasturni komplyatsiya qilib, bir necha komponentalarni birgalikda ishga tushiradi;

- **new Breakpoints** - Yangi to’xtash nuqtalarini hosil qiladi.Bunda hosil bo’lgan nuqtalarni Shift+F5 tugmasi orqali bekor qilish mumkin;

- **Delete All Breakpoints** – Barcha qo’yilgan to’xtash nuqtalarini bekor qilish uchun ishlatiladi.

7. Data

- **Show Data Sources** – Shu loyiha bilan bog’langan barcha ma’lumotlar bazalarini ko’rsatib beradi;

- **Add New Data Sources** – Yangi ma’lumotlar bazasini qo’shish uchun ishlatiladi;

- **Schema Compare** – Shu loyihada mavjud bo’lgan ma’lumotlar bazalari orasidagi barcha bog’liqliklarni ko’rsatib beradi;

- **Preview Data** – Shu loyihadagi barcha ma’lumotlar bazalarini qidirish uchun ishlatiladi;

- **Refactor** – Refaktor bandi bilan bog’liqlik o’rnatadi.

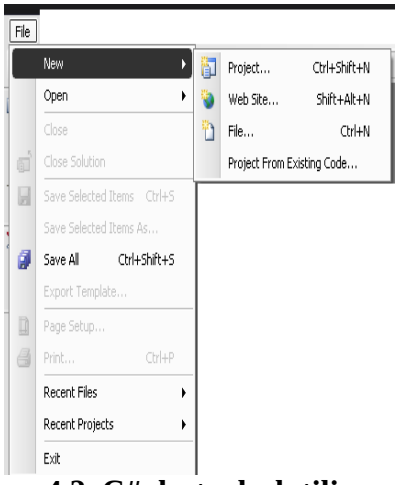
8. Format

- **Align** – Formaning joylashish koordinatlarini o’rnatish imkonini beradi. Bunda uning chap,o’ng,o’rta, past va yuqoridan chegaralash mumkin;

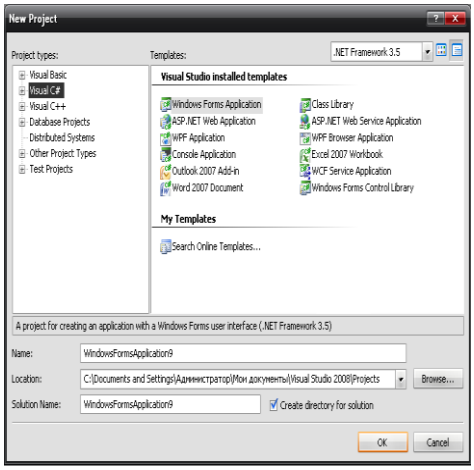
- **Make Same Size** – Formaning o’lchamini berish mumkin.Unda balandligi, eni qalinligi va chegara qismini berish mumkin;

- **New** - Yangi loyiha yaratish VS ning eng asosiy amallaridan biri. Bu menyuda yangi loyiha, web-sayt, file (o’zida kodlarni saqlab turuvchi *.cs fayl), oddiy fayl yaratish mumkin. VS da bir necha tillar jamlangan. Biz shu tillar orasidan hozirda dunyoning eng kuchli dasturlash tillaridan biri deb tan olinayotgan C# dasturlash tilidir. C# tili o’zining interfeysi va sintaksisi uning yuqori sathli dasturlash tillariga kirishiga olib keladi. VS dagi C# tili ham huddi shu vazifani bajarib beradi. Uning Console Application qismida qora oynali muloqot oynasi yaratiladi va shu qora oyna dastur ishga tushganda paydo bo’ladi.

C# tilining sintaksisi uning C# va C++ tillaridan “nusxa” ekanligini bildiradi. Andres Xijisberg (C# tilining asoschisi) bu tilni yaratayotganda o’z oldiga quyidagi vazifalarni asosiy maqsadi sifatida ishlata boshladi.



VS dagi loyiha oynasi.



C# tilida yaratish

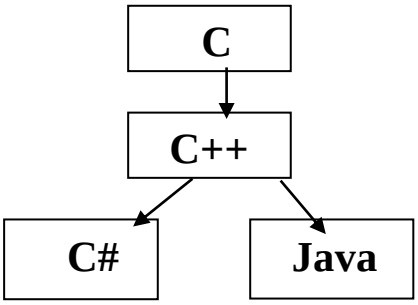
4.2. C# dasturlash tili.

C# dasturi 90-yillarning oxirida ishlab chiqilib **Microsoft .NET** ning bir qismiga aylandi. Al’fa versiya sifatida 2000 yildan boshlab ishlatila boshladi .C # bosh arxitektori butun dunyo

dasturchilari ichida birinchilar qatorida turgan va butun dunyo tomonidan tan olingan dasturlash tilidir. Asoschisi **Anders Hejlsberg** bo'ldi. Uning 1980 yillarda chiqarilgan Turbo Paskal dasturi orqali ham tanishimiz mumkin.

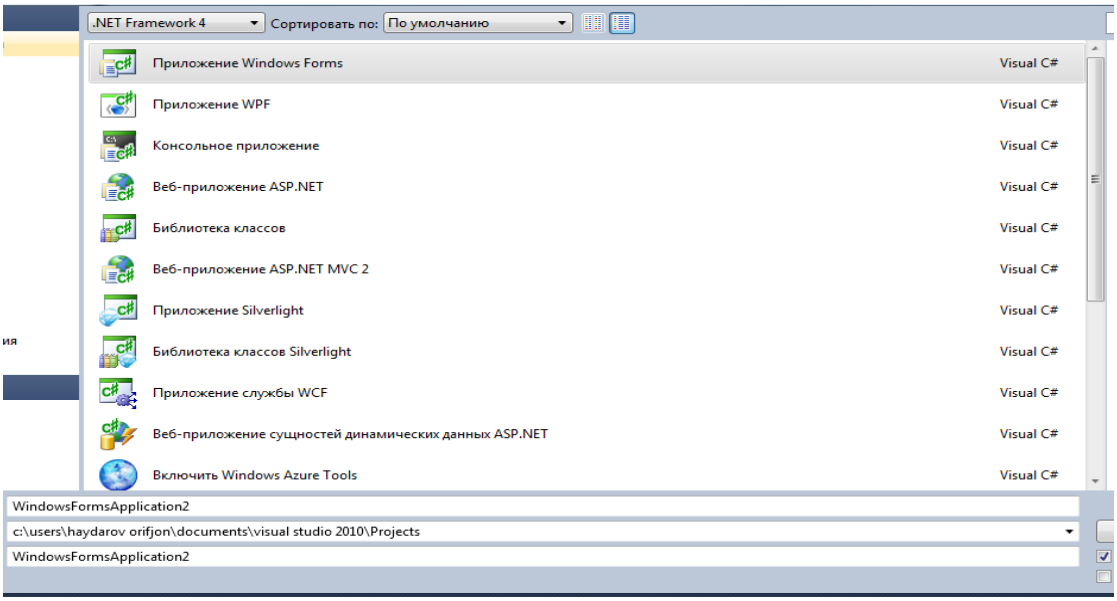
C# bevosita C, C++ va Java bilan bog'liq. Chunki bu uchta til dasturlash olamida eng mashhur tillardir. Bundan tashqari profisanal dasturchilar C va C++ ni va juda ko'pchilik Java tilida ish yuritadi.

C# ning kelib chiqish genealogik daraxti quyidagicha bo'ladi.



C# ning kelib chiqishi.

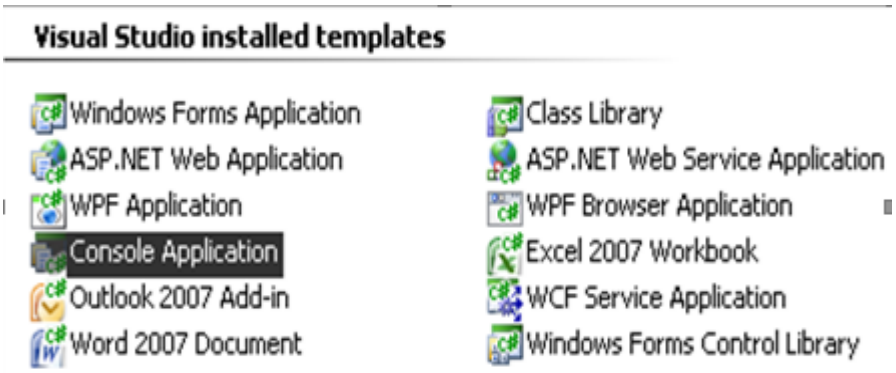
Chizmada C va C++ tillari C# ning asosini tashkil qiladi. Lekin C# va Java o'zaro o'zgacha ravishda bog'langan. Ularning kelib chiqishi C va C++ bo'lsada o'zaro bir biridan farq qiladi . C# tili ham obe'ktga mo'ljallangan tillar sirasiga kiradi. VS 2010 da C# tili yordamida 17 xil loyiha yaratish mumkin.



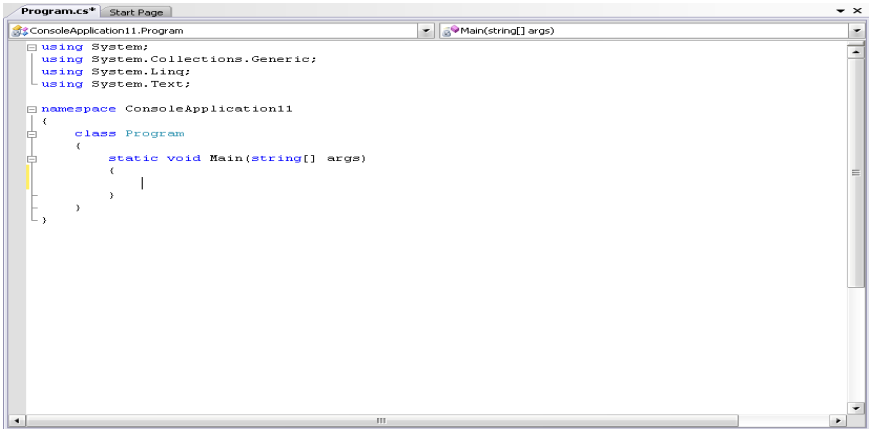
MS VS 2010 da C# tili yordamida 17 xil loyiha yaratish jadvali.

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 boshlang'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. VS dagi C# tili ham huddi shu vazifani bajarib beradi. Uning Console Application qismida qora oynali muloqot oynasi yaratiladi va shu qora oyna dastur ishga tushganda paydo bo'ladi. Uning ko'rinishi avtomatik tarzda VS tomonidan shakllantirilgan bo'lib u dasturchi tomonidan o'zgartirilishi mumkin emas. Consolda ishlatiladigan barcha komponentalar, ularning kodlari, parametrlar boshqa loyihalarda ham

qo'llanilishi mumkin. Birinchi navbatda **New->Project->Visual C#->Console Application** buyruqlar ketma ketligi bajarishi kerak va quyidagicha bo'ladi.



Buyruqlar jadvali.

Muloqot  oynasi.

Dastur tuzilishi: C# da dastur birta yoki bir necha fayllardan iborat bo'ladi. Har bir fayl o'zida bir yoki bir necha nomlarni saqlab turishi mumkin.Har bir nom esa o'zida qo'yilgan nom yoki tipni, qaysiki sinflar, struktura, interfeys, hisoblash va delegate – funksional tiplardir.**C#** da yangi loyiha yaratish paytida, **VS** muhitidagi 10 ta loyiha tiplaridan biri tanlanadi.Bularga Windows Application, Class Library, ASP.NET Application va ASP.NET Web Service misol bo'ladi. Birortasi tanlansa avtomatik tarzda **C#** va C++ ning shu loyiha tipiga mos qolipi hosil bo'ladi.

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

2. Shunday ob'yektga yo'naltirilgan dasturlash tilini yaratish kerakki, unda hamma narsa ob'yekt 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 ob'yektga 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.

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

1. Operatorlar.

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 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.

Nazorat savollari:

1. Ifoda nima?
2. Operator nima?
3. Shartli operatorlar.
4. Tanlash operatorlari.

2-Mavzu: Umumlashgan metod yaratish**.**

Reja:

1. Metodlar va ularni yaratish.
2. Umumlashgan metod

Umumiy turlardan tashqari .NET freymwork ham umumiy metod yaratishni, shuningdek generiklarni ham qo'llab-quvvatlaydi. Ushbu hodisaning o'ziga xos xususiyatlarini tushunish uchun avval umumiy turlar paydo bo'lishidan oldin paydo bo'lishi mumkin bo'lgan muammoni ko'rib chiqamiz. Keling, bir misolni ko'rib chiqaylik. Aytaylik, biz bank hisob raqamini ko'rsatish uchun sinfni aniqladik. Masalan, shunday bo'lishi mumkin:

```
class Account
{
    public int Id { get; set; }
    public int Sum { get; set; }
}
```

Hisob qaydnomasi klassi ikkita xususiyatni belgilaydi: Id - noyob identifikator va Sum - hisobdagi summa.

Bu erda identifikator raqamli qiymat sifatida o'rnatiladi, ya'ni bank hisobvaraqlari 1, 2, 3, 4 va boshqalarga ega bo'ladi. Shu bilan birga, identifikator uchun satr qiymatlarini ishlatish ham keng tarqalgan. Ikkala raqamli va satrli qiymatlarning ijobiy va salbiy tomonlari bor. Va sinfni yozish paytida biz identifikatorni saqlash uchun nimani tanlashni yaxshiroq bilmasligimiz mumkin - satr yoki raqam. Yoki, ehtimol ushbu sinf ushbu masala bo'yicha o'z fikriga ega bo'lgan boshqa ishlab chiquvchilar tomonidan ishlatilishi mumkin.

Va birinchi qarashda, ushbu vaziyatdan chiqish uchun Id xususiyatini tip ob'ektining xususiyati sifatida aniqlashimiz mumkin. Ob'ekt turi barcha turlar meros bo'lib o'tadigan universal turdagi ekan, biz ikkala satrni va raqamlarni ushbu turdagi xususiyatlarda saqlashimiz mumkin:

```
class Account
{
    public object Id { get; set; }
    public int Sum { get; set; }
}
```

Keyinchalik ushbu sinf dasturda bank hisobvaraqlarini yaratish uchun ishlatilishi mumkin:

```
Account account1 = new Account { Sum = 5000 };
Account account2 = new Account { Sum = 4000 };
account1.Id = 2;
account2.Id = "4356";
int id1 = (int)account1.Id;
string id2 = (string)account2.Id;
Console.WriteLine(id1);
Console.WriteLine(id2);
```

Har bir narsa juda yaxshi ishlaydi, ammo bu echim juda maqbul emas. Haqiqat shundaki, bu holda biz boks va boks qutisi kabi hodisalarga duch kelamiz.

Shunday qilib, Id xususiyatiga int qiymati berilganida, bu qiymat Object turiga to'planadi:

```
account1.Id = 2;  // int qiymatiga elementni ob'ekt turiga kiritish
```

Ma'lumotlarni int turidagi o'zgaruvchiga qaytarish uchun quyidagilarni ochishingiz kerak:

```
int id1 = (int)account1.Id;  // Int turidagi toplamni olib
tashlash
```

Boks qiymat turidagi ob'ektni (int kabi) ob'ekt turiga o'zgartirishni o'z ichiga oladi. Paketlashda umumiy CLR tili qiymatni System.Object tipidagi ob'ektga o'raladi va uni boshqariladigan uyumda saqlaydi. Boshqa tomondan qutidan chiqarish, turdagi ob'ekt ob'ektini qiymat turiga o'tkazishni o'z ichiga oladi. Qadoqlash va ochish ish faoliyatini pasaytiradi, chunki tizim kerakli o'zgarishlarni amalga oshirishi kerak.

Bundan tashqari, yana bir muammo bor - turlarning xavfsizligi muammosi. Shunday qilib, agar biz shunday yozsak, ish vaqtida xatolikka yo'l qo'yamiz:

```
Account account2 = new Account { Sum = 4000 };
account2.Id = "4356";
int id2 = (int)account2.Id;
```

Biz aniq qaysi ob'ekt Idni ifodalayotganini bilmasligimiz mumkin va bu holda raqamni olishga harakat qilsak, InvalidCastException bilan uchrashamiz.
Ushbu muammolar umumiy turlarni yo'q qilishga qaratilgan edi (ko'pincha ularni umumiy turlar deb ham atashadi). Umumiy turlar sizga ishlatilishi kerak bo'lgan ma'lum bir turni ko'rsatishga imkon beradi. Shuning uchun, Hisob sinfini umumiy deb belgilaylik:

```
class Account<T>
{
    public T Id { get; set; }
    public int Sum { get; set; }
}
```

<T> hisob qaydnomasi tavsifidagi burchakli qavslar sinfning umumiy ekanligini bildiradi va burchakli qavs ichiga olingan T tipi shu sinf tomonidan qo'llaniladi. T harfini ishlatish shart emas, u har qanday boshqa harf yoki belgilar to'plami bo'lishi mumkin. Va endi biz uning qaysi turi bo'lishini bilmaymiz, har qanday tur bo'lishi mumkin. Shuning uchun burchak qavslaridagi T parametri umumiy parametr deb ham ataladi, chunki uning o'rniga har qanday turni almashtirish mumkin.

Masalan, T parametri o'rniga int ob'ektidan, ya'ni hisob raqamini ifodalaydigan raqamdan foydalanishingiz mumkin. Bu shuningdek mag'lubiyat ob'ekti yoki boshqa har qanday sinf yoki tuzilma bo'lishi mumkin:

```
Account<int> account1 = new Account<int> { Sum = 5000 };
Account<string> account2 = new Account<string> { Sum = 4000 };
account1.Id = 2;         // упаковка не нужна
account2.Id = "4356";
int id1 = account1.Id;  // распаковка не нужна
string id2 = account2.Id;
Console.WriteLine(id1);
Console.WriteLine(id2);
```

Hisob klassi umumiy bo'lganligi sababli, burchak nomidagi qavsdagi tur nomidan keyin o'zgaruvchini belgilashda siz universal T parametr o'rniga ishlatiladigan turni belgilashingiz kerak. Bunday holda, hisob ob'ektlari int va string turlari bilan yoziladi:

```
Account<int> account1 = new Account<int> { Sum = 5000 };
Account<string> account2 = new Account<string> { Sum = 4000 };
```

Shuning uchun birinchi account1 ob'ekti int tipidagi Id xususiyatiga va account2 ob'ekti string turiga ega bo'ladi.

Id xususiyatining qiymatini boshqa turdagi o'zgaruvchiga berishga harakat qilsak, biz kompilyatsiya xatosiga duch kelamiz:

```
Account<string> account2 = new Account<string> { Sum = 4000 };
account2.Id = "4356";
int id1 = account2.Id;// kompilyatsiya xatosi
```

Bu turdagi xavfsizlik muammolarini oldini oladi. Shunday qilib, sinfning umumiy versiyasidan foydalanib, biz bajarilish vaqtini va yuzaga kelishi mumkin bo'lgan xatolarni kamaytiramiz.

Standart qiymatlar

Ba'zan umumiy parametrlarning o'zgaruvchilariga ba'zi boshlang'ich qiymatlarni tayinlash kerak bo'ladi, shu jumladan null. Ammo biz uni to'g'ridan-to'g'ri tayinlay olmaymiz:

```
T id = null;
```

Bunday holda, biz default (T) operatoridan foydalanishimiz kerak. U mos yozuvlar turlariga null va qiymat turlariga 0 belgilaydi:

```
class Account<T>
{
    T id = default(T);
}
```

Umumiy sinfning statik maydonlari

Umumiy sinfni ma'lum bir turga kiritishda uning statik a'zolar to'plami yaratiladi. Masalan, Account klassi quyidagi statik maydonni belgilaydi:

```
class Account<T>
{
    public static T session;

    public T Id { get; set; }
    public int Sum { get; set; }
}
```

Endi biz sinfni ikkita turdagi int va string bilan yozamiz:

```
Account<int> account1 = new Account<int> { Sum = 5000 };
Account<int>.session = 5436;

Account<string> account2 = new Account<string> { Sum = 4000 };
Account<string>.session = "45245";

Console.WriteLine(Account<int>.session);      // 5436
Console.WriteLine(Account<string>.session);   // 45245
```

Natijada, Account <string> va Account <int> uchun sessiya o'zgaruvchisi yaratiladi.

Bir nechta umumiy parametrlardan foydalanish

Generika bir vaqtning o'zida bir nechta umumiy parametrlardan foydalanishi mumkin, bu esa har xil turlarni ifodalashi mumkin:

```
class Transaction<U, V>
{
    public U FromAccount { get; set; }  // pul o'tkazmasi hisobi
    public U ToAccount { get; set; }    // qaysi hisob raqamiga
pul o'tkazish
    public V Code { get; set; }         // operatsion kod
    public int Sum { get; set; }        // pul o'tkazmasi summasi
}
```

transfert miqdori Bu erda Transaction klassi ikkita umumiy parametrdan foydalanadi. Keling, ushbu sinfni qo'llaymiz:

```
Account<int> acc1 = new Account<int> { Id = 1857, Sum = 4500 };
Account<int> acc2 = new Account<int> { Id = 3453, Sum = 5000 };

Transaction<Account<int>, string> transaction1 = new
Transaction<Account<int>, string>
{
    FromAccount = acc1,
    ToAccount = acc2,
    Code = "45478758",
    Sum = 900
};
```

Bu erda Transaction ob'ekti Account <int> va string turlari bilan yoziladi. Ya'ni U universal parametri sifatida Account <int> klassi, V parametri uchun string turi ishlatiladi. Shu bilan birga, siz ko'rib turganingizdek, Transaction yoziladigan sinf o'zi umumiydir.

Umumlashtirilgan usullar

Umumiy sinflardan tashqari, umumiy parametrlarni xuddi shu tarzda ishlatadigan umumiy usullarni ham yaratishingiz mumkin. Masalan; misol uchun:

```
private static void Main(string[] args)
    {
        int x = 7;
        int y = 25;
        Swap<int>(ref x, ref y); // или так Swap(ref x, ref y);
        Console.WriteLine($"x={x}    y={y}");   // x=25   y=7

        string s1 = "hello";
        string s2 = "bye";
        Swap<string>(ref s1, ref s2); // или так Swap(ref s1, ref
s2);
        Console.WriteLine($"s1={s1}    s2={s2}"); // s1=bye
s2=hello

        Console.Read();
    }
    public static void Swap<T> (ref T x, ref T y)
```

```
    {
        T temp = x;
        x = y;
        y = temp;
    }
}
```

Bu parametrlarni mos yozuvlar asosida qabul qiladigan va ularning qiymatlarini o'zgartiradigan umumiy almashtirish usulini belgilaydi. Bunday holda, ushbu parametrlarning qaysi turini anglatishi muhim emas.

Asosiy usulda biz almashtirish usulini chaqiramiz, uni ma'lum bir tip bilan yozamiz va unga bir nechta qiymatlarni beramiz.

Savollar:

1. Metod nima
2. Umumlashgan metod nima
3. Umumlashgan metodlarning avzalliklari.

3-Mavzu: Umumlashgan delegatlar.

Reja:

1. .NET da Delegatlar.
2. Umumlashgan delegatlar.

Usullar singari, delegatlar ham umumiy bo'lishi mumkin. Quyida umumiy delegatni e'lon qilishning umumiy shakli keltirilgan:

Tur parametrlari ro'yxati joylashgan joyiga e'tibor bering. U darhol delegat nomidan kelib chiqadi. Umumiy delegatlarning afzalligi shundan iboratki, ular yozilgan umumiy shaklda aniqlanishi mumkin, keyinchalik ularni har qanday mos keladigan usul bilan moslashtirish mumkin.

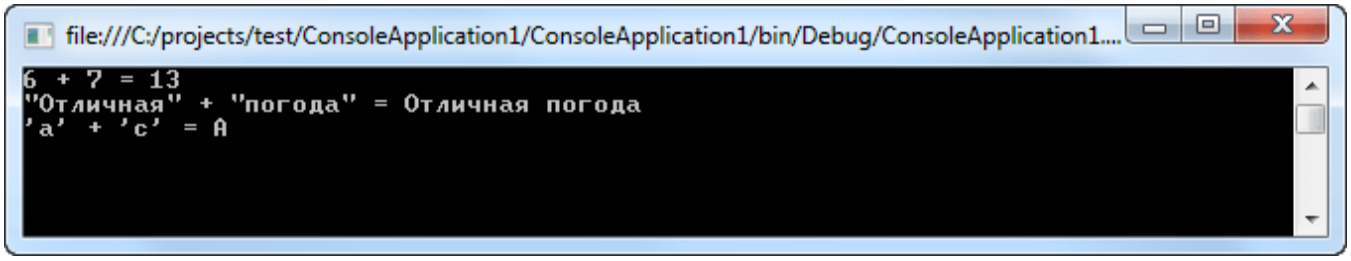
Umumiy delegatlar xavfsiz usulda chaqiriladigan usulni belgilashning yanada moslashuvchan usulini taklif qilishadi. General.dan oldin (.NET 2.0 da) xuddi shu yakuniy natijaga System.Object parametri yordamida erishish mumkin edi:

```
public delegate void MyDelegate(object arg);
```

Bu sizga vakilning maqsadiga biron bir dalillarni yuborishga imkon beradigan bo'lsa-da, u tipdagi xavfsizlikni ta'minlamaydi va boks / qutidan olib tashlash zaruratini bartaraf etmaydi. Keling, umumiy delegatlarni ishlatish misolini ko'rib chiqaylik:

```
using System;

namespace ConsoleApplication1
{
    // Создадим обобщенный делегат
    delegate T MyDel<T> (T obj1, T obj2);

    class MySum
    {
        public static int SumInt(int a, int b)
        {
            return a + b;
```

```
        }

        public static string SumStr(string s1, string s2)
        {
            return s1 + " " + s2;
        }

        public static char SumCh(char a, char b)
        {
            return (char)(a + b);
        }
    }

    class Program
    {
        static void Main()
        {
            // Реализуем несколько методов обобщенного делегата
            MyDel<int> del1 = MySum.SumInt;
            Console.WriteLine("6 + 7 = " + del1(6,7));

            MyDel<string> del2 = MySum.SumStr;
            Console.WriteLine("\"Отличная\" + \"погода\" = " +
del2("Отличная","погода"));

            MyDel<char> del3 = MySum.SumCh;
            Console.WriteLine("'a' + 'c' = " + del3('a','c'));

            Console.ReadLine();
        }
    }
}
```



Deylik, c # delegatini yaratishda biz uni yanada universalroq yoki aniqroq umumlashtirmoqchimiz!

```
public delegate void SomeDelegate<T>(T item);
```

T - bu usul qabul qiladigan parametrning umumiy (har qanday, qo'ng'iroq paytida aniq) turi.

Vakil imzosini aniqlash bilan ishimiz tugadi. Keyingi qadam delegat robotning o'zini namoyish qilish uchun usul (lar) ni yaratishdir. Mana usul:

```
1. public static void Show(string msg)
```

```
{
    Console.WriteLine(msg);
    Console.ReadLine();
}
```

Vakilning imzosi aniqlangan, uslublar yaratilgan, u bilan ishlaydigan vakilga uslubni ko'rsatish qoladi. Buni amalga oshirish uchun delegatimiz ob'ektini yarataylik:

```
SomeDelegate<int> d1 = new SomeDelegate<int>(Show);

d1(5);
```

Va aslida, bu hamma kod:

```
using System;

public delegate void SomeDelegate<T>(T item);
class Program
{
    static void Main(string[] args)
    {
        SomeDelegate<int> d1 =
            new SomeDelegate<int>(Show);
            d1(5);
        }

        public static void Show(int a)
        {
            Console.WriteLine(a + " " + a);
            Console.ReadLine();
        }
    }
```

Tabiiyki, siz bir nechta kirish parametrlari bilan usullarni tayinlashingiz mumkin:

```
using System;

public delegate void SomeDelegate
     <T1, T2, T3>(T1 i, T2 s, T3 b);
class Program
{
    static void Main(string[] args)
    {
        SomeDelegate<int, string, bool> d1 =
                  new SomeDelegate<int, string, bool>(Show);
            d1(5, "TECT", true);
        }

        public static void Show(int i, string s, bool b)
```

```
15.         {
16.             if (b)
17.                 for (int count = 0; count < i; count ++ )
18.                     Console.WriteLine(s + Environment.NewLine);
19.
20.             Console.ReadKey();
21.         }
22.     }
```

Hech qanday qiyinchiliklar mavjud emas, lekin siz shuni tushunishingiz kerakki, umumiy vakillar, boshqa turdagi generiklar singari, kerak bo'lganda siz haqiqatan ham foydalanishingiz kerak, chunki umumlashtirish (universallikni o'qing) ijro etilish tezligida qurbonlik bilan beriladi.

Savollar:

1. Delegatlar nima.
2. Umumlashma nima.
3. Umumlashgan delegatlar qanday hossaga ega.

4-Mavzu: Interfeyslar.

Reja:

1. Muhid va uni boshqarish.
2. Interfeslar.

C # ni o'rganishni boshlaganlar ko'pincha interfeys nima va nima uchun kerakligi haqida savol berishadi.

Birinchidan, qidiruv tizimidagi birinchi havolada nimani topish mumkinligi haqida. Ko'pgina maqolalarda interfeysning ma'nosi sinf nimani o'z ichiga olishi, qanday xususiyatlari va usullari to'g'risida "shartnoma" sifatida izohlanadi.

Interfeys ba'zi bir funktsiyalarni aniqlay oladigan mos yozuvlar turini ifodalaydi - amalga oshirilmasdan usullar va xususiyatlar to'plami. Keyinchalik, ushbu funktsiya ushbu interfeyslardan foydalanadigan sinflar va tuzilmalar tomonidan amalga oshiriladi.

Interfeysni aniqlash

Interfeysni aniqlash uchun interfeys kalit so'zi ishlatiladi. Odatda, C # dagi interfeys nomlari I kapitali bilan boshlanadi, masalan, IComparable, IEnumerable (venger yozuvi deb ataladi), ammo bu shart emas, aksincha dasturlash uslubi.

Interfeys nimani belgilashi mumkin? Umuman olganda, interfeyslar quyidagi ob'ektlarni belgilashi mumkin:

metodlar

Xususiyatlari

Indeksatorlar

hodisalar

Statik maydonlar va doimiylar (C # 8.0 dan boshlangan)

Biroq, interfeyslar statik bo'lmagan o'zgaruvchilarni aniqlay olmaydi. Masalan, ushbu komponentlarning barchasini belgilaydigan eng oddiy interfeys:

```
interface IMovable
{
    // константа
    const int minSpeed = 0;     // минимальная скорость
    // статическая переменная
    static int maxSpeed = 60;   // максимальная скорость
    // метод
    void Move();                // движение
    // свойство
    string Name { get; set; }   // название

    delegate void MoveHandler(string message);  // определение делегата для
события
    // событие
    event MoveHandler MoveEvent;    // событие движения
}
```

Bunday holda, harakatlanuvchi interfeys aniqlanadi, bu ba'zi bir harakatlanuvchi ob'ektni aks ettiradi. Ushbu interfeysda harakatlanuvchi ob'ektning imkoniyatlarini tavsiflovchi turli xil komponentlar mavjud. Ya'ni, interfeys harakatlanuvchi ob'ektga ega bo'lishi kerak bo'lgan ba'zi funktsiyalarni tavsiflaydi.

Interfeysning usullari va xususiyatlari bajarilmasligi mumkin, bunda ular mavhum sinflarning mavhum usullari va xususiyatlariga yaqinlashadi. Bunday holda, interfeys ba'zi bir harakatni ifodalaydigan Move usulini belgilaydi. U amalga oshirilmaydi, parametrlarni qabul qilmaydi va hech narsa qaytarmaydi.

Xuddi shu narsa Name xususiyatiga ham tegishli. Bir qarashda u avtomatik xususiyatga o'xshaydi. Ammo aslida, bu interfeysdagi avtomatik ta'rif emas, balki amalga oshiriladigan xususiyatning ta'rifi.

Interfeysni e'lon qilishda yana bir nuqta: agar uning a'zolari - usullari va xususiyatlarida kirish modifikatorlari bo'lmasa, lekin aslida sukut bo'yicha kirish ochiqdir, chunki interfeysning maqsadi uni sinf tomonidan amalga oshirish uchun funksionallikni aniqlashdir. . Bu sinflar va tuzilmalarda sukut bo'yicha xususiy bo'lgan doimiy va statik o'zgaruvchilarga ham tegishli. Interfeyslarda ular sukut bo'yicha umumiy modifikatorga ega. Masalan, IMovable interfeysining minSpeed doimiy va maxSpeed o'zgaruvchisiga murojaat qilishimiz mumkin:

```
static void Main(string[] args)
{
    Console.WriteLine(IMovable.maxSpeed);
    Console.WriteLine(IMovable.minSpeed);
}
```

Bundan tashqari, C # 8.0 versiyasidan boshlab, biz interfeys komponentlari uchun kirish modifikatorlarini aniq belgilashimiz mumkin:

```
interface IMovable
{
    public const int minSpeed = 0;      // минимальная скорость
    private static int maxSpeed = 60;   // максимальная скорость
    public void Move();
    protected internal string Name { get; set; }    // название
    public delegate void MoveHandler(string message);  // определение делегата
для события
    public event MoveHandler MoveEvent;    // событие движения
}
```

C # 8.0 dan boshlab interfeyslar standart usul va xususiyatlarni amalga oshirishni qo'llab-quvvatlaydi. Bu shuni anglatadiki, biz oddiy sinflar va tuzilmalar singari amalga oshiriladigan interfeyslarda to'liq usullar va xususiyatlarni aniqlay olamiz. Masalan, Move usulining sukut bo'yicha bajarilishini aniqlaylik:

```
interface IMovable
{
    // реализация метода по умолчанию
    void Move()
    {
        Console.WriteLine("Walking");
    }
}
```

Standart xususiyatlarni interfeyslarda amalga oshirish bilan vaziyat biroz murakkablashadi, chunki biz interfeyslarda statik bo'lmagan o'zgaruvchilarni aniqlay olmaymiz, shuning uchun biz interfeys xususiyatlaridagi ob'ekt holatini boshqarolmaymiz. Shu bilan birga, biz xususiyatlar uchun standart dasturni ham belgilashimiz mumkin:

```
interface IMovable
{
    void Move()
    {
        Console.WriteLine("Walking");
    }
    // реализация свойства по умолчанию
    // свойство только для чтения
    int MaxSpeed { get { return 0; } }
}
```

Shuni ta'kidlash kerakki, agar interfeysda xususiy usullar va xususiyatlar mavjud bo'lsa (ya'ni xususiy modifikator bilan), unda ular standart dasturga ega bo'lishi kerak. Xuddi shu narsa har qanday statik usul va xususiyatlarga nisbatan qo'llaniladi (albatta xususiy emas):

```
interface IMovable
{
    public const int minSpeed = 0;      // минимальная скорость
    private static int maxSpeed = 60;   // максимальная скорость
```

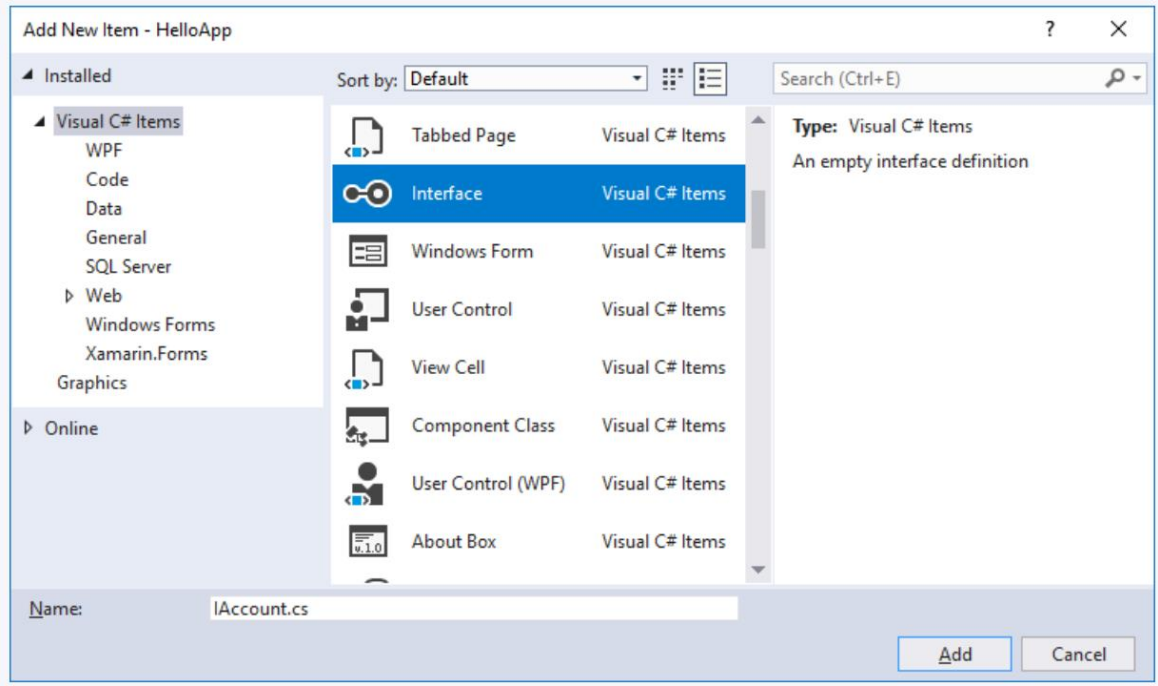
```
        // находим время, за которое надо пройти расстояние distance со
скоростью speed
    static double GetTime(double distance, double speed) => distance / speed;
    static int MaxSpeed
    {
        get { return maxSpeed; }
        set
        {
            if (value > 0) maxSpeed = value;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(IMovable.MaxSpeed);
        IMovable.MaxSpeed = 65;
        Console.WriteLine(IMovable.MaxSpeed);
        double time = IMovable.GetTime(100, 10);
        Console.WriteLine(time);
    }
}
```

Interfeysga kirish modifikatorlari

Sinflar singari, interfeyslar sukut bo'yicha ichki kirish darajasiga ega, ya'ni bunday interfeys faqat joriy loyihada mavjud. Ammo umumiy modifikator bilan biz interfeysni umumiy qilishimiz mumkin:

```
public interface IMovable
{
    void Move();
}
```

Ta'kidlash joizki, Visual Studio-da alohida faylga yangi interfeys qo'shish uchun maxsus komponent mavjud. Loyihaga interfeys qo'shish uchun siz sichqonchaning o'ng tugmachasini bosishingiz va paydo bo'lgan kontekst menyusida Qo'shish -> Yangi element ... ni tanlashingiz va yangi komponent qo'shish uchun dialog oynasidagi Interface elementini tanlashingiz mumkin:



Savollar:

1. Interfesylar nima.
2. Interfeyslarni sozlash qanday amalga oshiriladi.
3. Interfeysning asosiy parametrlari.

5-Mavzu: Umumlashgan siniflar iyerarxiyasi.

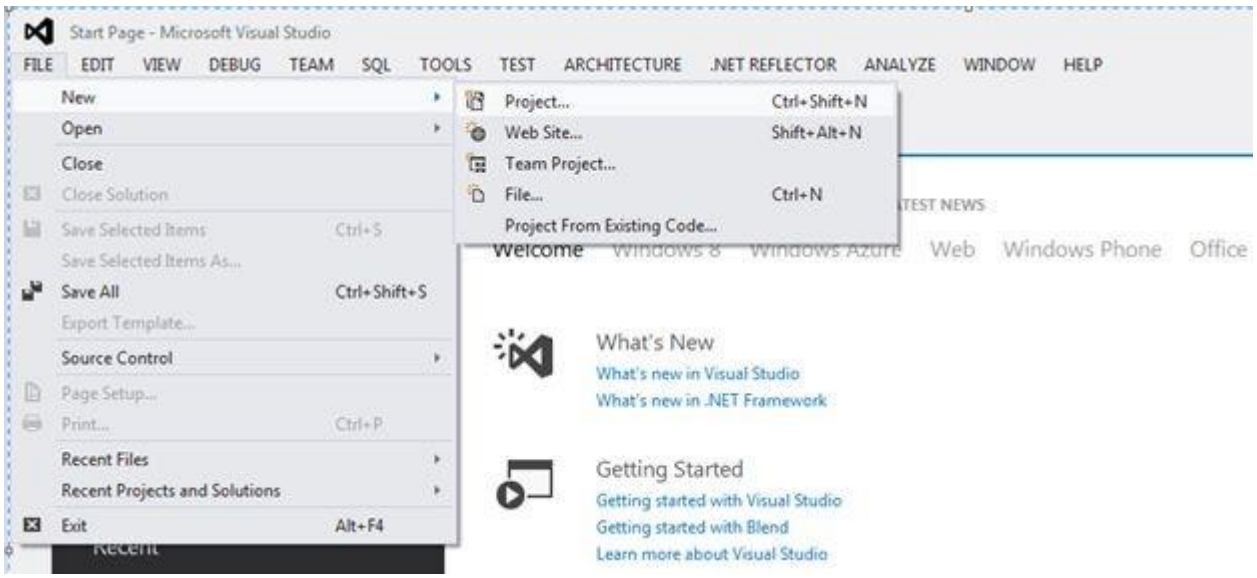
Reja:

1. **Siniflar.**
2. **Umumlashgan siniflar.**

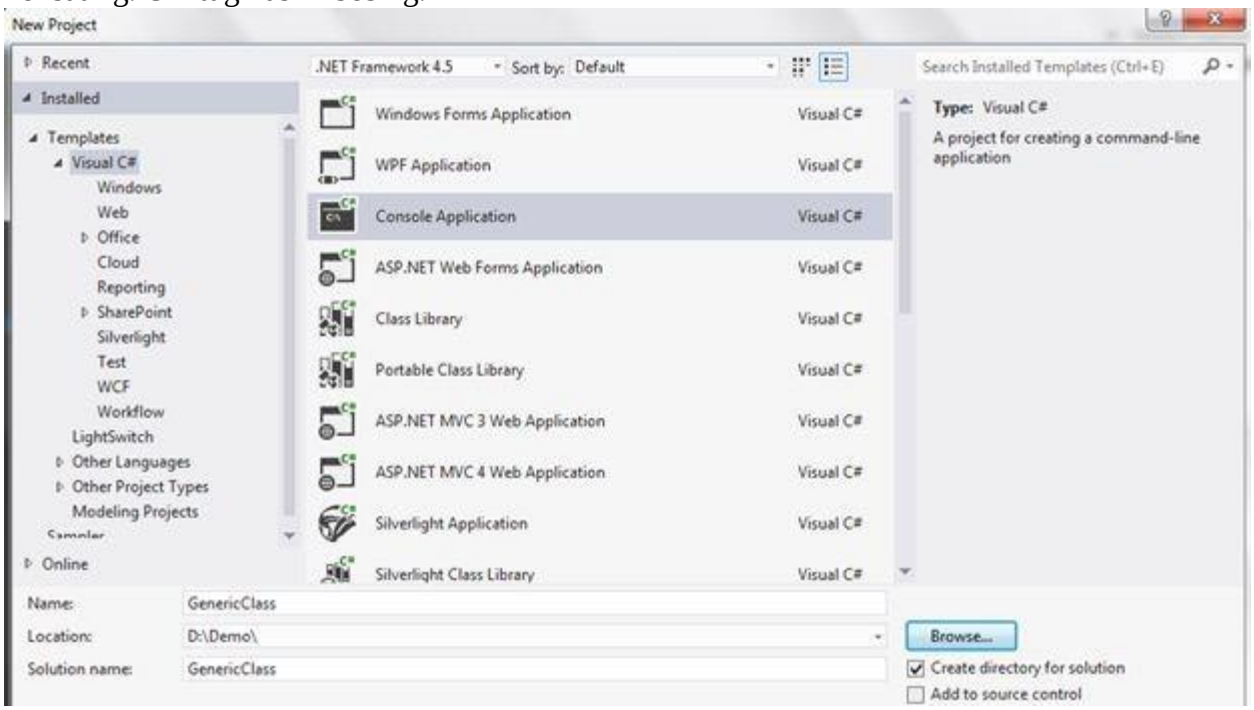
C # dagi umumiy sinflar tip parametrlarini ifodalaydi. Ular 5 ta parametrga ega. Umumiy sinf o'zi shartli sinfning bir qismiga aylanadi. T tipidagi sinf quyidagi misolda keltirilgan. T harfi asosan abonent maydoniga asoslangan turni belgilaydi.
Visual Studio dasturini ishga tushiring. Loyiha turini va chiziqli konsol dasturini tanlang.

1-qadam

Fayl yorlig'ini bosing -> Yangi -> Dastur.



2-qadam
Oynaning chap tomonida Visual C # qatorini tanlang. O'ng oynada Konsol ilovasini bosing. Dasturga "GenericClass" deb nom bering. Dasturni qaerda saqlamoqchi ekanligingizni ko'rsating. OK tugmasini bosing.

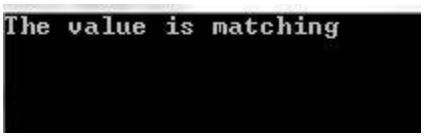


3-qadam
Qo'shimcha ravishda quyidagi kodni kiriting:

```
public class Myclass

{

    public void Compareme(T v1, T v2)

        {

        if (v1.Equals(v2))
```

```
            {
                Console.Write("The value is matching");
            }
        else
            {
                Console.Write("The value is not matching");
            }
        }
}
class Program
{
    static void Main(string[] args)
        {
            Myclass objmyint = new Myclass();
            objmyint.Compareme("Amit", "Amit");
            Console.ReadLine();
    }
}
```

Kodni kiritgandan so'ng, siz bunday dastlabki ma'lumotlarni olasiz.



Chiziqni boshqa narsaga o'zgartiring, asl ma'lumotni tekshiring. Misol "amit" ikkinchi satr parametrini yaratadi.

```
public class Myclass
{
    public void Compareme(T v1, T v2)
        {
```

```
        if (v1.Equals(v2))
                {
                        Console.Write("The value is matching");
                }
        else
                {
                        Console.Write("The value is not matching");
                }
        }
}
class Program
{
    static void Main(string[] args)
        {
                Myclass objmyint = new Myclass();
                objmyint.Compareme("Amit", "amit");
                Console.ReadLine();
    }
}
```

Natijaga qarang.

Umumiy (yozilgan) sinf quyidagilarni o'z ichiga olishi mumkin.

• umumiy parametrlarga ega usullar	class Book<**T**>
	{
	// метод с универсальным параметром, потому что тип T
	public void **SetPrice**(**T** price) // T может быть любым типом
	{
	Price = price;
	}
	}

• universal sinf a'zolari	class Book<**T**>

	{
	// универсальный член класса, потому что тип T
	protected **T Price**; // T может быть любым типом
	...
	}

• universal xususiyatlar	class Book<**T**>
	{
	// универсальное свойство, потому что тип T
	public **T Money** { set; get; } // T может быть любым типом
	...
	}

C # kutubxonasidagi umumiy (tipik) sinflarga misollar
C # ning umumiy (yozilgan) to'plamlari mavjud:

List<T>	elementlarni qo'shish, olib tashlash uchun sinf
LinkedList<T>	har bir element 2 ta havolani saqlaydigan ikki tomonlama bog'langan ro'yxat: keyingi elementga va oldingi elementga havola.
SortedList<TKey, TValue>	kalitlar bo'yicha tartiblangan kalit-qiymat juftlari to'plamlarini saqlaydigan sinf
Stack<T>	sinfda birinchi bo'lib amalga oshiriladigan algoritm qo'llaniladi.
Queue<T>	sinf birinchi bo'lib birinchi bo'lib chiqadigan algoritmdan foydalanadi. do'konda navbat bilan bir xil
HashSet<T>	HashSet <T> elementlarning takrorlanmaydigan to'plamini o'z ichiga oladi. HashSet <T> elementlarni tartibsiz (saralashsiz) o'z ichiga oladi. HashSet <T> sizga bunday element mavjudligini yoki yo'qligini tezda aniqlashga imkon beradi (chunki tezda elementning xash kodidan hisoblangan indeksdan foydalaniladi). HashSet <T> - Qo'shish, O'chirish, O'z ichiga olish usullari mavjud, ammo u xash dasturidan foydalanganligi sababli ushbu operatsiyalar 1 ta harakatni amalga oshiradi (<T> ro'yxatidagi Contains and Remove usullari n ta harakatni amalga oshiradi.) HashSet <T> usullariga ega: UnionWith (birlashma) Kesishish bilan (kesishish) Bundan tashqari (farq) SymmetricExceptWith (nosimmetrik farq)
SortedSet<T>	SortSet <T> elementlarning takrorlanmaydigan to'plamini o'z ichiga oladi. SortedSet <T> tartiblangan tartibda

	elementlarni o'z ichiga oladi (elementlar saralangan). SortedSet <T> sizga bunday element mavjudligini yoki yo'qligini tezda aniqlashga imkon beradi (chunki tezda elementning xash kodidan hisoblangan indeksdan foydalaniladi). SortedSet <T> - qo'shish, olib tashlash, o'z ichiga olish usullari mavjud, ammo u xash dasturidan foydalanganligi sababli, ushbu operatsiyalar 1 ta harakatni amalga oshiradi (<T> ro'yxatidagi Contains and olib tashlash usullari n ta harakatni amalga oshiradi.) SortSet <T> usullariga ega: UnionWith (birlashma) Kesishish bilan (kesishish) Bundan tashqari (farq) SymmetricExceptWith (nosimmetrik farq)
ObservableCollection<T>	to'plamdagi narsalarni qo'shish, olib tashlash, almashtirish paytida xabarnomalarni oladigan ma'lumotlar to'plami
Dictionary<TKey, TValue>	kalit-qiymat juftliklari to'plamini saqlash klassi
SortedDictionary<TKey, TValue>	Bu kalit bilan buyurtma qilingan kalit-qiymat juftliklari to'plami.
ConcurrentDictionary<TKey, TValue>	Bu bir nechta iplar bilan bir vaqtning o'zida kirish mumkin bo'lgan kalit / qiymat juftliklarining xavfsiz to'plamidir.

Umumiy (tipik) sinflarga umumiy nuqtai
• Kodni maksimal darajada qayta ishlatish, xavfsizlik xavfsizligi va ishlash uchun umumiy turlardan foydalaning.

• Umumiy shablonlar uchun eng ko'p qo'llaniladigan holat - bu C # da yozilgan umumiy to'plamlar.

• Siz quyidagilarni yaratishingiz mumkin:
umumiy (yozilgan) interfeyslar
umumiy (yozilgan) usullar
umumiy (yozilgan) hodisalar
umumiy (terilgan) delegatlar

• Umumiy (tipik) sinflarga metodlarga kirish ma'lum ma'lumotlar turlari bilan cheklanishi mumkin. 2-misolga qarang

• Ma'lumotlarning umumiy turida ishlatiladigan turlar haqida ma'lumotni aks ettirish yordamida olish mumkin.

Savollar:

1. Siniflar qanday hususiyatga ega.
2. Siniflarni ishlatish jarayoni.
3. Umumlashgan siniflar.

6-Mavzu: Delegatlar
Reja:

1. **Metodlar va delegatlar.**
2. **Delegatlarni aniqlash.**

Delegat - bu metodga murojaat qilishi mumkin bo'lgan ob'ekt. Shuning uchun, delegat yaratilganda, yakuniy natijada metod ma'lumotnomasini o'z ichiga olgan ob'ekt bo'ladi. Bundan tashqari, usulni ushbu havoladan chaqirish mumkin. Boshqacha qilib aytganda, vakil sizga murojaat qilgan usulni chaqirishga imkon beradi.

Aslida, delegat - bu dasturda keyinroq chaqirilishi mumkin bo'lgan boshqa usulga (yoki ehtimol usullar ro'yxatiga) ishora qiluvchi xavfsiz turdagi ob'ekt. Xususan, delegat ob'ekti uchta muhim ma'lumotlarni saqlaydi:

u chaqirilgan usulning manzili;

ushbu uslubning argumentlari (agar mavjud bo'lsa);

ushbu usulning qaytish qiymati (agar mavjud bo'lsa).

Vakil yaratilgandan va kerakli ma'lumotlar bilan ta'minlanganidan so'ng, u ish vaqtida ko'rsatadigan usullarni dinamik ravishda chaqira oladi. .NET Framework-dagi har bir delegat (shu jumladan maxsus delegatlar) avtomatik ravishda o'z usullarini sinxron yoki asenkron chaqirish imkoniyatiga ega. Bu haqiqat dasturlash vazifalarini ancha soddalashtiradi, chunki bu sizga Thread ob'ektini qo'lda yaratmasdan va manipulyatsiya qilmasdan, ikkilamchi bajarilish satrida usul chaqirishga imkon beradi.

C # da delegat turini aniqlash

Vakil turi delegat so'zi yordamida e'lon qilinadi. Quyida delegat e'lon qilishning umumiy shakli keltirilgan:

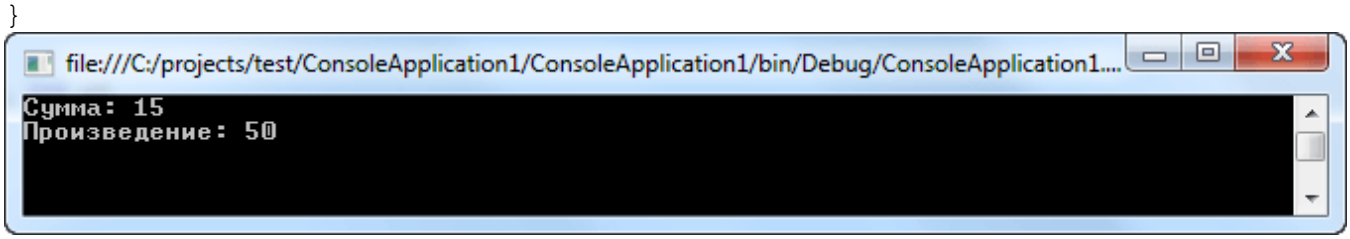
bu erda return_type delegat tomonidan chaqiriladigan usullar bilan qaytarilgan qiymat turini bildiradi; ism - delegatning aniq ismi; parameter_list - vakil tomonidan chaqirilgan usullar uchun zarur bo'lgan parametrlar. Vakil qo'zg'atilgandan so'ng, qaytish turi va parametrlari delegat deklaratsiyasida ko'rsatilganlarga mos keladigan usullarni chaqirishi va murojaat qilishi mumkin.

Eng muhimi, delegat har qanday usulni tegishli imzo va qaytish turiga qo'ng'iroq qilish uchun ishlatilishi mumkin. Bundan tashqari, chaqirilayotgan usul alohida ob'ekt bilan bog'langan instansiya usuli yoki ma'lum bir sinf bilan bog'liq statik usul bo'lishi mumkin. Faqatgina muhim narsa shundaki, uslubning qaytish turi va imzosi delegat deklaratsiyasida ko'rsatilganlarga mos kelishi kerak.

Misol keltiraylik:

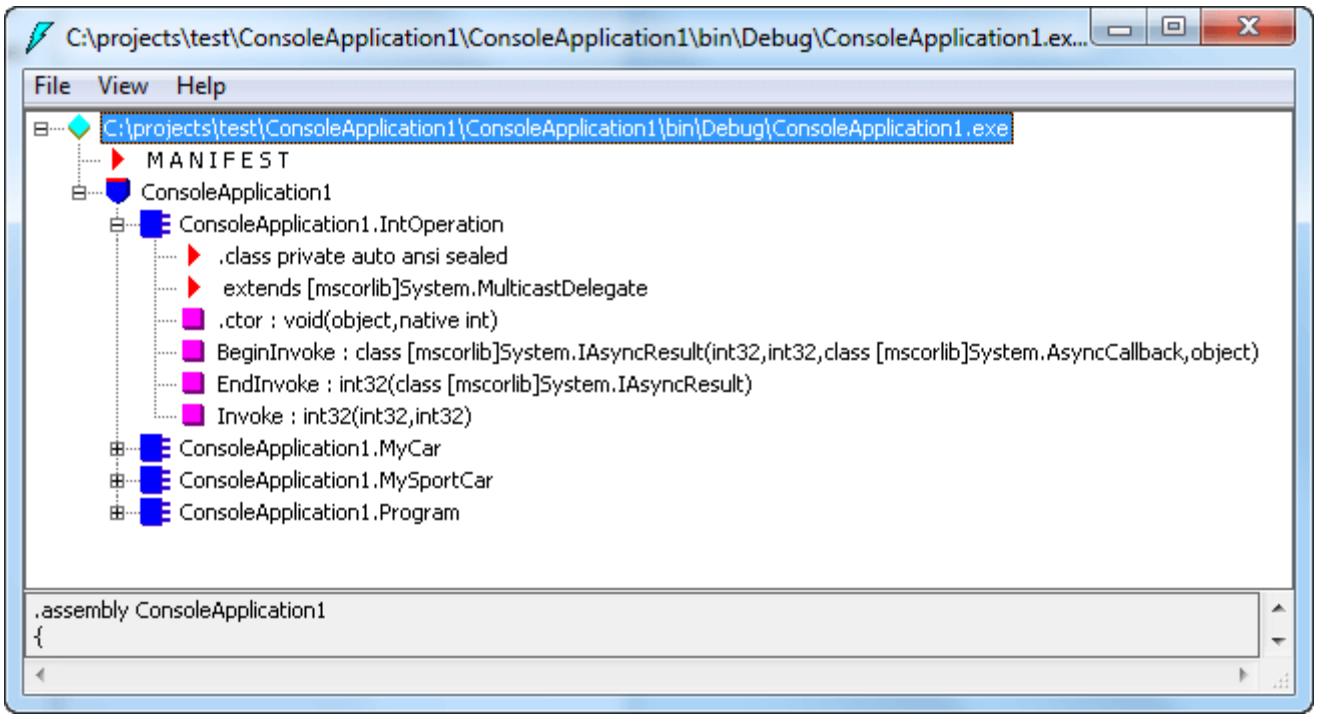
```
using System;

namespace ConsoleApplication1
{
    // Создадим делегат
    delegate int IntOperation (int i, int j);

    class Program
    {
```

```
        // Организуем ряд методов
        static int Sum(int x, int y)
        {
            return x + y;
        }

        static int Prz(int x, int y)
        {
            return x * y;
        }

        static int Del(int x, int y)
        {
            return x / y;
        }

        static void Main()
        {
            // Сконструируем делегат
            IntOperation op1 = new IntOperation(Sum);

            int result = op1(5, 10);
            Console.WriteLine("Сумма: " + result);

            // Изменим ссылку на метод
            op1 = new IntOperation(Prz);
            result = op1(5, 10);
            Console.WriteLine("Произведение: " + result);

            Console.ReadLine();
        }
    }
}
```



Ushbu misoldan olingan asosiy narsa quyidagicha: IntOperation delegat misoli chaqirilganda, u murojaat qilgan usul chaqiriladi. Shuning uchun, usul qo'ng'iroqlari kompilyatsiya vaqtida emas, balki ish vaqtida hal qilinadi.

System.MulticastDelegate va System.Delegate asosiy sinflari
C # kompilyatori vakil turini qayta ishlaganda, u avtomatik ravishda System.MulticastDelegate-dan meros qilib olingan muhrlangan sinfni hosil qiladi. Ushbu sinf (System.Delegate asosiy klassi bilan birgalikda) delegat uchun keyinchalik chaqiriladigan usullar ro'yxatini saqlash uchun zarur infratuzilmani ta'minlaydi. Masalan, agar siz ilgari misoldan IntOperation delegatini ildasm.exe yordam dasturi yordamida ko'rsangiz, siz rasmda ko'rsatilgan sinfni topasiz:



Ko'rib turganingizdek, kompilyator tomonidan yaratilgan IntOperation klassi uchta ommaviy usulni belgilaydi. Invoke (), ehtimol asosiy hisoblanadi, chunki u vakil ob'ekti tomonidan qo'llab-quvvatlanadigan usullarning har birini sinxron ravishda chaqirish uchun ishlatiladi; bu shuni anglatadiki, qo'ng'iroqni davom ettirishdan oldin qo'ng'iroq tugashini kutish kerak. Sinxronlash Invoke () usuli C # kodida aniq chaqirilmasligi kerakligi g'alati tuyulishi mumkin. Tegishli C # sintaksisini qo'llanganda Invoke () parda ortida chaqiriladi.

BeginInvoke () va EndInvoke () usullari amaldagi usulni alohida bajarilish satrida asenkron ravishda chaqirish imkoniyatini taklif qiladi. Ko'p tarmoqli rivojlanish tajribasiga ega bo'lganlar shuni bilishlari kerakki, ishlab chiquvchilarni bajarilishning ikkilamchi yo'nalishlarini yaratishga majbur qilishning asosiy sabablaridan biri bu ma'lum bir vaqtni talab qiladigan usullarni chaqirish zarurati. Garchi .NET asosiy sinf kutubxonalari ko'p satrga (System.Threading) bag'ishlangan butun nom maydonini taqdim etsa ham, delegatlar ushbu funktsiyani qutidan tashqarida ta'minlaydi.

System.MulticastDelegate-ning tanlangan usullaridan ba'zilari quyida keltirilgan:

```
public abstract class MulticastDelegate : Delegate
{
  // Возвращает список методов, на которые "указывает" делегат.
  public sealed override Delegate[] GetInvocationList ();
  // Перегруженные операции.
  public static bool operator ==(MulticastDelegate d1,
MulticastDelegate d2) ;
  public static bool operator !=(MulticastDelegate d1,
MulticastDelegate d2) ;
  // Используются внутренне для управления списком методов,
поддерживаемых делегатом.
  private IntPtr _invocationCount;
  private object _invocationList;
```

```
}
```

System.MulticastDelegate sinfi o'zining asosiy System.Delegate sinfidan qo'shimcha funktsiyalar oladi. Sinf ta'rifi parchasi quyida keltirilgan:

```
public abstract class Delegate : ICloneable, ISerializable
{
  // Методы Для взаимодействия со списком функций.
  public static Delegate Combine(params Delegate[] delegates);
  public static Delegate Combine(Delegate a, Delegate b) ;
  public static Delegate Remove(Delegate source, Delegate
value);
  public static Delegate RemoveAll(Delegate source, Delegate
value);
  // Перегруженные операции.
  public static bool operator ==(Delegate d1, Delegate d2) ;
  public static bool operator != ( Delegate d1, Delegate d2) ;
  // Свойства, показывающие цель делегата.
  public Methodlnfo Method { get; }
  public object Target { get; }
}
```

O'zingizning kodingizdagi ushbu asosiy sinflardan hech qachon to'g'ridan-to'g'ri meros qilib ololmasligingizni yodda tuting (agar siz buni bajarishga harakat qilsangiz, kompilyatsiya xatosi paydo bo'ladi). Biroq, delegat kalit so'zidan foydalanishda bevosita "MulticastDelegate" sinfini yaratadi. Quyidagi jadvalda barcha delegat turlari uchun umumiy bo'lgan asosiy a'zolar tasvirlangan:

C # Delegatning usullari va xususiyatlari

Method	Ushbu xususiyat delegat tomonidan qo'llab-quvvatlanadigan statik usul tafsilotlarini aks ettiruvchi System.Reflection.Method ob'ektini qaytaradi.
Target	Agar chaqiriladigan usul ob'ekt darajasida aniqlangan bo'lsa (ya'ni, u statik emas), unda Target delegat tomonidan qo'llab-quvvatlanadigan usulni ifodalovchi ob'ektni qaytaradi. Agar qaytarilgan Maqsad qiymati null bo'lsa, u holda chaqiriladigan usul statik bo'ladi
Combine()	Ushbu statik usul usulni delegat tomonidan saqlanadigan ro'yxatga qo'shadi. C # -da bu usul stenografik yozuv sifatida haddan tashqari yuklangan operator + = yordamida chaqiriladi
GetlnvokationList()	Ushbu usul har biri chaqirilishi mumkin bo'lgan ma'lum bir uslubni aks ettiruvchi System.Delegate turlarining qatorini qaytaradi
Remove() RemoveAll()	Ushbu statik usullar usulni (yoki barcha usullarni) delegat chaqiruvlari ro'yxatidan olib tashlaydi. C # da, (=) haddan tashqari yuklangan operator orqali Remove () usulini bevosita chaqirish mumkin

Savollar:

1. Delegatlar nima.
2. Metodlar va delegatlar.
3. Delegatlar turlari.

7-Mavzu: Hodisalar

Reja:

1. Hodisalar.
2. Hodisalarni aniqlash.

Hodisalar tizimga ma'lum bir harakat sodir bo'lganligidan signal beradi. Agar biz ushbu harakatlarni kuzatib borishimiz kerak bo'lsa, unda biz faqatgina voqealarni qo'llashimiz mumkin.

Masalan, bank hisob raqamini tavsiflovchi quyidagi sinfni oling:

```
class Account
{
    public Account(int sum)
    {
        Sum = sum;
    }
    // сумма на счете
    public int Sum { get; private set;}
    // добавление средств на счет
    public void Put(int sum)
    {
        Sum += sum;
    }
    // списание средств со счета
    public void Take(int sum)
    {
        if (Sum >= sum)
        {
            Sum -= sum;
        }
    }
}
```

Konstruktorda biz Sum xossasida saqlanadigan boshlang'ich miqdorni o'rnatdik. Put usulidan foydalangan holda biz hisobvarag'imizga mablag 'qo'shishimiz mumkin, va "Take" usulidan foydalangan holda, aksincha, hisobdan pul olishimiz mumkin. Keling, dasturda sinfdan foydalanishga harakat qilaylik - hisob yarating, pul qo'ying va undan pul oling:

```
tatic void Main(string[] args)
{
    Account acc = new Account(100);
    acc.Put(20);    // добавляем на счет 20
    Console.WriteLine($"Сумма на счете: {acc.Sum}");
    acc.Take(70);   // пытаемся снять со счета 70
    Console.WriteLine($"Сумма на счете: {acc.Sum}");
    acc.Take(180);  // пытаемся снять со счета 180
```

```
    Console.WriteLine($"Сумма на счете: {acc.Sum}");
    Console.Read();
}
```

Konsol chiqishi:

Barcha operatsiyalar kutilganidek ishlaydi. Ammo biz foydalanuvchini o'z faoliyati natijalari to'g'risida xabardor qilmoqchi bo'lsak nima bo'ladi. Buning uchun, masalan, Put usulini o'zgartirishimiz mumkin:

```
public void Put(int sum)
{
    Sum += sum;
    Console.WriteLine($"На счет поступило: {sum}");
}
```

Endi biz konsolda tegishli xabarni ko'rish orqali operatsiya to'g'risida xabardor bo'lamiz. Ammo bu erda bir qator fikrlar mavjud. Sinf ta'rifi paytida, biz pulni qo'shib qo'yishga javoban Qaysi usulda harakat qilishni aniq bilmasligimiz mumkin. Bu konsolga chiqarilishi mumkin yoki ehtimol foydalanuvchini elektron pochta yoki sms orqali xabardor qilmoqchimiz. Bundan tashqari, biz ushbu sinfni o'z ichiga olgan alohida sinf kutubxonasini yaratib, uni boshqa loyihalarga qo'shishimiz mumkin. Va allaqachon ushbu loyihalardan qanday harakatlar amalga oshirilishi kerakligini hal qilish kerak. Hisob sinfini grafik dasturda ishlatishni xohlaymiz va konsolga emas, balki grafik xabarda qayd yozuviga qo'shilganda ko'rsatamiz. Yoki bizning sinf kutubxonamizni hisobga qo'shilganda nima qilish kerakligi haqida o'z fikriga ega bo'lgan boshqa ishlab chiquvchi foydalanadi. Va bu savollarning barchasini voqealar yordamida hal qilishimiz mumkin.

Hodisalarni aniqlash va ko'tarish

Voqealar sinfda voqea so'zi yordamida e'lon qilinadi, so'ngra voqeani ifodalovchi vakil turi:

```
delegate void AccountHandler(string message);
event AccountHandler Notify;
```

Bunday holda, biz avval AccountHandler vakolatxonasini aniqlaymiz, u bitta turdagi string parametrini oladi. Keyin, voqea kalit so'zidan foydalanib, AccountHandler vakili tomonidan taqdim etilgan xabar berish nomli voqea aniqlanadi. Tadbirning nomi o'zboshimchalik bilan bo'lishi mumkin, ammo har qanday holatda u qandaydir delegatni ifodalashi kerak.

Hodisani aniqlab, uni voqea nomidan foydalanib usul sifatida dasturda chaqira olamiz:

```
Notify("Произошло действие");
```

Notify hodisasi bir turdagi string string - stringni oladigan AccountHandler vakili bo'lgani uchun, biz voqeani chaqirganimizda unga mag'lubiyatni etkazishimiz kerak.

Biroq, voqealarni chaqirganda, biz hech qanday ishlov beruvchi aniqlanmagan bo'lsa, voqea null ekanligiga duch kelishimiz mumkin. Shuning uchun, tadbirga qo'ng'iroq qilishda har doim null holatini tekshirganingiz ma'qul. Masalan, shunga o'xshash:

```
if(Notify !=null) Notify("Произошло действие");
```

Yoki shunga o'xshash:

```
Notify?.Invoke("Произошло действие");
```

Bunday holda, voqea delegat bo'lganligi sababli, uni parametrlar uchun kerakli qiymatlarni o'tkazib, uni Invoke () usuli yordamida chaqirishimiz mumkin.

Barchasini birlashtiring va voqea yarating va yoqing:

```
class Account
{
    public delegate void AccountHandler(string message);
    public event AccountHandler Notify;              // 1.Определение события
    public Account(int sum)
    {
        Sum = sum;
    }
    public int Sum { get; private set;}
    public void Put(int sum)
    {
        Sum += sum;
        Notify?.Invoke($"На счет поступило: {sum}");   // 2.Вызов события
    }
    public void Take(int sum)
    {
        if (Sum >= sum)
        {
            Sum -= sum;
            Notify?.Invoke($"Со счета снято: {sum}");   // 2.Вызов события
        }
        else
        {
            Notify?.Invoke($"Недостаточно денег на счете. Текущий баланс:
{Sum}"); ;
        }
    }
}
```

Endi "Xabar berish" tadbiridan foydalanib, biz tizimga mablag 'qo'shilganligi va hisobdan mablag' olinishi yoki hisobda mablag 'etishmasligi to'g'risida xabar beramiz.

Hodisalar boshqaruvchisi qo'shilmoqda

Hodisa u bilan bog'liq bo'lgan bir yoki bir nechta ishlov beruvchiga ega bo'lishi mumkin. Voqealar ishlovchilari - voqealar chaqirilganda aniq bajariladigan narsa. Usullar ko'pincha voqea ishlovchilari sifatida ishlatiladi. Har bir voqea ishlovchisi parametrlar ro'yxati va qaytish turi

bo'yicha hodisani ifodalaydigan delegatga mos kelishi kerak. Hodisa ishlovchilarini qo'shish uchun + = amalidan foydalaning:

```
Notify += обработчик события;
```

Dasturda kerakli bildirishnomalarni olish uchun bildirish hodisasi uchun ishlovchilarni aniqlaylik:

```
class Program
{
    static void Main(string[] args)
    {
        Account acc = new Account(100);
        acc.Notify += DisplayMessage;   // Добавляем обработчик для события
Notify
        acc.Put(20);    // добавляем на счет 20
        Console.WriteLine($"Сумма на счете: {acc.Sum}");
        acc.Take(70);   // пытаемся снять со счета 70
        Console.WriteLine($"Сумма на счете: {acc.Sum}");
        acc.Take(180);  // пытаемся снять со счета 180
        Console.WriteLine($"Сумма на счете: {acc.Sum}");
        Console.Read();
    }
    private static void DisplayMessage(string message)
    {
        Console.WriteLine(message);
    }
}
```

Bunday holda, ishlovchilar sifatida DisplayMessage usuli ishlatiladi, bu parametrlar ro'yxati va qaytish turidagi AccountHandler delegatiga mos keladi. Natijada, xabar berish. DisplayMessage-da biz shunchaki voqeadan olingan xabarni namoyish qilamiz, ammo har qanday mantiqni aniqlash mumkin edi.

Agar bu holda ishlov beruvchi o'rnatilmagan bo'lsa, unda Notify? .Invoke () hodisasi chaqirilganda hech narsa bo'lmaydi, chunki Notify hodisasi bekor bo'lgan bo'lar edi.

Dasturning konsol chiqishi:

```
На счет поступило: 20
Сумма на счете: 120
Со счета снято: 70
Сумма на счете: 50
Недостаточно денег на счете. Текущий баланс: 50
Сумма на счете: 50
```

Endi biz "Account" sinfini alohida sinf kutubxonasiga ajratib, har qanday loyihaga qo'shishimiz mumkin.

Ishlovchilarni qo'shish va olib tashlash

Bitta voqea uchun siz bir nechta ishlov beruvchilarni o'rnatishingiz va keyin ularni istalgan vaqtda olib tashlashingiz mumkin. Ishlovchilarni olib tashlash uchun - = amalidan foydalaning. Masalan; misol uchun:

```
class Program
{
    static void Main(string[] args)
    {
        Account acc = new Account(100);
        acc.Notify += DisplayMessage;       // добавляем обработчик
DisplayMessage
        acc.Notify += DisplayRedMessage;    // добавляем обработчик
DisplayMessage
        acc.Put(20);    // добавляем на счет 20
        acc.Notify -= DisplayRedMessage;     // удаляем обработчик
DisplayRedMessage
        acc.Put(20);    // добавляем на счет 20
        Console.Read();
    }

    private static void DisplayMessage(string message)
    {
        Console.WriteLine(message);
    }

    private static void DisplayRedMessage(String message)
    {
        // Устанавливаем красный цвет символов
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine(message);
        // Сбрасываем настройки цвета
        Console.ResetColor();
    }
}
```

Konsol chiqishi:

Qabul qilingan hisob: 20
Hisobga olingan: 20
Qabul qilingan hisob: 20

Ishlovchilar sifatida nafaqat oddiy usullardan, balki delegatlardan, noma'lum usullardan va lambda iboralaridan ham foydalanish mumkin. Delegatlar va usullardan foydalanish:

```
static void Main(string[] args)
{
    Account acc = new Account(100);
    // установка делегата, который указывает на метод DisplayMessage
    acc.Notify += new ActionHandler(DisplayMessage);
    // установка в качестве обработчика метода DisplayMessage
```

```
    acc.Notify += DisplayMessage;       // добавляем обработчик
DisplayMessage

    acc.Put(20);    // добавляем на счет 20
    Console.Read();
}

private static void DisplayMessage(string message)
{
    Console.WriteLine(message);
}
```

Bunday holda, ikkita ishlovchilar o'rtasida farq bo'lmaydi.

Anonim usulni ishlov beruvchi sifatida o'rnatish:

```
static void Main(string[] args)
{
    Account acc = new Account(100);
    acc.Notify += delegate (string mes)
    {
        Console.WriteLine(mes);
    };

    acc.Put(20);
    Console.Read();
}
```

Lambda ifodasi uchun ishlov beruvchi sifatida sozlash:

```
static void Main(string[] args)
{
    Account acc = new Account(100);
    acc.Notify += mes =>Console.WriteLine(mes);

    acc.Put(20);
    Console.Read();
}
```

Savollar:

1. Hodisalar nima.
2. Hodisalar qanday aniqlanadi.
3. Hoqisalarni boshqarish

8-Mavzu: Lyamda-ifodalar

Reja:

1. **Lyamda-ifodalar.**
2. **Lambda ifodalarini blokirovka.**

C # 3.0 dan boshlab delegatlarga lambda ifodalari deb nomlangan kodlarni tayinlash uchun yangi sintaksis mavjud. Lambda iboralari vakolat turi parametri bo'lgan joyda ishlatilishi mumkin.

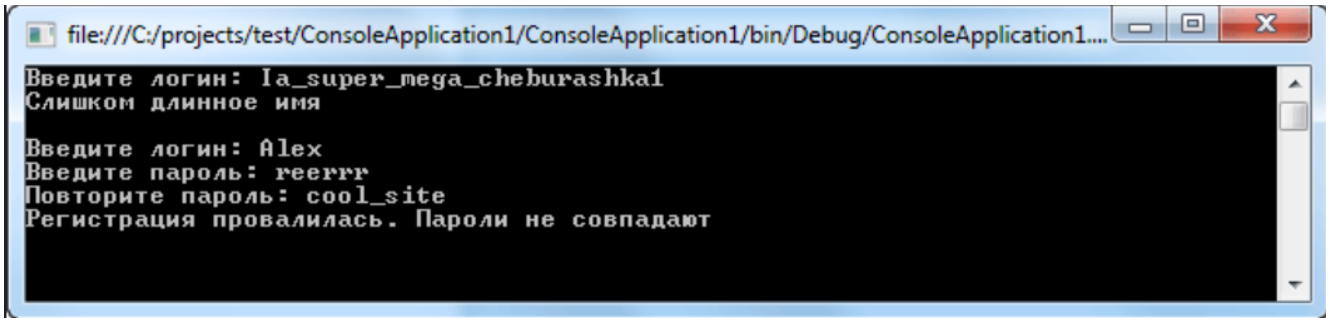
Lambda ifodalari uchun sintaksis anonim usullar sintaksisiga qaraganda sodda. Agar chaqiriladigan usul parametrlarga ega bo'lsa va bu parametrlar kerak bo'lmasa, anonim usullarning sintaksisini soddalashtiradi, chunki bu holda siz parametrlarni ko'rsatishingiz shart emas.
Barcha lambda iboralarida lambda ifodasini ikkiga bo'ladigan yangi lambda operatori => ishlatiladi. Kirish parametri (yoki bir nechta parametr) uning chap tomonida, lambda ifodasining tanasi esa o'ng tomonida ko'rsatilgan. => Operatori ba'zida "ketadi" yoki "bo'ladi" kabi so'zlar bilan tavsiflanadi.
C # lambda ifodasining tanasiga qarab, lambda iboralarining ikkita ta'mini qo'llab-quvvatlaydi. Shunday qilib, agar lambda ifodasining tanasi bitta ifodadan iborat bo'lsa, unda bitta lambda ifodasi hosil bo'ladi. Bunday holda, ifoda tanasi jingalak qavslar ichiga kiritilmagan. Agar lambda ifodasining tanasi jingalak qavslar ichiga olingan bayonotlar blokidan iborat bo'lsa, unda blok lambda ifodasi hosil bo'ladi. Biroq, bloklangan lambda ifodasi turli xil bayonotlarni o'z ichiga olishi mumkin, shu jumladan looplar, usul qo'ng'iroqlari va if operatorlari. Lambda iboralarining har ikkala turi quyida alohida muhokama qilinadi.

Yagona lambda iboralari

Bitta lambda ifodasida => operatorining o'ng tomonidagi qismi chap tomonda ko'rsatilgan parametrga (yoki parametrlar to'plamiga) ta'sir qiladi. Bunday ifodani baholashning qaytish natijasi lambda operatorini bajarish natijasidir. Quyida bitta parametrni qabul qiladigan bitta lambda ifodasining umumiy shakli keltirilgan:
Shuning uchun, ikki yoki undan ortiq parametr kerak bo'lganda, ular qavs ichiga olinishi kerak. Agar ifoda parametrlarni talab qilmasa, bo'sh qavslardan foydalanish kerak.
Lambda ifodasi ikki bosqichda qo'llaniladi. Birinchidan, lambda ifodasi bilan mos keladigan vakil turi, keyin esa lambda ifodasi tayinlangan vakil nusxasi e'lon qilinadi. Lambda ifodasi keyinchalik vakil nusxasiga kirishda baholanadi. Qaytish qiymati uni baholash natijasiga aylanadi. Misol keltiraylik:

```
using System;

namespace ConsoleApplication1
{
    // Создадим несколько делегатов имитирующих
    // простейшую форму регистрации
    delegate int LengthLogin(string s);
    delegate bool BoolPassword(string s1, string s2);

    class Program
    {
        private static void SetLogin()
        {
            Console.Write("Введите логин: ");
            string login = Console.ReadLine();

            // Используем лямбда-выражение
            LengthLogin lengthLoginDelegate = s => s.Length;

            int lengthLogin = lengthLoginDelegate(login);
            if (lengthLogin > 25)
            {
                Console.WriteLine("Слишком длинное имя\n");

                // Рекурсия на этот же метод, чтобы ввести
заново логин
```

```
                SetLogin();
            }
        }

        static void Main()
        {
            SetLogin();

            Console.Write("Введите пароль: ");
            string password1 = Console.ReadLine();
            Console.Write("Повторите пароль: ");
            string password2 = Console.ReadLine();

            // Используем лямбда выражение
            BoolPassword bp = (s1, s2) => s1 == s2;

            if (bp(password1, password2))
                Console.WriteLine("Регистрация удалась!");
            else
                Console.WriteLine("Регистрация провалилась.
Пароли не совпадают");

            Console.ReadLine();
        }
    }
}
```



Lambda ifodalarini blokirovka qiling

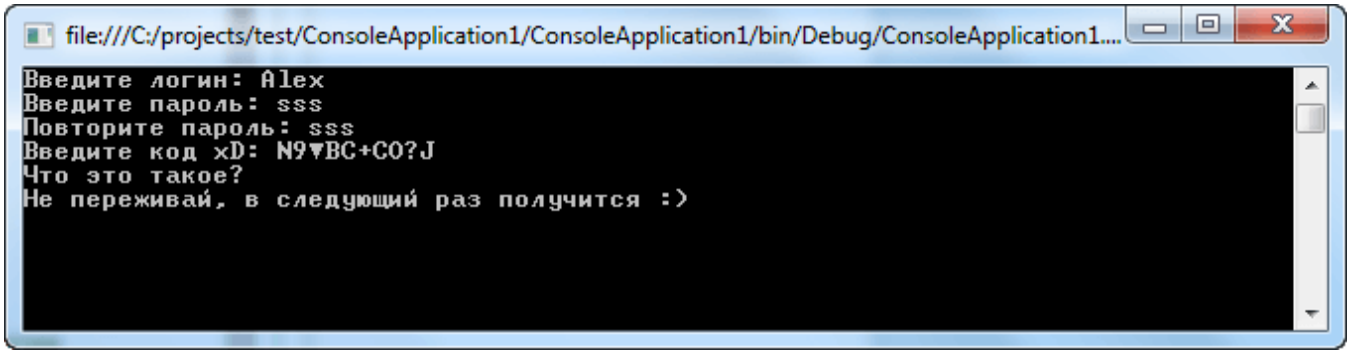
Lambda ifodasining ikkinchi turi blokli lambda ifodasidir. Bunday lambda ifodasi turli xil operatsiyalarni bajarish uchun kengaytirilgan imkoniyatlar bilan tavsiflanadi, chunki uning tanasida bir nechta operatorlar ko'rsatilishi mumkin. Masalan, bloklangan lambda ifodasida siz ko'chadan foydalanishingiz mumkin va agar iboralar, o'zgaruvchilarni e'lon qilsangiz va hk. Blok lambda ifodasini yaratish qiyin emas. Buning uchun ifoda korpusini jingalak qavs ichiga solib qo'yish kifoya. Bir nechta operatorlardan foydalanish qobiliyatidan tashqari, blok lambda ifodasining qolgan qismi hozirda muhokama qilingan bitta lambda ifodasidan deyarli farq qilmaydi.

Ro'yxatdan o'tish shakliga captcha qo'shib, avvalgi misolni o'zgartiraylik:

```
using System;

namespace ConsoleApplication1
{
    // Создадим несколько делегатов имитирующих
```

```
    // простейшую форму регистрации
    delegate int LengthLogin(string s);
    delegate bool BoolPassword(string s1, string s2);
    delegate void Captha(string s1, string s2);

    class Program
    {
        private static void SetLogin()
        {
            Console.Write("Введите логин: ");
            string login = Console.ReadLine();

            // Используем лямбда-выражение
            LengthLogin lengthLoginDelegate = s => s.Length;

            int lengthLogin = lengthLoginDelegate(login);
            if (lengthLogin > 25)
            {
                Console.WriteLine("Слишком длинное имя\n");

                // Рекурсия на этот же метод, чтобы ввести
заново логин
                SetLogin();
            }
        }

        static void Main()
        {
            SetLogin();

            Console.Write("Введите пароль: ");
            string password1 = Console.ReadLine();
            Console.Write("Повторите пароль: ");
            string password2 = Console.ReadLine();

            // Используем лямбда выражение
            BoolPassword bp = (s1, s2) => s1 == s2;

            if (bp(password1, password2))
            {
                Random ran = new Random();
                string resCaptha = "";
                for (int i = 0; i < 10; i++)
                    resCaptha += (char)ran.Next(0, 100);
                Console.WriteLine("Введите код xD: " +
resCaptha);
                string resCode = Console.ReadLine();

                // Реализуем блочное лямбда-выражение
                Captha cp = (s1, s2) =>
                {
                    if (s1 == s2)
```

```
                        Console.WriteLine("Регистрация
удалась!");
                    else
                        Console.WriteLine("Не переживай, в
следующий раз получится :)");
                    return;
                };
                cp(resCaptha, resCode);
            }
            else
                Console.WriteLine("Регистрация провалилась.
Пароли не совпадают");

            Console.ReadLine();
        }
    }
}
```



Savollar:

1. **Lambda ifodalarini blokirovka qilish.**
2. **Lambda ifodalar nima.**
3. **Yagona Lambda ifodalar.**

9-Mavzu: Obektlarga LINQ texnologiyalarini qo’llash.

Reja:

1. **LINQ texnologiyalari.**
2. **Obektlarga LINQ texnologiyalarini qo’llash.**

LINQ-ni ob'ektlarga qo'llash, siz to'plamdan ob'ektni so'roq qilish uchun LINQ-dan foydalanishingiz mumkinligini anglatadi. LINQ yordamida RAMda saqlanadigan ma'lumotlar tuzilmalariga (xotiradagi ma'lumotlar tuzilmalariga) kirish uchun foydalanish mumkin. Umumiy tipdagi IEnumerable yoki IEnumerable interfeyslarini amalga oshiradigan har qanday turdagi ob'ektlar uchun so'rov o'tkazish mumkin. Ro'yxatlar, massivlar, lug'atlar - LINQ yordamida so'ralgan ba'zi ob'ektlar to'plamlari.

Ushbu maqola LINQ operatorlari yordamida turli xil ob'ektlarga qanday qilib so'rovlar berishni va ma'lumotlarni filtrlash uchun loop usulidan foydalanishning oldini olishni ko'rsatib beradi.

LINQ-dan foydalanmasdan, siz qiymatlarni qayta-qayta ko'rib chiqishingiz va keyin kerakli ma'lumotlarni topishingiz kerak. Biroq, LINQ-dan foydalanib, siz to'g'ridan-to'g'ri ma'lumotlar to'plamlarini so'rashingiz va looping-dan foydalanmasdan kerakli qiymatlarni filtrlashingiz mumkin. LINQ katta miqdordagi manba kodini talab qilmaydigan kuchli filtrlash, guruhlash va buyurtma qilish imkoniyatlarini taqdim etadi. Masalan, agar siz assambleyada saqlanadigan

turlarni aniqlab olishingiz kerak bo'lsa, unda kerakli ma'lumotlarni filtrlasangiz, LINQ yordamida System.Reflection sinflari yordamida yig'ilish tafsilotlarini so'rashingiz mumkin.

System.Reflection nom maydonida yig'ilishlar, modullar, a'zolar, parametrlar va boshqa ob'ektlar haqidagi ma'lumotlarni metama'lumotlarini o'rganish orqali boshqariladigan kodlar to'plami sifatida olish turlari mavjud. Bundan tashqari, papkada joylashgan fayllar ob'ektlar to'plamidir va ushbu ob'ektlar LINQ yordamida so'ralishi mumkin. So'rovlarning ba'zi bir misollari quyida keltirilgan.

Butun sonlar massivi

Quyidagi misolda butun sonlar to'plamini o'z ichiga olgan butun sonli qator ko'rsatilgan. Siz kerakli qiymatlarni olish uchun massivda LINQ so'rovlaridan foydalanishingiz mumkin.

```
int[] integers = { 1, 6, 2, 27, 10, 33, 12, 8, 14, 5 };
       IEnumerable twoDigits =
       from numbers in integers
       where numbers >= 10
       select numbers;
       Console.WriteLine("Integers > 10:");
       foreach (var number in twoDigits)
       {
          Console.WriteLine(number);
       }
```

o'zgaruvchisi turli xil qiymatlarga ega bo'lgan bir qator tamsayılardan iborat. IEnumerable turidagi twoDigits o'zgaruvchisi so'rovni bajaradi. Natija olish uchun so'rov bajarilishi kerak

So'rovning bajarilishi so'rovning o'zgaruvchisi natijani sanab olish uchun GetEnumerator () ga qo'ng'iroq qilish orqali amalga oshiriladi. IEnumerable turidagi har qanday o'zgaruvchini foreach ko'chadan foydalanib sanab o'tish mumkin. IEnumerable yoki IQueryable kabi olingan interfeysni qo'llab-quvvatlovchi turlar so'raladigan turlar deb nomlanadi. ArrayList kabi ba'zi bir tipik bo'lmagan ma'lumotlar to'plamlari ham mavjud, ular LINQ yordamida so'ralishi mumkin. Buning uchun quyida keltirilgan misollarda bo'lgani kabi to'plamdagi ma'lum bir turdagi ob'ekt uchun tartiblangan o'zgaruvchining turini aniq e'lon qilishingiz kerak.

O'zgaruvchisi twoDigits kamida 10 ga teng qiymatlarni olish uchun so'rovni bajaradi. Shunday qilib, raqamlar qatordan birma-bir olinadi. Loop so'rovni bajaradi va keyin massivdan olingan qiymatlarni konsolga chop etadi. Ko'rib turganingizdek, yuqoridagi ma'lumotlar to'plamdan kerakli ma'lumotlarni olishning juda oddiy usulini namoyish etadi.

Agar sizga to'plamning faqat dastlabki to'rtta qiymati kerak bo'lsa, kerakli to'plamda Take () so'rov usulidan foydalanishingiz mumkin. Quyida siz to'plamning dastlabki to'rt elementini qanday ajratib olishingiz va ularni ilmoq yordamida konsolga bosib chiqarishingiz mumkin.

```
IEnumerable firstFourNumbers = integers.Take(4);
   Console.WriteLine("First 4 numbers:");
   foreach (var num in firstFourNumbers)
   {
      Console.WriteLine(num);
   }
```

Take () uslubiga qarama-qarshi bo'lib, bu Skip () operatori bo'lib, u ma'lum elementlarning bir qismini o'tkazib yuborish va qolgan qismini olish uchun ishlatiladi. Quyidagi misol dastlabki 4 elementni o'tkazib yuboradi.

```
IEnumerable skipFirstFourNumbers = integers.Skip(4);
   Console.WriteLine("Skip first 4 numbers:");
   foreach (var num in skipFirstFourNumbers)
   {
      Console.WriteLine(num);
   }
```

Yuqoridagi misollar ma'lum bir qator boshlang'ich ro'yxat elementlarini qanday olib kelishni / o'tkazib yuborishni ko'rsatib berdi. Agar oldindan noma'lum sonli elementlarni ajratib olish / o'tkazib yuborish zarur bo'lsa, mos kelguniga qadar ishlaydigan TakeWhile () va SkipWhile () usullari qo'llaniladi.

Quyidagi kod sizga to'plamdagi barcha raqamlarni 50 qiymatiga qadar qanday olishni ko'rsatib beradi. TakeWhile () shart mavjud bo'lganda to'plamga elementlarni kiritish uchun iborani ishlatadi va ro'yxatdagi boshqa elementlarni e'tiborsiz qoldiradi. Ifoda - bu match uchun to'plam elementlarini tekshiradigan shart.

```
int[] integers = { 1, 9, 5, 3, 7, 2, 11, 23, 50, 41, 6, 8 };
   IEnmerable takeWhileNumber = integers.TakeWhile(num =>
      num.CompareTo(50) != 0);
   Console.WriteLine("Take while number equals 50");
   foreach (int num in takeWhileNumber)
      {
         Console.WriteLine(num.ToString());
      }
```

SkipWhile () usuli xuddi shu tarzda ishlaydi, faqat qiymatlarni olish o'rniga ularni o'tkazib yuboradi. Ushbu usullardan foydalanishning eng yuqori samaradorligi buyurtma qilingan ro'yxatlarda qo'llanilganda kuzatiladi. ular qidiruv sharti birinchi marta bajarilganda tugatiladi.

```
IEnumerable skipWhileNumber = integers.SkipWhile(num =>
      num.CompareTo(50) != 0);
   Console.WriteLine("Skip while number equals 50");
   foreach (int num in skipWhileNumber)
   {
      Console.WriteLine(num.ToString());
   }
```

Ob'ektlar to'plamlari

Ushbu bo'lim sizga qanday qilib o'zboshimchalik bilan ob'ektlar to'plamini so'rov qilishingiz mumkinligini ko'rsatadi. Icecream ob'ekti ishlatiladi, kollektsiya quriladi va keyin so'ralishi mumkin. Quyidagi koddagi Icecream klassi turli xil xususiyatlarni (nomi, tarkibiy qismlari, vazni, xolesterin va boshqalarni) o'z ichiga oladi.

```
public class Icecream
    {
        public string Name { get; set; }
        public string Ingredients { get; set; }
        public string TotalFat { get; set; }
        public string Cholesterol { get; set; }
        public string TotalCarbohydrates { get; set; }
        public string Protein { get; set; }
        public double Price { get; set; }
    }
```

Keyinchalik, muzqaymoqlar ro'yxati oldindan belgilangan sinf yordamida tuziladi.

```
List icecreamsList = new List
        {
            new Icecream {Name="Chocolate Fudge Icecream",
Ingredients="cream,
                milk, mono and diglycerides...",
Cholesterol="50mg",
                Protein="4g", TotalCarbohydrates="35g",
TotalFat="20g",
                Price=10.5
        },
        new Icecream {Name="Vanilla Icecream",
Ingredients="vanilla extract,
            guar gum, cream...", Cholesterol="65mg",
Protein="4g",
            TotalCarbohydrates="26g", TotalFat="16g", Price=9.80
},
            new Icecream {Name="Banana Split Icecream",
Ingredients="Banana, guar
            gum, cream...", Cholesterol="58mg", Protein="6g",
            TotalCarbohydrates="24g", TotalFat="13g", Price=7.5
}
        };
```

Icecream turidagi qiymatlarga ega uchta ob'ektning icecreamsList to'plami mavjud. Deylik, endi siz 10 dan kam bo'lgan barcha muzqaymoqlarni chiqarib olishingiz kerak. Siz looplardan foydalanishingiz mumkin, bunda ro'yxatdagi har bir buyumning narxini birin-ketin ko'rib chiqishingiz, so'ngra narxning pastroq qiymatiga ega bo'lgan moslamalarni chiqarishingiz kerak. maydon. LINQ-dan foydalanish, kerakli narsalarni topish uchun barcha ob'ektlar va ularning xususiyatlarini takrorlashdan qochishga imkon beradi, ya'ni. qidirishni osonlashtiradi. Keyinchalik, to'plamdan arzon narxdagi muzqaymoqni tanlagan so'rov taqdim etiladi. So'rovda qaerda ishlash uchun band ishlatiladi. Tashqi tomondan, so'rov relyatsion ma'lumotlar bazasidagi so'rovga o'xshaydi. So'rov IEnumerable turidagi o'zgaruvchini tsiklda sanab chiqilganda bajariladi.

```
List Icecreams = CreateIcecreamsList();
    IEnumerable IcecreamsWithLessPrice =
    from ice in Icecreams
    where ice.Price < 10
    select ice;
    Console.WriteLine("Ice Creams with price less than 10:");
    foreach (Icecream ice in IcecreamsWithLessPrice)
    {
        Console.WriteLine("{0} is {1}", ice.Name, ice.Price);
    }
```

Ob'ektlarni List ishlatilganidek saqlash uchun ArrayList-dan foydalanishingiz mumkin. So'ngra LINQ so'rovi yordamida to'plamdan kerakli narsalarni olish uchun foydalanish mumkin. Masalan, ArrayList-ga oldingi misolda bo'lgani kabi bir xil Icecreams moslamalarini qo'shish uchun quyidagi kod.

```
ArrayList arrListIcecreams = new ArrayList();
    arrListIcecreams.Add( new Icecream {Name="Chocolate Fudge
Icecream",
        Ingredients="cream, milk, mono and diglycerides...",
```

```
        Cholesterol="50mg", Protein="4g",
TotalCarbohydrates="35g",
        TotalFat="20g", Price=10.5 });
    arrListIcecreams.Add( new Icecream {Name="Vanilla Icecream",
        Ingredients="vanilla extract, guar gum, cream...",
        Cholesterol="65mg", Protein="4g",
TotalCarbohydrates="26g",
        TotalFat="16g", Price=9.80 });
    arrListIcecreams.Add( new Icecream {Name="Banana Split
Icecream",
        Ingredients="Banana, guar gum, cream...",
Cholesterol="58mg",
        Protein="6g", TotalCarbohydrates="24g", TotalFat="13g",
Price=7.5
    });
```

Keyingi so'rov ro'yxatdan arzon muzqaymoq tanlaydi.

```
var queryIcecreanList = from Icecream icecream in
arrListIcecreams
    where icecream.Price < 10
    select icecream;
```

Quyida ko'rsatilgandek, yuqoridagi so'rov bo'yicha olingan ob'ektlarning narxini ko'rsatish uchun pastadirdan foydalanishingiz mumkin.

```
foreach (Icecream ice in queryIcecreanList)
    Console.WriteLine("Icecream Price : " + ice.Price);
```

Savollar:

1. **LINQ texnologiyalari.**
2. **Massivlarni ifodalash.**
3. **Obektga LINQ texnologiyalarini qo’llash.**

10-Mavzu: LINQ operatorlari

Reja:

1. **LINQ texnologiyalari.**
2. **LINQ operatorlari**

LINQ (Language-Integrated Query) - bu ma'lumot manbasini so'rash uchun oddiy va qulay til. Ma'lumotlar manbai sifatida IEnumerable interfeysini (masalan, standart to'plamlar, massivlar) amalga oshiradigan ob'ekt, DataSet, XML hujjati bo'lishi mumkin. Ammo manba turidan qat'i nazar, LINQ har kimga ma'lumot olish uchun bir xil usulni qo'llashga imkon beradi.

LINQ ning bir nechta lazzatlari mavjud:

LINQ to Objects: Massivlar va to'plamlar bilan ishlash uchun ishlatiladi

LINQ to Entities: Entity Framework texnologiyasi orqali ma'lumotlar bazalariga kirishda foydalaniladi

LINQ dan Sql: MS SQL Server-da ma'lumotlarga kirish texnologiyasi

LINQ to XML: XML fayllari bilan ishlashda ishlatiladi

LINQ to DataSet: DataSet ob'ekti bilan ishlashda foydalaniladi

Parallel LINQ (PLINQ): parallel so'rovlarni bajarish uchun ishlatiladi

Ushbu bobda asosan LINQ-dan Ob'ektlarga e'tibor qaratilgan, ammo keyingi maqolalarda LINQ-ning boshqa lazzatlari haqida ham so'z boradi.

LINQ qulayligi nimada? Eng oddiy misolni ko'rib chiqamiz. Massivdan ma'lum bir harf bilan boshlanadigan qatorlarni tanlaymiz va natijada olingan ro'yxatni saralaymiz:

```
string[] teams = {"Бавария", "Боруссия", "Реал Мадрид", "Манчестер Сити",
"ПСЖ", "Барселона"};

var selectedTeams = new List<string>();
foreach(string s in teams)
{
    if (s.ToUpper().StartsWith("Б"))
        selectedTeams.Add(s);
}
selectedTeams.Sort();

foreach (string s in selectedTeams)
    Console.WriteLine(s);
```

Endi LINQ yordamida xuddi shunday qilaylik:

```
string[] teams = {"Бавария", "Боруссия", "Реал Мадрид", "Манчестер Сити",
"ПСЖ", "Барселона"};

var selectedTeams = from t in teams // определяем каждый объект из teams как t
                    where t.ToUpper().StartsWith("Б") //фильтрация по
критерию
                    orderby t  // упорядочиваем по возрастанию
                    select t; // выбираем объект

foreach (string s in selectedTeams)
    Console.WriteLine(s);
```

LINQ funksiyasidan foydalanish uchun System.LINQ nom maydoni faylga kiritilganligiga ishonch hosil qiling.

Shuning uchun kod kichikroq va sodda. Aslida, butun ifoda bitta satrda yozilishi mumkin edi: var selectedTeams = from t in teams where t.ToUpper().StartsWith("Б") orderby t select t. Ammo aniqroq mantiqiy buzilish uchun men har bir alohida pastki ekspressionni alohida satrga qo'ydim.

LINQ so'rovining eng sodda ta'rifi quyidagicha:

```
from переменная in набор_объектов
select переменная;
```

Xo'sh, bu LINQ so'rovi nima qiladi? Jamoa ichidagi t bandi jamoalar qatoridagi barcha elementlarni takrorlaydi va har bir elementni t deb belgilaydi. T o'zgaruvchisidan foydalanib, unga turli xil operatsiyalarni bajarishimiz mumkin.

T o'zgaruvchining turini aniqlamagan bo'lsak-da, LINQ iboralari kuchli tarzda teriladi. Ya'ni, ramka avtomatik ravishda jamoalar to'plami string ob'ektlaridan iborat ekanligini tan oladi, shuning uchun t satr sifatida ko'rib chiqiladi.

Bundan tashqari, qaerda joylashgan banddan foydalanib, ob'ektlar filtrlanadi va agar ob'ekt mezonga javob bersa (bu holda boshlang'ich harfi "B" bo'lishi kerak), unda bu ob'ekt uzatiladi.

Orderby operatori o'sish tartibida buyurtma beradi, ya'ni tanlangan moslamalarni saralaydi.

Select operatori tanlangan qiymatlarni natijalar to'plamiga o'tkazadi, LINQ ifodasi qaytaradi.

Bunday holda, LINQ ifodasining natijasi IEnumerable <T> bo'ladi. Ko'pincha, natijada tanlov var kalit so'zi yordamida aniqlanadi, so'ngra kompilyator kompilyatsiya vaqtida turni o'zi chiqaradi.

Bunday so'rovlarning afzalligi shundaki, ular SQL tilidagi so'rovlarga intuitiv ravishda o'xshashdir, ammo ba'zi bir farqlari bor.

.. in .. select sintaksisidan standartga qo'shimcha ravishda, LINQ so'rovini yaratish uchun IEnumerable interfeysi uchun aniqlangan maxsus kengaytma usullaridan foydalanishimiz mumkin. Odatda, ushbu usullar LINQ bilan bir xil funktsiyalarni ta'minlaydi va bu erda operatorlar buyurtma berishadi.

Masalan; misol uchun:

```
tring[] teams = { "Бавария", "Боруссия", "Реал Мадрид", "Манчестер Сити",
"ПСЖ", "Барселона" };

var selectedTeams = teams.Where(t=>t.ToUpper().StartsWith("B")).OrderBy(t =>
t);

foreach (string s in selectedTeams)
    Console.WriteLine(s);
```

Jamoalar.Qaerda (t => t.ToUpper (). StartsWith ("B")). OrderBy (t => t) so'rovi avvalgisiga o'xshash bo'ladi. Bu "Where and OrderBy" usullarini zanjirlashdan iborat. Ushbu usullar delegat yoki lambda ifodasini argument sifatida qabul qiladi.

Har bir kengaytma usuli LINQ operatorlari orasida analogga ega emas, ammo bu holda siz ikkala yondashuvni birlashtira olasiz. Masalan, biz standart linq sintaksisidan va tanlovdagi elementlar sonini qaytaradigan Count () kengaytma usulidan foydalanamiz:

```
int number = (from t in teams where t.ToUpper().StartsWith("B") select
t).Count();
```

Amaldagi LINQ kengaytmasi usullari ro'yxati
Select: tanlangan qiymatlarning proektsiyasini belgilaydi

Qaerda: tanlov filtrini belgilaydi

OrderBy: buyurtma buyumlarini o'sish tartibida

OrderByDescending: buyumlarni kamayish tartibida buyurtma qiladi

ThenBy: o'sish tartibida buyumlarni buyurtma qilish uchun qo'shimcha mezonlarni belgilaydi

ThenByDescending: elementlarni kamayish tartibida buyurtma qilish uchun qo'shimcha mezonlarni belgilaydi

Birlashtirish: ma'lum bir asosda ikkita to'plamga qo'shiladi

GroupBy: elementlarni kalitlarga ko'ra guruhlaydi

ToLookup: barcha elementlarni lug'atga qo'shgan holda elementlarni kalitlarga ko'ra guruhlaydi

GroupJoin: Ikkala to'plamni birlashtirish va kalitlarga ko'ra elementlarni guruhlash

Teskari: Buyurtmani teskari yo'naltirish

Hammasi: To'plamdagi barcha narsalar ma'lum bir shartga javob berishini aniqlaydi

Har qanday: To'plamdagi kamida bitta narsa ma'lum bir shartga javob berishini aniqlaydi

Tarkibida: To'plamda ma'lum bir element mavjudligini aniqlaydi

Alohida: to'plamdagi nusxalarni olib tashlaydi

Istisno: ikkita to'plamning farqini, ya'ni faqat bitta to'plamda yaratilgan elementlarni qaytaradi

Union: ikkita bir xil to'plamlarni birlashtiradi

Kesish: Ikki kolleksiyaning, ya'ni ikkala to'plamda ham bo'lgan narsalarning kesishishini qaytaradi

Count: To'plamdagi ma'lum bir shartga javob beradigan elementlar sonini sanaydi

Sum: yig'indagi raqamli qiymatlar yig'indisini hisoblaydi

O'rtacha: to'plamdagi raqamli qiymatlarning o'rtacha qiymatini hisoblab chiqadi

Min: minimal qiymatni topadi

Maks: maksimal qiymatni topadi

Qabul qiling: ma'lum miqdordagi narsalarni tanlaydi

O'tkazib yuborish: ma'lum miqdordagi narsalarni o'tkazib yuboradi

TakeWhile: Agar shart to'g'ri bo'lsa, ketma-ketlik elementlari zanjirini qaytaradi

SkipWhile: ular berilgan shartni bajarishi sharti bilan elementlarni ketma-ketlikda o'tkazib yuboradi va keyin qolgan elementlarni qaytaradi

Concat: ikkita to'plamni birlashtiradi

Zip: ma'lum bir shartga muvofiq ikkita to'plamni birlashtiradi

Birinchisi: to'plamdagi birinchi elementni tanlaydi

FirstOrDefault: to'plamdagi birinchi elementni tanlaydi yoki sukut bo'yicha qaytaradi

Yagona: to'plamning bitta elementini tanlaydi, agar to'plamda bir yoki bir nechta element bo'lsa, istisno qo'yiladi

SingleOrDefault: To'plamdagi birinchi elementni tanlaydi yoki sukut bo'yicha qaytaradi

ElementAt: ma'lum bir indeksda ketma-ketlik elementini tanlaydi

ElementAtOrDefault: Muayyan indeksda yig'ish elementini tanlaydi yoki indeks doiradan tashqarida bo'lsa, standart qiymatni qaytaradi

Oxirgi: to'plamdagi oxirgi narsani tanlaydi

LastOrDefault: to'plamdagi so'nggi elementni tanlaydi yoki standartni qaytaradi

Savollar:

1. **LINQ operatorlari.**
2. **LINQ texnologiyasi nima.**
3. **Boshqarishda LINQ texnologiyasi.**

11-12-Mavzu: WPF texnologiyasi

Reja:

1. **WPF texnologiyasi.**
2. **.NET platformasi.**

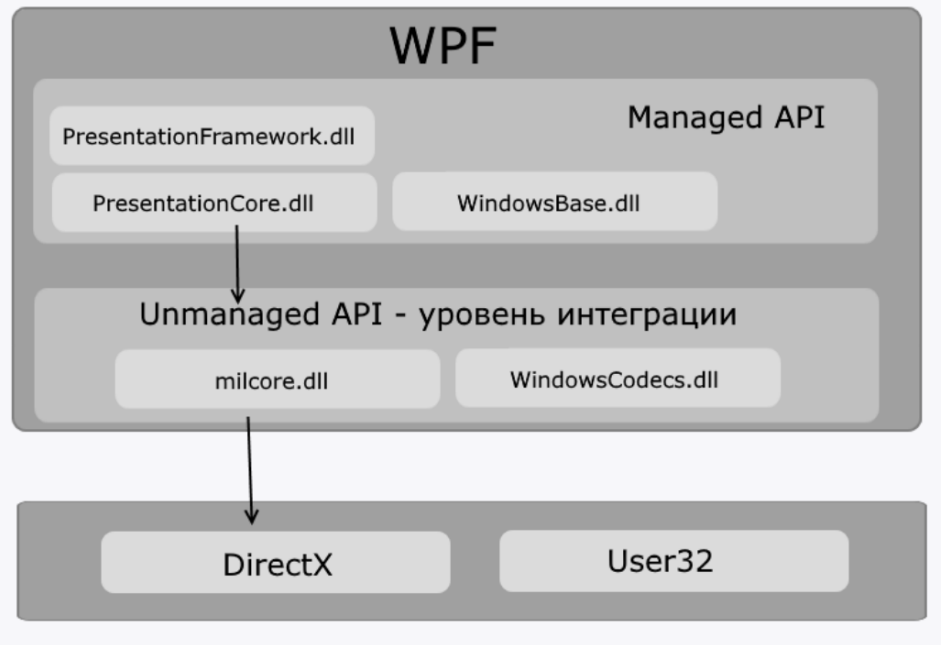
WPF (Windows Presentation Foundation) texnologiyasi .NET platformasi ekotizimining bir qismidir va grafik interfeyslarni yaratish uchun quyi tizimdir.
User32 va GDI + an'anaviy WinForms-ga asoslangan dasturlar uchun boshqaruv va grafikalarni ko'rsatish uchun mas'ul bo'lgan bo'lsa, WPF dasturlari DirectX-ga asoslangan. Bu WPF-da grafikani ko'rsatishning asosiy xususiyati: WPF-dan foydalanib, grafikani ko'rsatish bo'yicha ishlarning muhim qismi, ham oddiy tugmalar, ham murakkab 3D modellar, video kartadagi GPU-ga tushadi, bu ham sizga imkoniyatlardan foydalanishga imkon beradi apparat grafik tezlashtirish.

XML asosidagi deklarativ interfeys XAML markirovkasidan foydalanish muhim xususiyatdir: deklarativ interfeys deklaratsiyalari, boshqariladigan C # va VB.NET kodlari yoki ikkalasining aralashmasi yordamida boy GUI yaratishingiz mumkin.

WPF imtiyozlari
WPF sizga dasturchi sifatida nimani taklif qiladi?

Ilovalar mantig'ini yaratish uchun an'anaviy .NET tillaridan foydalanish - C # va VB.NET

Grafik va boshqaruv elementlarini dasturiy ravishda yaratishga alternativa beradigan xml asosida tayyorlangan XAML belgilash tili yordamida grafik interfeysni deklarativ tarzda aniqlash qobiliyati, shuningdek XAML va C # / VB.NET-ni birlashtirish qobiliyati
Mustaqil ekran ravshanligi: WPF barcha elementlarni qurilmalarning mustaqil bo'linmalarida o'lchaganligi sababli, WPF dasturlari har xil ekran o'lchamlarini har xil o'lchamlarda osongina kattalashtiradi.
WinForms-da erishish qiyin bo'lgan yangi xususiyatlar, masalan, 3D modellarni yaratish, ma'lumotlarni bog'lash, uslublar, shablonlar, mavzular va boshqalar kabi elementlardan foydalanish.
Masalan, WPF dasturlari an'anaviy WinForms boshqaruv elementlaridan foydalanishlari uchun WinForms bilan yaxshi o'zaro ishlash.
Turli xil dasturlarni yaratish uchun boy imkoniyatlar: bu multimediya va ikki o'lchovli va uch o'lchovli grafikalar, shuningdek, o'rnatilgan boshqaruv elementlari to'plami, shuningdek yangi elementlarni o'zingiz yaratish, animatsiyalar yaratish, ma'lumotlarni bog'lash, uslublar, shablonlar, mavzular va boshqa ko'p narsalar
Uskuna tezlashtirilgan grafikalar - xoh siz 2D yoki 3D bilan ishlaysizmi, grafikalar yoki matn bilan bo'lsin, barcha dastur komponentlari Direct3D tushunadigan narsalarga tarjima qilinadi, so'ngra video kartadagi protsessor yordamida ishlaydi, bu esa ish faoliyatini yaxshilaydi va grafikani yanada silliq qiladi.
Ko'p Windows operatsion tizimlari uchun dasturlar yaratish - Windows XP dan Windows 10 gacha
Shu bilan birga, WPF ma'lum cheklovlarga ega. 3D-renderni qo'llab-quvvatlashiga qaramay, 3D-rasmlarning ko'pligi, ayniqsa o'yinlari bo'lgan dasturlarni yaratish uchun boshqa vositalar - DirectX yoki Monogame yoki Unity kabi maxsus ramkalardan foydalanish yaxshiroqdir.
Shuni ham yodda tutish kerakki, Windows Forms-dagi ilovalar bilan taqqoslaganda, WPF dasturlarining hajmi va ish paytida ularning xotirani sarflashi o'rtacha darajada yuqori. Ammo bu kengroq grafik imkoniyatlari va yuqori grafik ko'rsatkichlari bilan qoplanadi.
WPF arxitekturasi
WPF arxitekturasi sxematik tarzda quyidagicha ifodalanishi mumkin:



Diagrammada ko'rib turganingizdek, WPF ikki darajaga bo'lingan: boshqariladigan API va boshqarilmaydigan API (DirectX integratsiya darajasi). Boshqariladigan API (Boshqariladigan

API) Umumiy Til Ish vaqti ostida ishlaydigan kodni o'z ichiga oladi. Ushbu API WPF platformasining asosiy funktsiyalarini tavsiflaydi va quyidagi tarkibiy qismlardan iborat:

PresentationFramework.dll: GUI qurishda ishlatilishi mumkin bo'lgan barcha asosiy komponentlar va boshqaruv dasturlarini o'z ichiga oladi

PresentationCore.dll: PresentationFramework.dll-dan ko'p sinflar uchun barcha asosiy turlarni o'z ichiga oladi

WindowsBase.dll: WPF-da ishlatiladigan, ammo ushbu platformadan tashqarida ham foydalanish mumkin bo'lgan bir qator yordamchi sinflarni o'z ichiga oladi.

Yuqori darajani DirectX bilan birlashtirish uchun boshqarilmaydigan API ishlatiladi:

milcore.dll: aslida WPF komponentlarining DirectX bilan integratsiyasini ta'minlaydi. Ushbu komponent DirectX bilan ishlash uchun boshqarilmaydigan kodda (C / C ++) yozilgan.

WindowsCodecs.dll: WPF-da rasmlarni past darajadagi qo'llab-quvvatlashni ta'minlaydigan kutubxona

Hatto undan pastroq bo'lsa ham, aslida dastur komponentlarini ta'minlaydigan yoki boshqa past darajadagi ishlov berishni amalga oshiradigan operatsion tizim va DirectX komponentlari mavjud. Xususan, DirectX tarkibiga kiruvchi past darajadagi Direct3D interfeysidan foydalanib,

Bu erda user32.dll kutubxonasi ham bir xil darajada. Yuqorida aytilgan bo'lsa-da, WPF ushbu kutubxonani ko'rsatish va ko'rsatish uchun foydalanmaydi, ammo bir qator hisoblash vazifalari uchun (shu jumladan, ko'rsatishni o'z ichiga olmaydi), ushbu kutubxonadan foydalanish davom etmoqda.

Rivojlanish tarixi
WPF .NET ekotizimining bir qismidir va .NET ramkasi bilan rivojlanadi va bir xil versiyalarga ega. WPF 3.0 birinchi marta 2006 yilda .NET 3.0 va Windows Vista bilan birga chiqarilgan. O'shandan beri platforma doimiy ravishda rivojlanib bormoqda. WPF 4.6 ning so'nggi versiyasi .NET 4.6 bilan birga 2015 yil iyul oyida platformaning to'qqiz yilligi munosabati bilan chiqarildi.

Savollar:

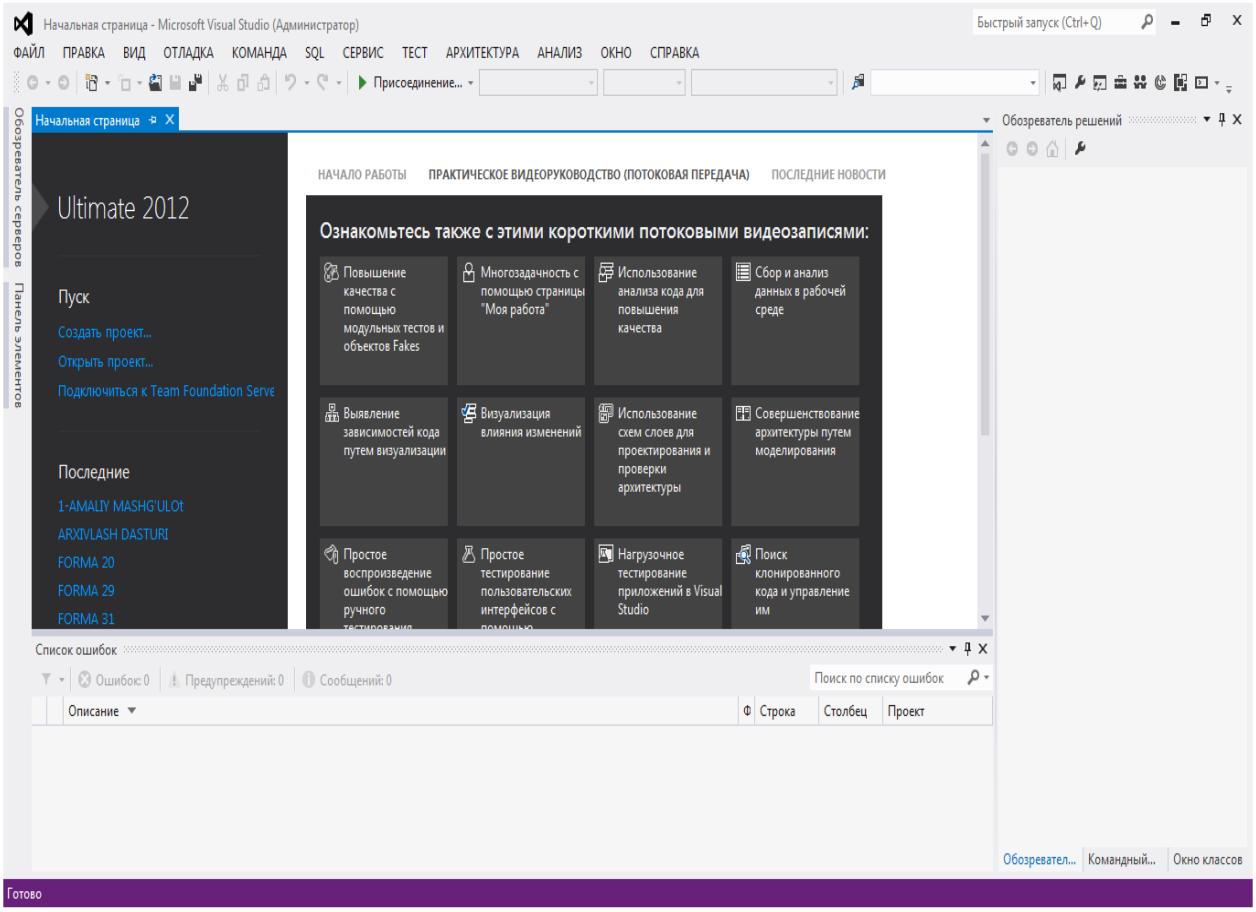
1. .NET platformasi nima.
2. WPF qanday texnologiya.
3. WPF arxitekturasi.

AMALIY MASHG`ULOTLAR.

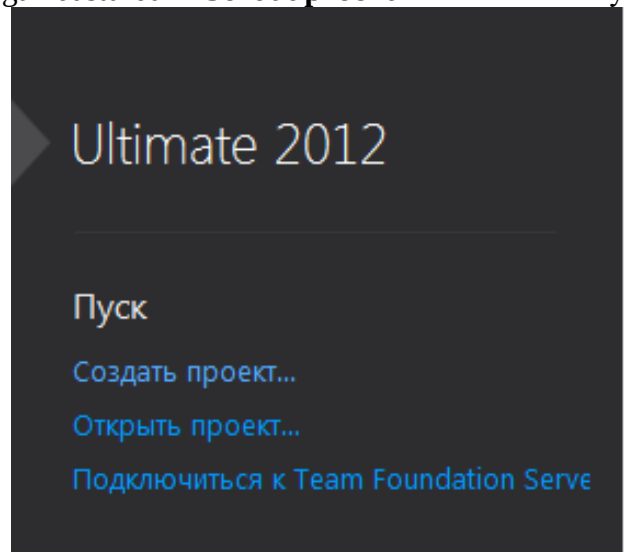
1-Amaliy mashg'ulot

Mavzu: Umumlashtirish.Umumlashgan metodlar yaratish.

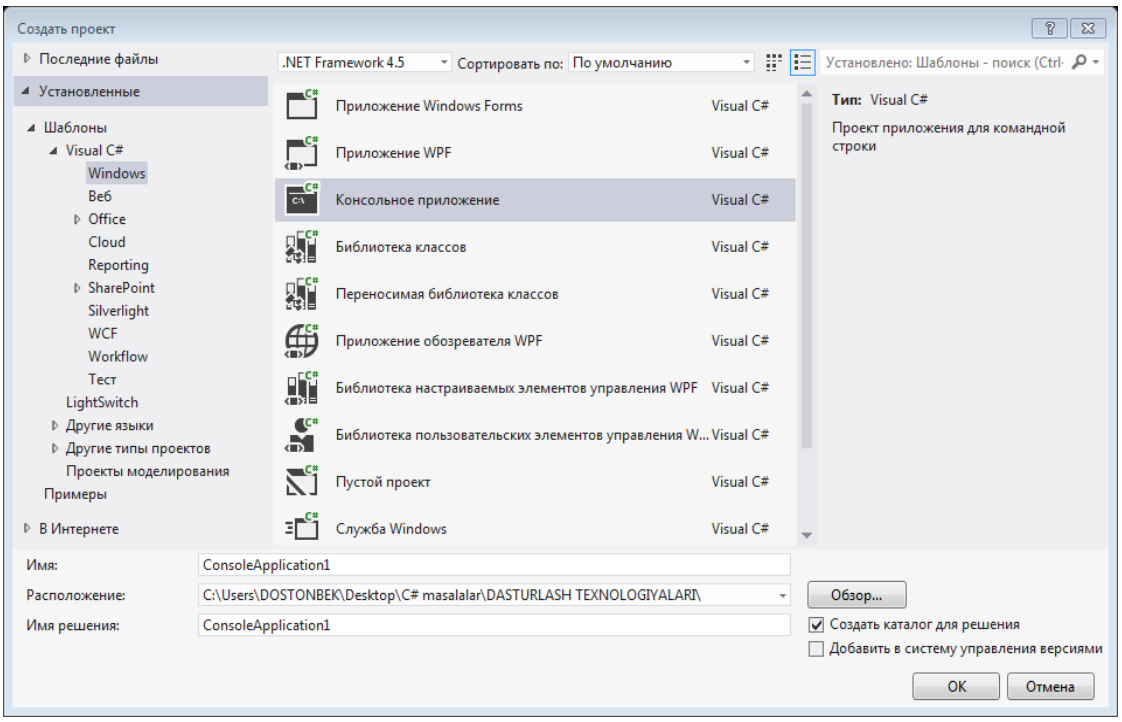
S# dasturni ishga tushiramiz:



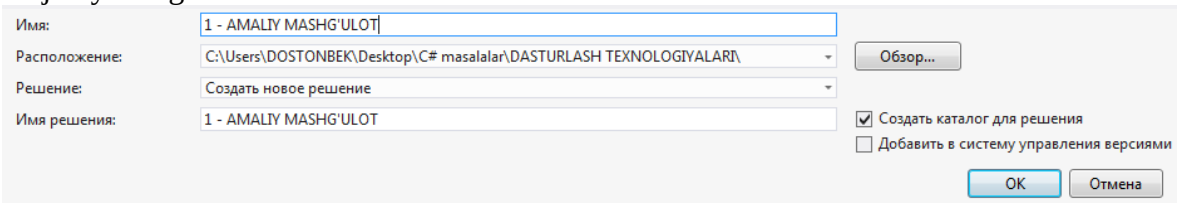
S# dasturni ishga tushirgan dasturdan **"Sozdat proekt"** bo'limini tanlaymiz:



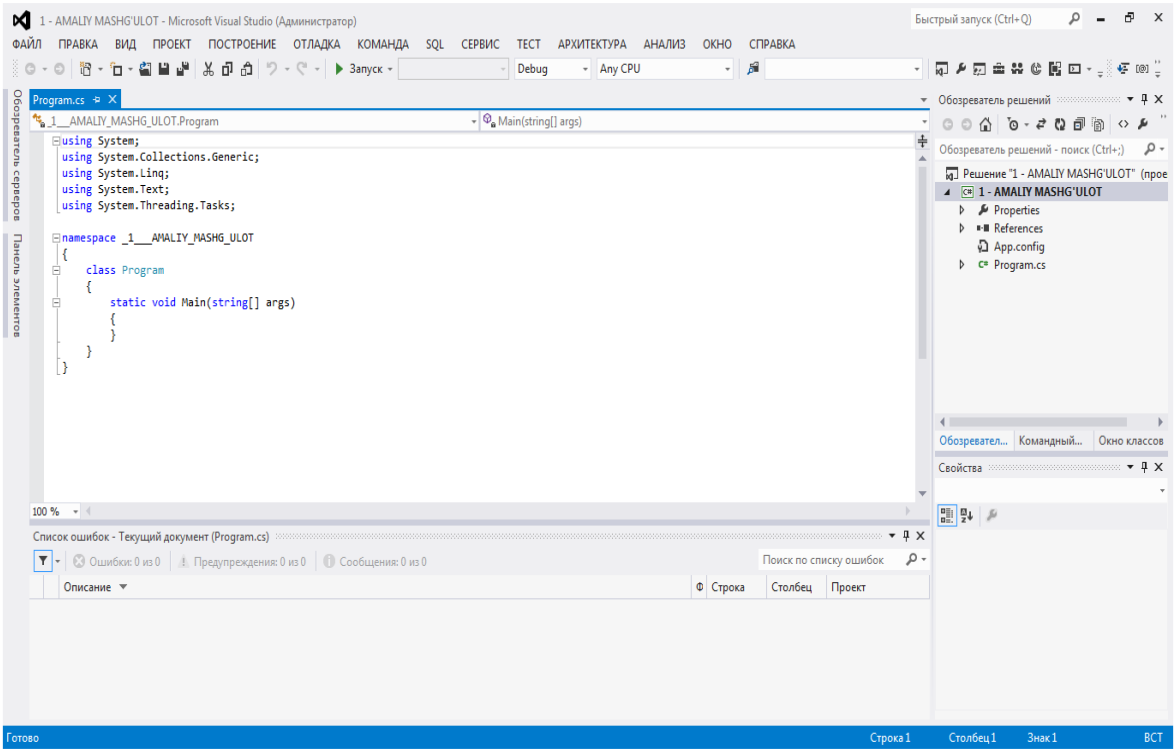
S# dasturni ishga tushirgan dasturdan **"Sozdat proekt"** bo'limini tanlaganimizdan keyin, **Visual C# -> Windows->"Konsolnoe pridlojeniya"** bo'limini tanlaymiz:



”Konsolnoe pridlojeniya” bo’limini tanlaymizdan keyin “Imya” bo’limiga o’tib Konsol pridlojeniyamizga nom beramiz:



va **“OK”** tugmasini tanlaymiz:

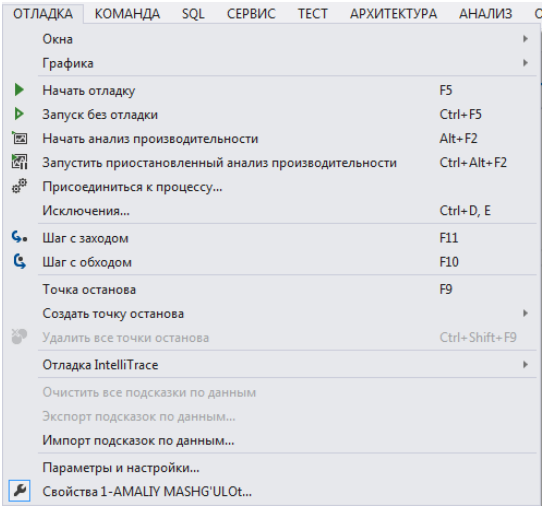


1-Misol: ArrayList klass yordamida massiv elementlarni boshqarish dasturini tuzing.

```
using System;
using System.Collections.Generic;
```

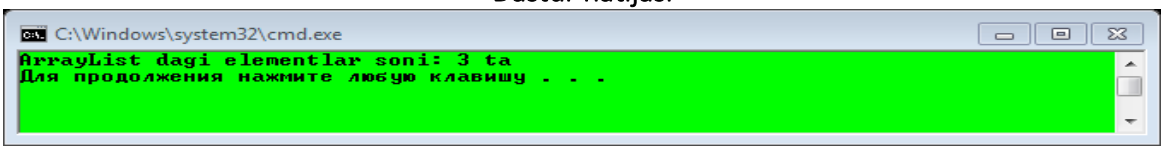
```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
namespace _1_AMALIY_MASHG_ULOT
{
   class Program
   {
      static void Main(string[] args)
      {
         ArrayList alist q new ArrayList();
         alist.Add("Bir");
         alist.Add("Ikki");
         alist.Add("To'rt");
         Console.WriteLine("ArrayList dagi elementlar soni: " Q alist.Count Q " ta ");
      }   }   }
```

Menyular satridan “OTLADKA” bo’limiga o’tamiz:



Bu bo’limdan **“Zapusk bez otladki”** yoki klaviaturadan **“CtrlQF5”** tugmalarini birgalikda bosib, dasturni kompilyatsiya qilamiz.

Dastur natijasi

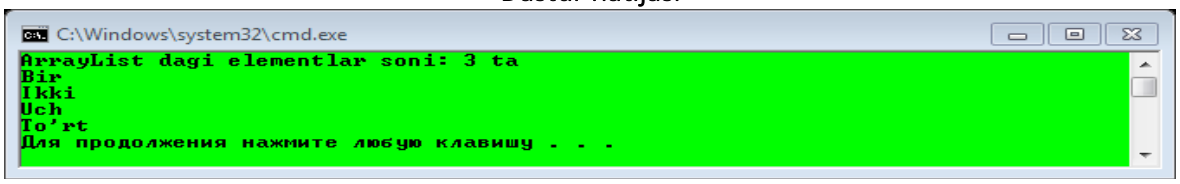


2-Misol: ArrayList klass yordamida massiv elementlariga biror elementni qo’shish dasturini tuzing.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
namespace _1_AMALIY_MASHG_ULOT
{
   class Program
   {
      static void Main(string[] args)
```

```
    {
      ArrayList alist q new ArrayList();
      alist.Add("Bir");
      alist.Add("Ikki");
      alist.Add("To'rt");
      Console.WriteLine("ArrayList dagi elementlar soni: " Q alist.Count Q " ta ");
      G'G'ArrayList dagi elementlar soniga yangi element kiritish
      alist.Insert(2, "Uch");
      foreach (string item in alist)
      { Console.WriteLine(item); }
    }  }  }
```

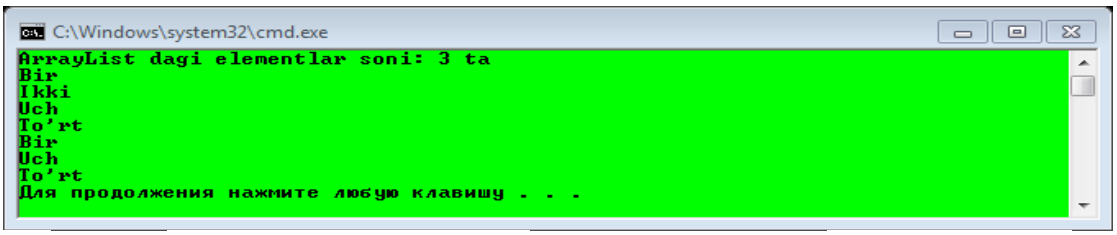
Dastur natijasi



3-Misol: ArrayList klass yordamida massiv elementlariga qo'shilgan elementni o'chirish dasturini tuzing.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
namespace _1_AMALIY_MASHG_ULOT
{
  class Program
  {
    static void Main(string[] args)
    {
      ArrayList alist q new ArrayList();
      alist.Add("Bir");
      alist.Add("Ikki");
      alist.Add("To'rt");
      Console.WriteLine("ArrayList dagi elementlar soni: " Q alist.Count Q " ta ");
      G'G'ArrayList dagi elementlar soniga yangi element kiritish
      alist.Insert(2, "Uch");
      foreach (string item in alist)
      { Console.WriteLine(item); }
      G'G'ArrayList dagi elementdan ikkinchi elementni o'chirish
      alist.RemoveAt(1);
      foreach (string item in alist)
      { Console.WriteLine(item); }
      G'G'ArrayList dagi elementlarni tozalash
      alist.Clear();
      foreach (string item in alist)
      { Console.WriteLine("G'ArrayList dagi elementlarni tozalash"); }
    }  }  }
```

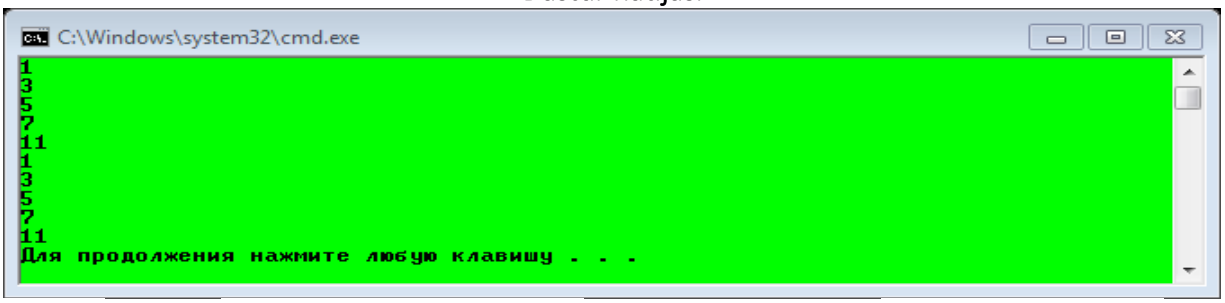
Dastur natijasi



4-Misol: List<int> klass yordamida foreach va list klasslari yordamida elementni qo'shish dasturini tuzing.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
namespace _1_AMALIY_MASHG_ULOT
{
   class Program
   {
      static void Main(string[] args)
      {
         List<int> list q new List<int>();
         G'G'Yangi ro'yhat e'lon qilish (o'lchami oldindan ko'rsatilgan):
         List<int> list2 q new List<int>(10);
         G'G'Ro'yhat ohiriga yangi element qo'shish:
         list.Add(1);        list.Add(3);        list.Add(5);        list.Add(7);        list.Add(11);
         foreach (int son in list)
         { Console.WriteLine(son); }        for (int i q 0; i < list.Count; iQQ)        {
Console.WriteLine(list[i]); }
      }   }   }
```

Dastur natijasi

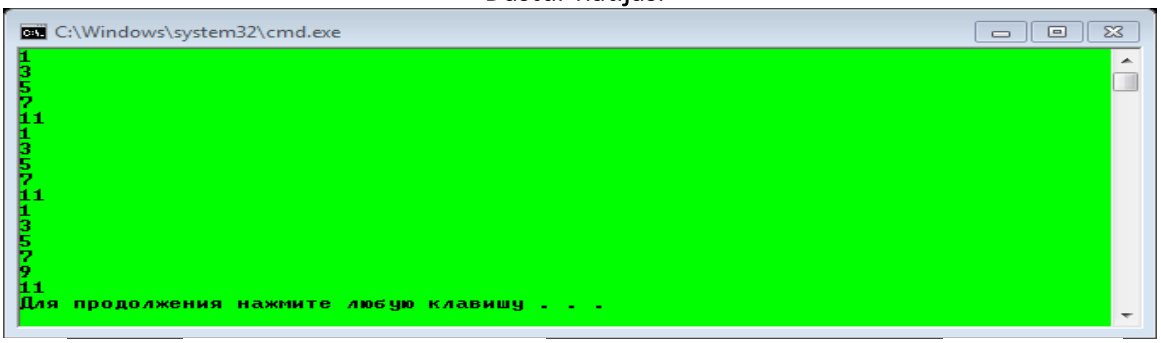


5-Misol: List<int> klass yordamida foreach va list klasslari yordamida element orasiga yangi elementni qo'shish dasturini tuzing.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
namespace _1_AMALIY_MASHG_ULOT
{
   class Program
   {
```

```
        static void Main(string[] args)
        {
          G’G’Yangi ro'yhat e'lon qilish:
           List<int> list q new List<int>();
           G’G’Yangi ro'yhat e'lon qilish (o'lchami oldindan ko'rsatilgan):
           List<int> list2 q new List<int>(10);
           G’G’Ro'yhat ohiriga yangi element qo'shish:
           list.Add(1);        list.Add(3);        list.Add(5);        list.Add(7);        list.Add(11);
           foreach (int son in list)
           { Console.WriteLine(son); }
           for (int i q 0; i < list.Count; iQQ)
           { Console.WriteLine(list[i]); }
         G’G’Ro'yhat orasiga yangi elementni kiritish (indeks yordamida):
           list.Insert(4, 9);
           foreach (int son in list)        { Console.WriteLine(son); }      }    }  }
```

Dastur natijasi

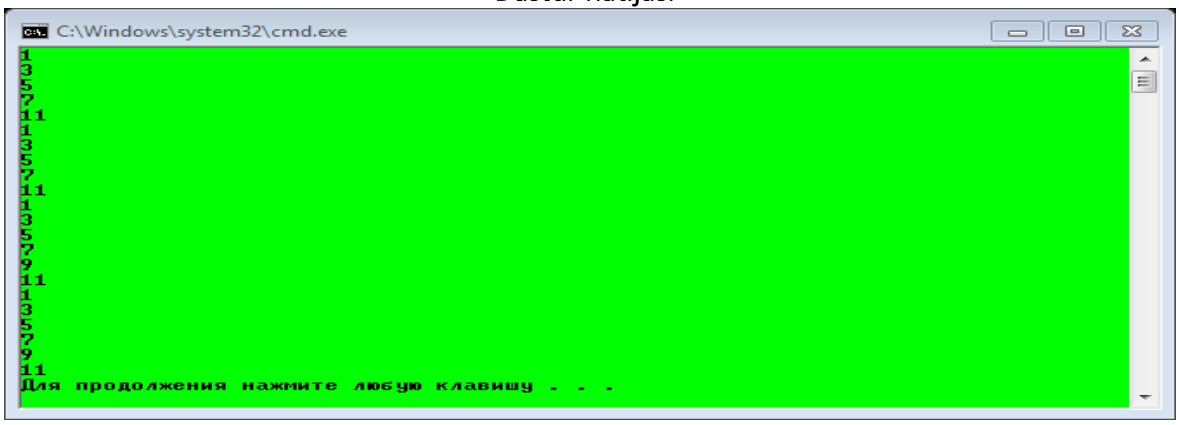


6-Misol: List<int> klass yordamida foreach va list klasslari yordamida ro’yhat elementlarnin sortirovka qilish dasturini tuzing.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;
namespace _1_AMALIY_MASHG_ULOT
{
   class Program
   {
      static void Main(string[] args)
      {
        G’G’Yangi ro'yhat e'lon qilish:
         List<int> list q new List<int>();
         G’G’Yangi ro'yhat e'lon qilish (o'lchami oldindan ko'rsatilgan):
         List<int> list2 q new List<int>(10);
         G’G’Ro'yhat ohiriga yangi element qo'shish:
         list.Add(1);        list.Add(3);        list.Add(5);        list.Add(7);        list.Add(11);
         foreach (int son in list)
         { Console.WriteLine(son); }
         for (int i q 0; i < list.Count; iQQ)
         { Console.WriteLine(list[i]); }
       G’G’Ro'yhat orasiga yangi elementni kiritish (indeks yordamida):
```

```
    list.Insert(4, 9);
    foreach (int son in list)
    { Console.WriteLine(son); }
    G’G’Ro'yhat elementlarini sortirovka qilish (indeks yordamida):
    list.Sort();
    foreach (int son in list)
    { Console.WriteLine(son); }
}   }   }
```

Dastur natijasi



2-Amaliy mashg`ulot

Mavzu: System Collections.Generic nomlar fazosi bilan ishlash.

Malumotlar to’plamini saqlash uchun massivdan boshqa umumlashmalar ham ishlatilishi mumkin. Bunda **ArrayList, Hashtable, List, Queue, Stack, Dictionary** kabi klaslarni keltirishimiz mumkin.

Umumlashmalardan foydalanish uchun **System.Collections**; va **System.Collections.Generic**; nomlar makonini elon qillib qo’yish kerak.

Arraylist o’zgaruvchan umumlashmalar

E’lon qilish va element qo’shish

Arraylist alist=new ArrayList();

Alist.Add(“bir”);
Alist.Add(“ikki”);

Elementlar sonini aniqlash

Cosol.Writeline(“e soni:” +alist.count);

Element qo’shish

Alist.Insert(2, “Uck”);

```
Foreach(string item in alist)
{
Console.Writeline(iteam);
}
```

Elementni o'chirish

```
Alist.Removeat(1);

Foreach(string iteam in alist);
{
Console.Writeline(iteam);
}
```

Tozalash

```
Alist.clear();
Foreach (string item in alist)
{
Console.Writeline(item);
}
```

List<T> umumlashmasi

Yangi ro'yxatni e'lon qilish

```
List<int> list=new List<int>();          1 usul
List<int> list=new List<int>(10);        2 usul


List.add(1);
List.add(3);
List.add(5);
List.add(7);
List.add(11);

Forech (int son in list);
{
Consol.writeline(son);
}
```

For operatori yordamida

```
For (int i=0; i<list.cont; i++);
{
Consol.writeline(list [i]);
}


List.insert(4, 9);

Foreach (int son in list)
{
```

```
Console.writeline(son);
}
```

Sartirovka qilish

```
List.sort();
Foreach(int son in list);
{
Console.writeline(son);
}
```

Berilgan element mavjudligini aniqlash

```
If (list.contains(2))
Console.writline("2 bor");
Else
Console.writeline("2 yoq");
```

Ro'yxatdan elementni o'chirish

```
List.Remove(3);
Foreach (int son in list)
{
Console.Writeline(son);
}
```

Bir nechta elementni o'chirish

```
List.RemoveRange(2, 2);
Foreach (int son in list)
{
Console.Writeline(son);
}
```

Massivga aylantirish

```
List<string> shaxar=new list<string>();
Shaxar.add("Toshkent");
Shaxar.add("tokio");

String line=string.join(",", shaxar.ToArray());
Console.Writeline(line);
```

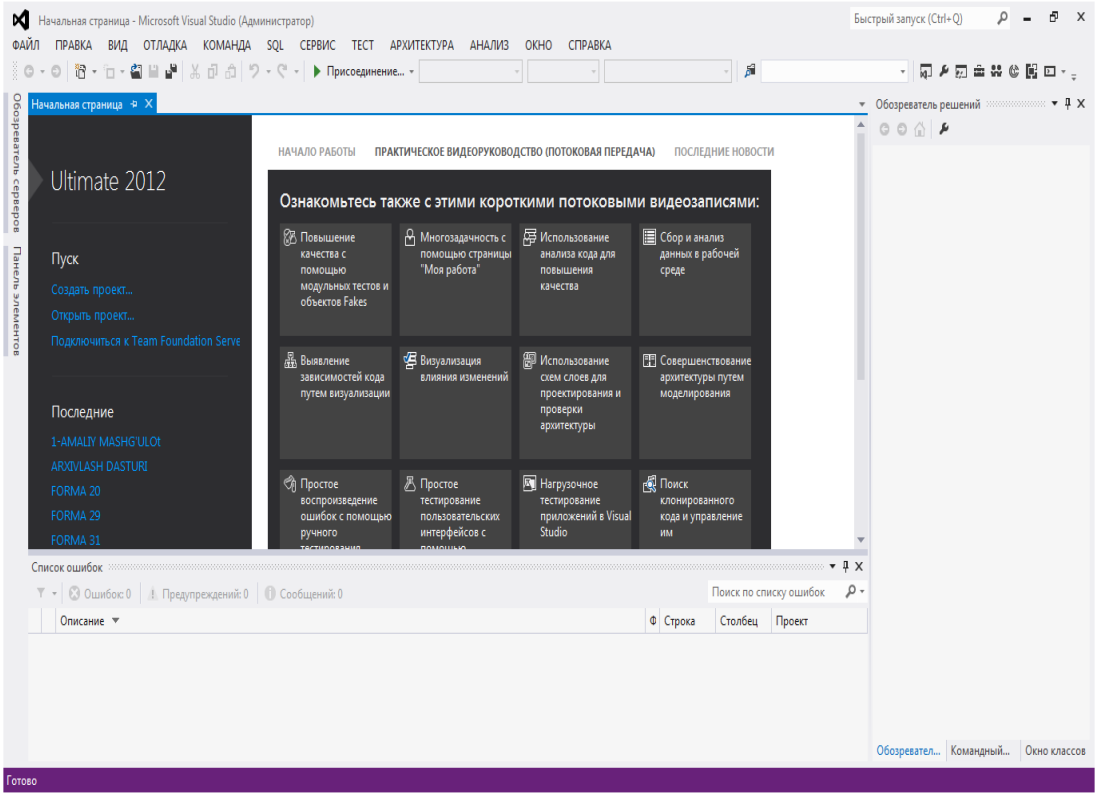
Element indeksini toppish

```
Int indeks1=shaxar.IndexOf("Toshkent");
Console.writeline(indeks1);


Int indeks2=shaxar.BinarySearch("tokio");
Consol.writeline(indeks2);
```

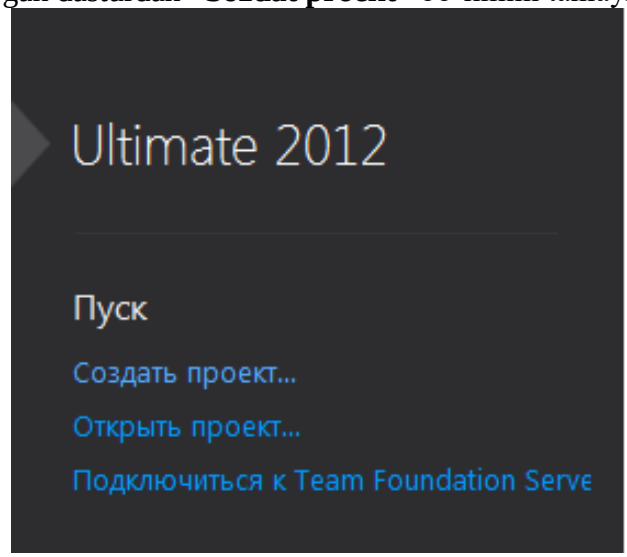
3-Amaliy mashg'ulot

Mavzu: Umumlashgan sinflar yaratish

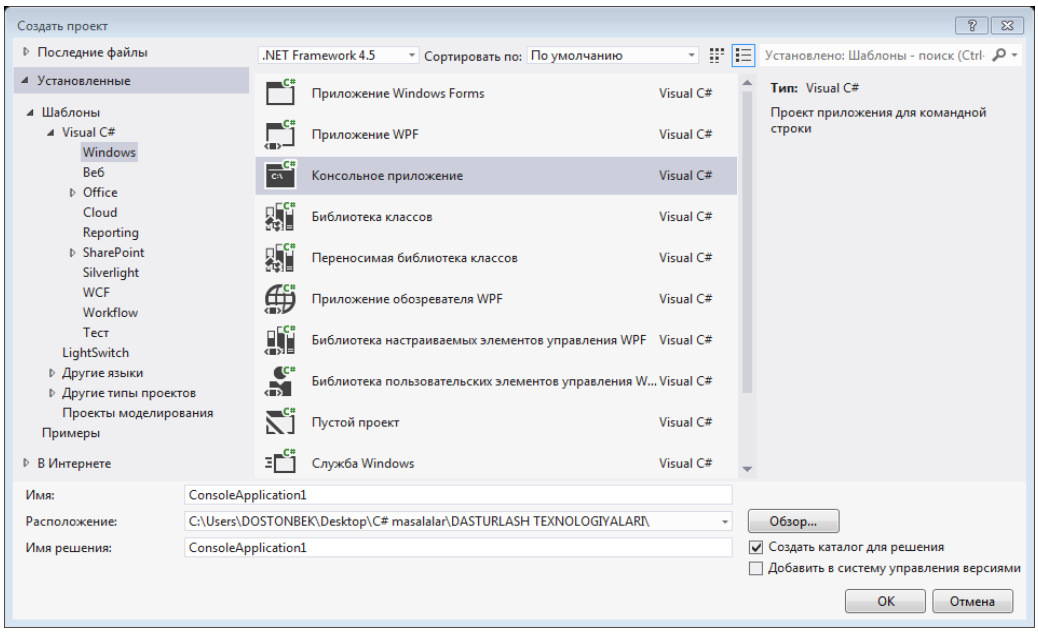
S# dasturni ishga tushiramiz:



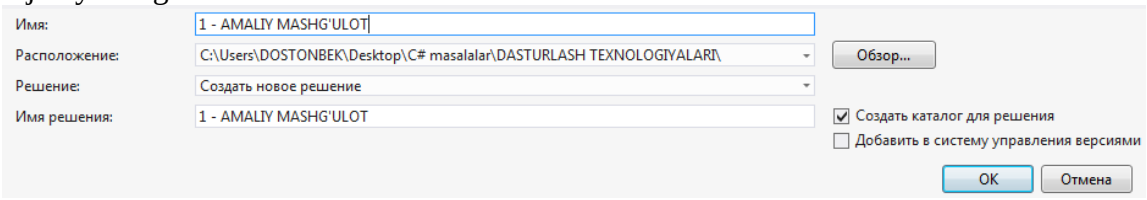
S# dasturni ishga tushirgan dasturdan **"Sozdat proekt"** bo'limini tanlaymiz:



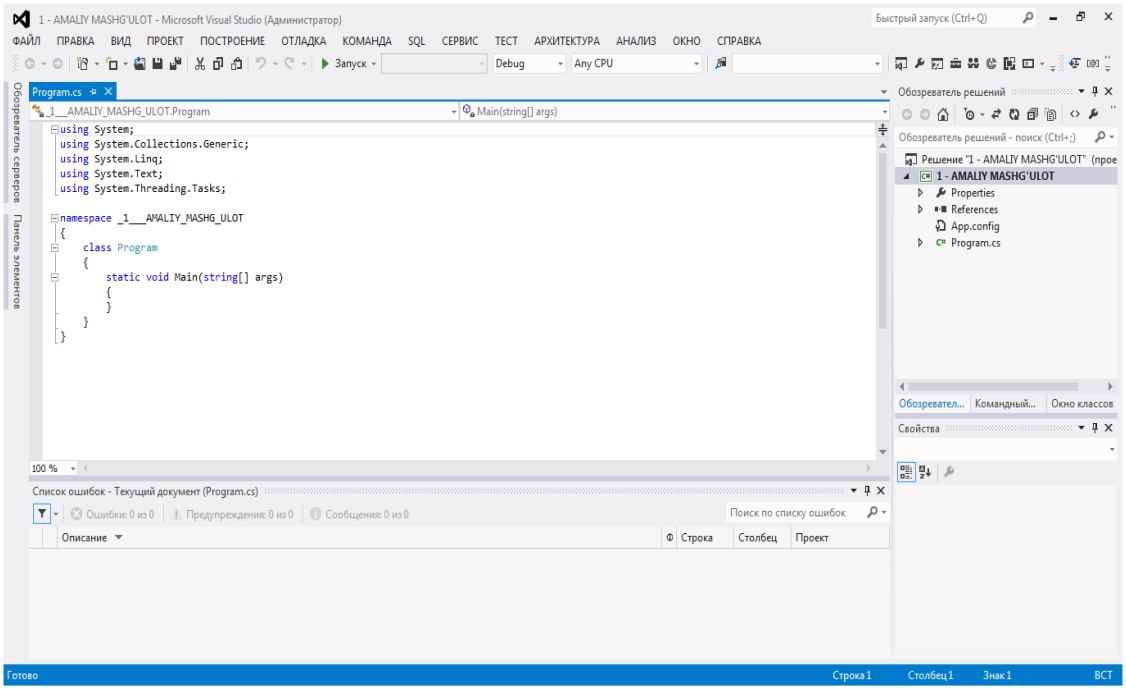
S# dasturni ishga tushirgan dasturdan **"Sozdat proekt"** bo'limini tanlaganimizdan keyin, **Visual C# -> Windows->"Konsolnoe pridlojeniya"** bo'limini tanlaymiz:



”Konsolnoe pridlojeniya” bo’limini tanlaymizdan keyin “Imya” bo’limiga o’tib Konsol pridlojeniyamizga nom beramiz:



va **“OK”** tugmasini tanlaymiz:



Mustaqil ta'lim uchun tavsiya etiladigan mavzular:

1. O’zbekistonda AKT sohasini rivojlantirishdagi qonunlar, farmonlar va qarorlar
2. Zamonaviy kompyuterlarning tarixi, ishlash prinsiplari va arxitekturasi
3. Shaxsiy kompyuterlarning appart qismlari
4. Shaxsiy kompyuterlarning dasturiy ta’minoti
5. Office paketi dasturlarining oxirgi yangi versiyalarining imkoniyatlari
6. Kompyuter viruslari va antivirus dasturlari
7. Ma’lumotlar bazasi.
8. So’rovlar. SQL so’rovlar yaratish.
9. So’rovlar, shakllar va hisobotlar tashkil qilish
10. Kompyuter tarmog'i va ulaming turlari.
11. Formalar bilan ishlash. Forma elementlari xossalari bilan ishlash
12. HTML asoslari.
13. Respublikamizda davlat organlari tomonidan taqdim etilayotgan interaktiv xizmatlar va ularning ahamiyati.
14. Elektron ta'lim tizimlari.
15. Taqdimot yaratuvchi dasturlar bilan ishlash. Prezi dasturi.
16. ABBYY FineReader dasturi imkoniyatlari.
17. Xorijiy ta'limiy ma'lumotlar bazalaridan foydalanish.
18. Dasturlash tillari klassifikasiyasi
19. Dasturlash tillari.O’zgaruvchilar va ularning tiplar.
20. Ifodalar va operatorlar. Arifmetik amallar va munosabat operatorlari
21. Shart operatorlari. Ichma-ich joylashgan if-else operatorlari
22. Sikl operatorlari. For sikli, while sikli, do while sikli
23. Bir o’lchovli va ko’p o’lchovli massivlar
24. Ob’ektga yo’naltirilgan dasturlash
25. Pyhton dasturlash tili.
26. PHP tili va unda web sahifalar yaratish.
27. Dreamweaver dasturi va uning imkoniyatlari.
28. Java dasturlash tili.
29. ActionScript dasturlash tili.
30. MyTestX dasturida nazorat testlari yaratish.
31. Adobe Flash dasturi va uning imkoniyatlari

Test savollari

1. Komputer zahiralarini va amaliy dasturlarni boshqaruvchi, komputer bilan foydalanuvchi o‘rtasidagi “muloqot”ni ta’minlab beruvchi dasturlar majmuasi …deyiladi.
*A) Operatsion tizim;
B) Amaliy dasturiy ta’minot;
C) Informatsion texnologiya;
D) Dasturlash tizimi.
2. Kompyuter bilan ishlashda foydalanuvchiga qo‘shimcha xizmatlarni taqdim etadigan va operatsion tizimning imkoniyatini oshirishga qaratilgan dasturiy mahsulotlar yig‘indisiga … deyiladi.
A) Amaliy dasturlar;
B) Translyatorlar;
*C) Servis dasturlar;
D) Texnik xizmat ko‘rsatish dasturlari.
3. Tizimli dasturiy ta’minot tarkibiga kiruvchi dasturlar qaysi qatorda bexato berilgan?
*****A) operatsion tizimlar, servis dasturlar;
B) operatsion tizimlar, MS Offise paketi;
C) MS Paint, translyator dasturlar;
D) Javoblarning barchasi to‘g‘ri.
4. Boshqa fayllar haqida ma’lumot saqlanadigan maxsus faylga … deyiladi:
*A) Katalog;
B) Kengaytmali fayl;
C) fayl;
D) To‘g‘ri javob yo‘q.
5. MSDOS da katalogni o‘chirishning asosiy shartlari qaysi qatorda to‘g‘ri ko‘rsatilgan?
A) O‘chirilayotgan katalogda fayllar va qism kataloglar mavjud bo‘lmasligi kerak;
B) O‘chirilayotgan katalog joriy bo‘lmasligi lozim;
C) O‘chirilayotgan katalog bo‘sh bo‘lmasligi kerak;
*D) A) va B) shartlar bajarilganda.
6. Tasqi qurilmada saqlanishi uchun ma’lum nom bilan disk fazosining muayyan qismini band etgan, mantiqan bir-biriga bog‘langan axborotga … deyiladi:
A) Katalog;
*B) Fayl;
C) Axborot;
D) Dastur.
7. Magnit diskda jamlovchilar qanday qismlardan tashkil topgan:
A) Diskovod kontrollyori va xususiy diskovod;
B) Interfeysli kabel;
C) Magnit disk;
*D) A) B) va C) dan tashkil topgan.
8. Qaysi qatorda buyruq protsessorining nomi to‘g‘ri ko‘rsatilgan?
A) MSDOS.SYS;
*B) COMMAND.COM;
C) BOOT.RECORD;
D) IO.SYS.
9. Diskning nolinchi sektorida saqlanuvchi, IO.SYS va MSDOS.SYS larni tizimli diskdan operativ xotiraga yuklovchi dastur-fayl nomi qaysi qatorda to‘g‘ri berilgan?

*A) BOOT.RECORD;
B) COMMAND.COM;
C) CONFIG.SYS;
D) BIO.COM

10. DOS modullarini hamda DOSning imkoniyatlarini kengaytiruvchi servis dasturlarni o‘zida mujassamlashtirgan va yuklovchi vazifasini bajaruvchi disk aynan qanday nomlanadi?

A) Tashqi axborot tashuvchilar;
*B) Tizimli disk;
C) Floppi disk;
D) Oddiy disk.

11. Har bir tizimli disk fazosining 0-sektorida qaysi fayl joylashadi?

A) IO.SYS;
B) MSDOS.SYS;
*C) BOOT RECORD;
D) COMMAND.COM.

12. Maxsus dastur yordamida disk fazosidagi fayllar o‘rnini uzluksiz sohaga keltirish jarayoniga diskni … deyiladi.

A) Formatlash;
B) Tozalash;
C) Fragmentlash;
*D) Defragmentlash.

13. Kompyuterda muayyan dastur bajarilishi vaqtida chaqiruv orqali yuzaga kelgan boshqa protseduraning bajarilishi uchun joriy protseduraning qisqa muddatli to‘xtab qolish jarayoniga … deyiladi:

*A) Uzilish;
B) Qayta yuklash;
C) Kompyuterni o‘chirish;
D) To‘g‘ri javob yo‘q.

14. Kompyuter qurilmalaridan keladigan signallar orqali hosil bo‘ladigan uzilishlarga … deyiladi:

A) Dasturiy uzilishlar;
B) Mantiqiy uzilishlar;
*C) Apparat uzilishlar;
D) Javoblarning barchasi to‘g‘ri.

15. Nolga bo‘lish, registrlarning to‘lishi va shunga o‘xshash aniq nostandart holatlarning yuzaga kelishidan hosil bo‘ladigan uzilishlarga … deyiladi:

A) Dasturiy uzilishlar;
*B) Mantiqiy uzilishlar;
C) Apparat uzilishlar;
D) Javoblarning barchasi to‘g‘ri.

16. Faqat dasturlar tomonidan, ya’ni bir dasturning boshqa dastur tomonidan xizmat ko‘rsatishi talab qilinganda yuzaga keladigan uzilishlarga … deyiladi:

A) Mantiqiy uzilishlar;
B) Zahiraviy uzilishlar;
*C) Dasturiy uzilishlar;
D) Javoblarning barchasi to‘g‘ri.

17. Zahiraviy (apparat) uzilishlar qanday saviyadagi uzilishlar turkumiga kiradi va ularga qanday tartib raqamlari berilgan?

*A) Quyi saviyadagi uzilish, 0 – 1F;
B) Yuqori saviyadagi uzilish, 20 – 2F;
C) Yuqori saviyadagi uzilish, 30 – 57;

D) Quyi saviyadagi uzilish, 0 – FF.

18. Uzilish vektori deganda nima tushuniladi?

*A) Uzilishlarga xizmat ko‘rsatuvchi dastur adreslari;

B) Uzilish drayverlari;

C) Uzilishlarga xizmat qiluvchi DOS funksiyalari;

D) Javoblarning barchasi to‘g‘ri.

19. Shaxsiy uzilishlarni yozishga zaruriyat tug‘diradigan sabablar qaysi qatorda to‘g‘ri ko‘rsatilgan?

A) Dasturchi tomonidan operatsion tizimning dasturlar kutubxonasiga o‘z dasturini qo‘shib qo‘yish;

B) Yangi bir zahiraviy uzilishni qurish;

C) Yangi qurilmalar drayverini o‘rnatish yoki olib tashlash;

*D) A) va B) javoblar to‘g‘ri.

20. Bajarilgandan keyin ham tezkor xotirada saqlanib turiladigan xizmat ko‘rsatuvchi dasturlarga … deyiladi.

A) Drayverlar;

*B) Rezident dasturlar;

C) Overleylar;

D) Amaliy dasturlar.

21. Rezident dasturlarning diskda saqlanadigan qismlariga … deyiladi.

A) Qism dastur;

B) Yuklanuvchi dasturlar;

*C) Dastur overleylari;

D) Qobiq dasturlar.

22. Rezident dasturlarni odatda qanday kengaytmali fayl sifatida yozish maqsadga muvofiq keladi?

*A) COM kengaytmali fayl sifatida;

B) EXE kengaytmali fayl sifatida;

C) SYS kengaytmali fayl sifatida;

D) INI kengaytmali fayl sifatida.

23. DOS tarkibida dasturni rezident holatiga keltiradigan nechta uzilish funksiyasi mavjud?

A) 4 ta;

B) 1 ta;

C) 3 ta;

*D) 2 ta;

24. Amaliy dasturlarning tashqi qurilmalarga bo‘lgan ehtiyojini qondirish maqsadida chaqiriladigan IBMBIO.COM tarkibidagi drayverlar qanday nomlanadi?

A) Yuklanuvchi drayverlar;

*B) Rezident drayverlar;

C) Kengaytirilgan xotirani boshqaruvchi drayverlar;

D) Norezident drayverlar.

25. DOS tomonidan BIOS ni o‘zgartirmay turib, CONFIG.SYS faylidagi DEVICE buyrug‘i orqali tizimga qo‘shiladigan drayverlar:

A) Rezident drayverlar;

B) Kengaytirilgan xotirani boshqaruvchi drayverlar;

C) Standart drayverlar;

*D) Yuklanuvchi norezident drayverlar deyiladi.

26. TSR dasturlar qanday hosil qilinadi?

*A) DOS 21H uzilishlardagi 31H funksiyani chaqirish va shu orqali dasturni tugatish yo‘li bilan;

B) BIOS 13H uzilishdagi 11H funksiyani chaqirish yo‘li bilan;
C) DOS 21H uzilishlardagi 33H funksiyani chaqirish yo‘li bilan;
D) To‘g‘ri javob yo‘q.

27. TSR dasturlar asosan qanday turlarda mavjud bo‘ladi va qanday nomlanadi?
A) Faqat 1 turda, xizmat ko‘rsatuvchi dasturlar;
B) Faqat 1 turda, rezident utilitalar;
C) 2 turda, xizmat ko‘rsatuvchi dasturlar va rezident utilitalar;
D) To‘g‘ri javob yo‘q.

28. TSR dasturlarni qurishga qo‘yiladigan talablar odatda nimalardan iborat bo‘ladi?
A) Dastur juda katta hajmga ega bo‘lmaslik kerak;
B) Dastur tez va bevosita bajarilishi hamda bajarilishdan oldin o‘zini resident deb e’lon qilishi kerak;
C) Kuniga bir necha marta ishlatiladigan dastur bo‘lishi kerak;
*D) Javoblarning barchasi to‘g‘ri.

29. TSR dasturlarning bajarilishini to‘xtatish jarayoni qanday amallarni o‘z ichiga oladi?
A) Dasturda uning to‘xtalishini e’lon qilish;
B) Uzilish vektorlarini uning boshlang‘ich holatiga qaytarish;
C) Dasturda ochilgan barcha fayllarni yopish va dastur egallab turgan xotira qismini DOS ga qaytarish;
*D) Yuqoridagi amallar ketma-ket tartibda bajarilishi shart.

30. Muayyan operatsion tizimga muvofiq standart tashqi qurilmalarning ishlashini ta’minlovchi dasturlar … deyiladi.
A) Texnik xizmat dasturlari;
*B) Drayverlar;
C) Tashqi qurilmalar ish faoliyatini tekshiruvchi dasturlar;
D) Operatsion tizimning boshlang‘ich yuklash dasturlari.

31. Kompyuterning tashqi qurilmalarida saqlanadigan, tashqi qurilmalarni boshqaradigan, operatsion tizimga CONFIG.SYS fayli orqali qo‘shiladigan drayverlar … deyiladi.
A) Asosiy xotirani boshqaruvchi drayverlar;
B) Standart drayverlar;
C) Kengaytirilgan xotirani boshqaruvchi drayverlar;
*D) Yuklanuvchi drayverlar.

32. BIOS yoki uning kengaytmasi tarkibiga kiradigan, tashqi qurilmalarni boshqarishda qo‘llaniladigan, tizimga avtomatik ravishda qo‘shiladigan dasturlar:
A) Operatsion tizim;
B) Servis dasturlar;
*C) Standart drayverlar;
D) Yuklanuvchi drayverlar.

33. Stek – bu:
A) 16 ga qoldiqsiz bo‘linadigan chegaradan boshlanuvchi 16 baytli xotira birligi;
*B) Dastur tomonidan qiymatlarni vaqtincha saqlash uchun ishlatidadigan xotira qismi;
C) Segment registri bilan bog‘liq bo‘lgan adreslarni o‘rnatish jarayoni;
D) Javoblarning barchasi to‘g‘ri.

34. Bog‘lanish – bu:
A) 16 ga qoldiqsiz bo‘linadigan chegaradan boshlanuvchi 16 baytli xotira birligi;
B) Dastur tomonidan qiymatlarni vaqtincha saqlash uchun ishlatidadigan xotira qismi;
*C) Segment registri bilan bog‘liq bo‘lgan adreslarni o‘rnatish jarayoni;
D) Javoblarning barchasi to‘g‘ri.

35. Paragraf – bu:

A) *16 ga qoldiqsiz bo‘linadigan chegaradan boshlanuvchi 16 baytli xotira birligi;
B) Dastur tomonidan qiymatlarni vaqtincha saqlash uchun ishlatidadigan xotira qismi;
C) Segment registri bilan bog‘liq bo‘lgan adreslarni o‘rnatish jarayoni;
D) Javoblarning barchasi to‘g‘ri.

36. Qaysi tipdagi dasturlar MS DOS ning asosiy bajariluvchi dasturlariga kiradi?
A) Faqat COM tipidagi dasturlar;
B) Faqat EXE tipidagi dasturlar;
*C) COM va EXE tipidagi dasturlar;
D) Javoblarning barchasi to‘g‘ri.

37. MS DOS da COM kengaytmali fayllarning eng katta hajmi qanchagacha bo‘lishi mumkin?
*A) 64 Kb gacha;
B) 64 Kb dan yuqori;
C) 68 Bayt;
D) To‘g‘ri javob berilmagan.

38. Biror algoritmik tilda yozilgan boshlang‘ich modulni unga mos obyektli modulga o‘tkazadigan dastur … deyiladi
A) Operatsion tizim;
B) Kompilyator;
*C) Translyator;
D) To‘g‘ri javob yo‘q.

39. Kompilyator deyiladi, agar tarjimon dastur:
A) Assembler tilida yozilgan dasturlarni tarjima qilishga mo‘ljallangan bo‘lsa;
*B) Yuqori darajali algoritmik tilda yozilgan dasturni tarjima qilishga mo‘ljallangan bo‘lsa;
C) Dasturni hosil qilish jarayonida tarjima qilishga mo‘ljallangan bo‘lsa;
D) To‘g‘ri javob yo‘q.

40. Interpretator deyiladi, agar tarjimon dastur:
A) Assembler tilida yozilgan dasturlarni tarjima qilishga mo‘ljallangan bo‘lsa;
B) Yuqori darajali algoritmik tilda yozilgan dasturni tarjima qilishga mo‘ljallangan bo‘lsa;
*C) Dasturni hosil qilish jarayonida tarjima qilishga mo‘ljallangan bo‘lsa;
D) To‘g‘ri javob yo‘q.

41. Assemblerlar deyiladi, agar tarjimon dastur:
*A) Assembler tilida yozilgan dasturlarni tarjima qilishga mo‘ljallangan bo‘lsa;
B) Yuqori darajali algoritmik tilda yozilgan dasturni tarjima qilishga mo‘ljallangan bo‘lsa;
C) Dasturni hosil qilish jarayonida tarjima qilishga mo‘ljallangan bo‘lsa;
D) To‘g‘ri javob yo‘q.

42. Translyator uchun o‘tish tushunchasi – bu:
*A) Dastlabki dastur matnini biror-bir ko‘rinishda to‘liq tahlil qilib chiqish;
B) Dastur matnining bir blokida hech qanday funksiya bajarmasdan ikkinchi blogiga o‘tish;
C) Dasturni tekshirmasdan kompilyatsiyaga o‘tish;
D) Javoblarning barchasi to‘g‘ri.

43. Kiritilgan dastur matnini boshidan oxirigacha tekshirib, tahlil qilish natijasida obyektli modulni hosil qilish bilan chegaralangan translyatorlarga … deyiladi:
A) Ikki o‘tishli translyatorlar;
*B) Bir o‘tishli translyatorlar;
C) Ko‘p o‘tishli translyatorlar;
D) To‘g‘ri javob berilmagan.

44. Preprotsessor ishidan so‘ng leksik, semantic va sintaktik tahlillar asosida

assembler ko‘rinishidagi obyektli dasturni hosil qiladigan translyatorlarga … deyiladi:
A) Ko‘p o‘tishli translyatorlar;
B) Bir o‘tishli translyatorlar;
*C) Ikki o‘tishli translyatorlar;
D) To‘g‘ri javob berilmagan.
45. Quyidagilardan qaysi biri leksik tahlilchilarning (analizatorlarning) asosiy xususiyatlaridan birini aks ettiradi?
*A) Boshlang‘ich matnni o‘qish jarayonida jadvallarning boshlang‘ich elementlarini hosil qiladi;
B) Qiymatlarni jadvallarga kiritadi va matnning ichki ko‘rinishini hosil qiladi;
C) Biror algoritmik tilda yozilgan boshlang‘ich modulni qabul qiladi va bajaradi;
D) Dastur mudullari orasida bog‘lanishlar o‘rnatadi va ularni adres fazosida yig‘adi.
46. Quyidagilardan qaysi biri semantic tahlilchilarning (analizatorlarning) asosiy xususiyatlaridan birini aks ettiradi?
A) Boshlang‘ich matnni o‘qish jarayonida jadvallarning boshlang‘ich elementlarini hosil qiladi;
*B) Qiymatlarni jadvallarga kiritadi va matnning ichki ko‘rinishini hosil qiladi;
C) Biror algoritmik tilda yozilgan boshlang‘ich modulni qabul qiladi va bajaradi;
D) Dastur mudullari orasida bog‘lanishlar o‘rnatadi va ularni adres fazosida yig‘adi.
47. Quyidagilardan qaysi biri bo‘g‘lanishlar muharririning vazifasiga kiradi?
A) Boshlang‘ich matnni o‘qish jarayonida jadvallarning boshlang‘ich elementlarini hosil qiladi;
B) Qiymatlarni jadvallarga kiritadi va matnning ichki ko‘rinishini hosil qiladi;
C) Biror algoritmik tilda yozilgan boshlang‘ich modulni qabul qiladi va bajaradi;
D) *Dastur mudullari orasida bog‘lanishlar o‘rnatadi va ularni adres fazosida yig‘adi.
48. Agar bog‘lanishlar muharriri obyektli modullarni nisbiy adreslarga sozlasa, u holda hosil bo‘lgan faylga:
*A) Yuklanuvchi modul;
B) Absolyut fayl;
C) Birlashtirishlar xaritasi;
D) Boshlang‘ich modul deyiladi.
49. Axborotni joylashtirishga mo‘ljallangan, diskning magnit sirtidagi konsentrik aylanalardan iborat sohasi nima deb ataladi?
*A) Yo‘l;
B) Klaster;
C) Sektor;
D) Silindr.
50. Eng katta radiusli yo‘l diskda qanday tartib raqami bilan belgilanadi?
A) 39-yo‘l;
*B) 0-yo‘l;
C) 1-yo‘l;
D) 40-yo‘l.
51. Agar P – disketning sirtlari soni, D – disketning bir sirtidagi yo‘laklar soni, S – bitta yo‘lakdagi sektorlar soni bo‘lsa, ushbu P*D*S*512 (bayt) formula nimani aniqlaydi?
*A) Disketning hajmini;
B) Axborot o‘lcjov birligini;
C) Disketning fizik o‘lchamini;
D) Disket plastinkasining diametrini.
52. Ushbu N(sirt) + N(yo‘l) + N(sector) formula orqali disketning qaysi elementi aniqlanadi?

A) Disketning formatlanganlik darajasi;
B) Disket hajmi;
*C) Disket sektorining fizik adresi;
D) To‘g‘ri javob yo‘q.

53. Mantiqiy diskning faylli tuzilmasini xarakterlovchi asosiy elementlar qaysi qatorda to‘g‘ri ko‘rsatilgan?
A) FAT, Disk nomi, Disk hajmi;
*B) Boot-sector, FAT, Root-Directory, Ma’lumotlar sjhasi;
C) Ma’lumotlar sohasi, Kesh-xotira hajmi;
D) Javoblarning barchasi to‘g‘ri.

54. Ushbu N(silindr) + N(yo‘l) + N(sektor) formula orqali nima aniqlanadi?
*A) Vinchester sektorining fizik adresi;
B) Vinchester hajmi;
C) Diskdagi ma’lumotlar sohasi;
D) Tizimli disk holati.

55. BIOS 10h uzilishi asosan qanday xizmatni o‘tashga mo‘ljallangan?
A) Disklarga xizmat ko‘rsatish;
*B) Ekranga xizmat ko‘rsatish;
C) Tizimni qayta yuklash;
D) Klaviatura xizmati.

56. Dastur matnida konstantalar, xizmatchi so‘zlar, identifikatorlar va izohlarning belgilangan qoidalarga ko‘ra to‘g‘ri yozilganligini tahlil qilish jarayoniga … deyiladi:
*A)Leksik tahlil;
B) Sintaktik tahlil;
C) Semantik tahlil;
D) To‘g‘ri javob berilmagan.

57. Dastur matnida operatorlar, amallar, funksiya va o‘zgaruvchilarning to‘g‘ri qo‘llanilganligini tahlil qilish jarayoniga … deyiladi.
A) Leksik tahlil;
*B) Sintaktik tahlil;
C) Semantik tahlil;
D) Barcha javoblar to‘g‘ri.

58. Dastur tuzilishini ma’no va mazmun jihatdan tekshirish jarayoniga … deyiladi:
*A) Semantik tahlil;
B) Sintaktik tahlil;
C) Leksik tahlil;
D) To‘g‘ri javob berilmagan.

59. Dastur matnini to‘liq tahlil qilib chiqish natijasida ichki ko‘rinishdagi elementar amallarni bajarishga asoslangan tranlyatsiya jarayoniga … deyiladi:
*A) Kod generatsiyasi;
B) Yig‘ish;
C) Tizimni tekshirish;
D) To‘g‘ri javob berilmagan.

60. Assemblerlash jarayonidan so‘ng hosil qilingan dastur ko‘rinishiga … deyiladi:
A) Boshlang‘ich modul;
*B) Obyektli modul;
C) Yuklanuvchi modul;
D) Absolyut fayl.

61. Obyektli modul to‘plamidan tashkil topgan faylga … deyiladi:
*A) Obyektli modullar kutubxonasi;
B) Dastur – modul;

C) Amaliy dasturlar kutubxonasi;
D) To‘g‘ri javob berilmagan.
62. Obyektli modul faylining standart kengaytmasi qaysi qatorda to‘g‘ri berilgan?
A) .exe;
*B) .obj;
C) .com;
D) .bat.
63. Alohida fayl ko‘rinishida bajariluvchi dastur-faylning standart kengaytmasi qaysi qatorda to‘g‘ri berilgan?
*A).exe;
B) .bak;
C) .doc;
D) .dot.
64. FAT – bu:
A) Diskning ma’lumotlar sohasi klasterlarini adreslashtiruvchi elamentlar massivi;
B) Diskda fayllarning joylashish jadvali;
C) O‘zak katalog;
*D) A) va B) javoblar to‘g‘ri.
65. O‘zak katalog – bu:
*A) Diskni formatlash jarayonida yaratiladigan, ma’lumot saqlanadigan bosh katalog;
B) FAT ning ikkinchi nusxasi;
C) Ma’lumotlar sohasi (klasterlar);
D) To‘g‘ri javob berilmagan.
66. Quyidagilardan qaysi biri boshlang‘ich yuklash sektori (Boot-sector) tarkibiga kirmaydi?
A) Klasterdagi sektorlar soni;
B) Diskning format tipi;
C) Katalogdagi elementlar soni;
*D) To‘g‘ri javob berilmagan.
67. Biror matn muharriri yordamida foydalanuvchi tomonidan yaratilgan dasturlar fayl sifatida qanday kengaytmaga ega bo‘ladi?
A) .sys;
B) .txt;
*.C) bat;
D) .ovr.
68. Operatsion tizimning konfiguratsiyasini ifodalaydigan fayllar qanday standart kengaytmaga ega bo‘ladi?
A) .me;
B) .bmp;
C) .asm;
*D) .ini yoki .cgf.
69. 32-razryadli Intel mikroprotsessorida nechta tizimli registr mavjud?
*A) 16 ta;
B) 8 ta;
C) 6 ta;
D) 2 ta.
70. 32-razryadli Intel mikroprotsessorida nechta segmentli registr mavjud?
A) 8 ta;
*B) 6 ta;
C) 16 ta;
D) 2 ta.
71. 32-razryadli Intel mikroprotsessorida foydalanuvchining registrlari soni

nechtani tashkil etadi?
A) 6 ta;
B) 2 ta;
*C) 16 ta;
D) Bitta ham yo‘q.
72. 32-bitli registrlar ichida umumiy qiymatli registrlar soni nechta?
A) 10 ta;
B) 16 ta;
C) 6 ta;
*D) 8 ta.
73. Mikroprotsessorning arifmetik-mantiqiy qurilmasida joylashgan registrlar nomi qaysi qatorda to‘g‘ri berilgan?
* A) Umumiy qiymatli registrlar;
B) Segmentli registrlar;
C) Holat registrlari;
D) Tizimli registrlar.
74. Oraliq ma’lumotlarni saqlashda qo‘llaniladigan akkumulyator-registrlarning yozilishi qaysi qatorda to‘g‘ri berilgan?
*A) eax/ax/ah/al;
B) ebx/bx/bh/bl;
C) esi/si;
D) CS.
75. Stek ko‘rsatkichi registri qanday belgilanadi?
A) edi/di;
*B) esp/sp;
C) DS;
D)ebp/bp.
76. Mikroprotsessorda bazaviy registrlar qanday belgilanadi?
A) ebp/bp;
B) eip/ip;
* C) ebx/bx/bh/bl;
D) edx/dx.
77. Qaysi qatorda bayroq registrining nomi to‘g‘ri keltirilgan?
*A) eflags/flags;
B) eip/ip;
C) fs;
D) es.
78. Funksional jihatdan tizimli registrlar nechta guruhga ajratiladi?
A) 8 ta;
* B) 3 ta;
C) 4 ta;
D) 2 ta.
79. Boshqaruvchi registrlar funksional jihatdan qaysi registrlar turkumiga kiradi?
A) Foydalanuvchining registrlari;
* B) Tizimli registrlar;
C) Umumiy qiymatli registrlar;
D) Alohida turdagi registrlar.
80. Mikroprotsessorda rostlovchi registrlar nechta va u qaysi registrlar to‘plamiga kiradi?
*A) 8 ta, tizimli registrlar;
B) 4 ta, foydalanuvchi registrlari;
C) 8 ta, bayroq registrlari;

D) 2 ta, boshqaruvchi registrlar.

81. Mikroprotsessorning himoyaviy rejimda ishlashini ta’minlashga mo‘ljallangan registrlar … deyiladi.

A) Segmentli registrlar;
B) Bazaviy registrlar;
*C) Tizimli registrlar;
D) Umumiy qiymatli registrlar.

82. Mikroprotsessorning holati va ish rejimi haqidagi axborotlarni doimiy saqlab turishga mo‘ljallangan registrlar … deyiladi.

A) Hisobchi registrlar;
B) Rostlovchi registrlar;
C) Ma’lumot registrlari;
*D) Holat va boshqarish registrlari.

83. Sistemani to‘liq boshqarish va nazorat qilishga mo‘ljallangan registrlar … deyiladi.

A) Bayroq registrlari;
B) Holat registrlari;
C) Tizimli adres registrlari;
*D) Boshqarish registrlari.

84. Mikroprotsessorning ko‘p masalali rejimdagi ish faoliyati davomida turli dastur va ma’lumotlarni himoyalashga qaratilgan registrlar … deyiladi:

A) Ma’lumotlarning qo‘shimcha segmenti registrlari;
* B) Tizimli adres registrlari;
C) Stek segmenti registrlari;
D) Baza ko‘rsatkichi registri.

85) MP apparat vositalari bilan bog‘liq registrlar qanday nomlanadi va ular nechta?

* A) Rostlash registrlari, soni 8 ta;
B) Boshqarish registrlari, soni 4 ta;
C) Bayroq registrlari, soni 5 ta;
D) Tizimli registrlar, soni 16 ta;

86) Assembler tilidagi dasturda BIOS yoki DOS modulining biror *n*-funksiyasini chaqirish uchun umumiy holda qanday buyruq ishlatiladi?

A) Inslude *n*;
B) Add *n*;
*C) int *n*;
D) Mov *n*.

87. Elektr manbaiga ulanganda protsessor dastavval ishni qaysi adresdan boshlaydi?

A) 18B2:0100;
* B) FFFF:0000;
C) FFFF:000B;
D) FFFF:01A0;

88. Assemblerda Int 11h buyrug‘i nima uchun ishlatiladi?

* A) Komputer konfiguratsiyasi aniqlashda;
B) Tezkor xotira hajmini aniqlashda;
C) Tizimli soat bilan ishlash uchun;
D) Operatsion tizimni qayta yuklash uchun.

89. BIOS tarkibiga kiruvchi dasturlarni ko‘rsating:

A) POST dasturi;
B) Boshlang‘ich yuklovchi dastur;

C) Kompyuter apparat vositalarini boshqaruvchi dasturlar majmui;
*D) Barcha javoblar to‘g‘ri.
90. Adres – bu:
*A) Xotira yacheykalarining tartiblangan nomeri;
B) Xotiradan tashqari element;
C) Kompyuter ichki xotirasining nomi;
D) To‘g‘ri javob yo‘q.
91. Makroaniqlov – bu:
A) Makros yozishga mo‘ljallangan assembler dasturining bir elementi;
*B) Assemblerda yozilgan makros;
C) Izoh so‘z;
D) Operator.
92. Makrokengaytma – bu:
*A) Assemblerda makroalmashish jarayoni natijasida hosil qilingan dastur matni;
B) Dasturning kengaytirib yozilgan holati;
C) Dasturda ishlatish uchun maxsus buyruq;
D) To‘g‘ri javob yo‘q.
93. Makrobuyruq – bu:
A) Dastur qismida makroaniqlovga murojaat qilish buyrug‘i;
B) Dasturning ko‘rsatilgan joyiga makroaniqlovni qo‘yish uchun makrogeneratorga beriladigan ko‘rsatma;
C) Bir nechta buyruqlar to‘plami;
*D) A) va B) javoblar to‘g‘ri.
94) Makrogenerator – bu:
A) Assembler dasturida operator nomi;
B) Izoh so‘z;
*C) Makroslarni yozishda foydalaniladigan assembler translyatorining bir qismi;
D) Mikroprotsessor elementi.
95) Makrokutubxona – bu:
*A) Assemblerda amaliy dasturlar uchun foydalaniladigan barcha universal makrobuyruqlar to‘plami;
B) Assemblerda barcha operatorlar majmuasi;
C) Makroslarni yozishda foydalaniladigan assembler translyatorining bir qismi;
D) To‘g‘ri javob berilmagan.
96) Assembler dasturida maxsus buyruqlar asosida formal parametrlarning haqiqiy parametrlar bilan o‘rin almashish jarayoniga … deyiladi.
A) Makroaniqlov;
*B)Makroalmashish.
C) Makrokengaytma;
D) To‘g‘ri javob berilmagan.
97) Assemblerda makrogenerator tomonidan bajariladigan amallar qaysi qatorda to‘g‘ri berilgan?
A) Makrogenerator dasturning kerakli joyiga ko‘rsatilgan nom bilan makroaniqlovni joylashtiradi;
B) Makroaniqlov qismida barcha formal parametrlarni haqiqiy parametrlar bilan almashtiradi;
C) Hosil qilingan makrokengaytmani dasturga makrobuyruqlar bilan qo‘yadi;
*D) Barcha javoblar to‘g‘ri.
98. Faqat makrotillar tuzilmasida ishlatiladigan operatorlar … deyiladi.
A) Oddiy operatorlar;
B) Makroslar;
C) Makrobuyruqlar;

*D)Makrooperatorlar.

99. int 19h uzilish dastur tarkibida qanday maqsadda qo‘llaniladi?

A) Aniq vazifasi yo‘q;

*B) Tizimni qayta yuklash funksiyasi;

C) Satrlarni ekranga chiqaradi

D) Tizimli taymer xizmatini amalgam oshiradi.

100. Assemblerda ushbu 0ef15h yozuvi nimani bildiradi?

*A)16 lik sonni ifodalaydi;

B) Satrli yozuvni ifodalaydi;

C) 2 lik sonni ifodalaydi;

D) Ma’no bildirmaydigan ifoda.

FOYDALANILGAN ADABIYOTLAR RO'YXATI

O'zbekiston Respublikasi Prezidentining farmonlari va qarorlari:

1. «Kompyuterlashtirishni yanada rivojlantirish va axborot – kommunikatsiya texnologiyalarini joriy etish to'g'risida» (30.05.2002 y. № UP - 3080);
2. «Telekommunikatsiya sohasida boshqarishni takomillashtirish chora-tadbirlari to'g'risida» (28.06.2000 y. № UP – 2647);
3. «Axborot texnologiyalari sohasida kadrlar tayYorlash tizimini takomillashtirish to'g'risida» (02.06.2005 y.).

O'zbekiston Respublikasi qonunlari:

4. «Telekommunikatsiya to'g'risida» (20.08.1999 y. № 822 – I);
5. «Axborot erkinligi printsiplari va kafolatlari to'g'risida» (12.12.2002 y. № 439-II);
6. «Axborotlashtirish to'g'risida» (11.12.2003 y. № 560 – II);
7. «Elektron rakamli imzo to'g'risida» (11.12.2003 y. № 562 – II);
8. «Elektron hujjat aylanishi to'g'risida» (29.04.2004 y. № 611 – II);
9. «Elektron tijorat to'g'risida» (29.04.2004 y. № 613 – II);

Vazirlar maxkamasining qarorlari:

10. «Kompyuterlashtirishni yanada rivojlantirish va axborot – kommunikatsiya texnologiyalarini joriy etish chora-tadbirlari
11. to'g'risida» (06.06.2002 y. № 200 VMK);
12. «O'zbekaloqa faoliyatini va axborotlashtirishni takomillashtirish chora-tadbirlari to'g'risida» (07.05.2004 y. № 215);

O'zbekiston Respublikasi Prezidenti I.A.Karimov asarlari :

13. Karimov I.A.. O'zbekiston demokratik taraqqiyotning yangi bosqichida. – T.: O'zbekiston, 2005.

Asosiy adabiyotlar

1. Svetin Yakov, Veselin Kolev & Co. Fundamentals of computer programming with C#. Sofia. 2013.
2. Эндрю Троелсен. C# и платформа .NET. - СПб.: Питер. 2012 г. – 796 с.:
3. M.Aripov. M.Fayziyeva, S.Dottayev. Web texnologiyalar. O'quv qo llanma. T.; "Faylasuflar jamiyati". 2013. 350 bet.
4. M.Aripov. A.Madraximov. Informatika, informatsion texnologiyalar. Darslik. T: TDYul.. 2004.
5. Том Арчер. Основы C#. М.: Изд.-торговый дом «Русская редакция», 2010 г.

Qo'shimcha adabiyotlar

1. Mirziyoev Sh. "Qonun ustuvorligi va inson manfaatlarini ta'minlash-yurt taraqqiyoti va xalq farovonligining garovi". O'zbekiston Respublikasi Konstitutsiyasi qabul qilinganining 24 yilligiga bag'ishlangan tantanali marosimdagi ma'ruza. 2016 yil 7-dekabr. -Toshkent: "O'zbekiston", NMIU, 2017.-32 bet.
2. Mirziyoev Sh. "Erkin va farovon, demokratik O'zbekiston davlatini birgalikda

barpo etamiz”. O’zbekiston Respublikasi Prezidenti lavozimiga kirishish tantanali marosimiga bag’ishlangan Oliy Majlis palatalarining qo’shma majlisidagi nutqi.-Toshkent: “O’zbekiston”, NMIU, 2017.-32 bet.
3. Mirziyoev Sh. “Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz”.-Toshkent: “O’zbekiston”, NMIU, 2017.-482 bet.
4. O’zbekiston Respublikasi Prezidentining “O’zbekiston Respublikasini yanada rivojlantirish bo’yicha Harakatlar strategiyasi” to’g’risidagi Farmoni (Xalq so’zi, 2017 yil, 8 fevral).
5. O’zbekiston Respublikasi Prezidentining Qarori “Oliy ta'lim tizimini yanada rivojlantirish chora-tadbirlari” to’g’risida (Xalq so’zi, 2017 yil, 21 aprel).
6. Лабор В. В. Си Шарп: Создание приложений для Windows/ В. В. Лабор — Мн.: Харвест, 2013.
7. Usmonov A.I., Baxramov F.D. Kompyuter texnologiyalari asoslari. O’quv qo’llanma -T: “O’zdavJTU”, 2010.
8. Aripov M.M., Kabiljanova F.A., Yuldashev Z.X. “Информационные технологии”. Электронное пособие для студентов ВУЗов (CD), Ташкент 2008, NUUz.
9. M.Aripov, B.Begalov. U.Begimqulov, M.Mamarajabov. Axborot texnologiyalar. O’quv qo’llanma. T.: “Noshir". 2009.

Internet – saytlar :

14. http:G’G’www.diamond.stup.ac.ruG’ENGG’F4G’DIRECTG’4.html - “Ta’limdagi yangi axborot texnologiyalari bo’yicha” Rossiyaning ta’limiy sayti.
15. http:G’G’www.spb.runnet.ru - Sank-Peterburg ilmiy-texnik axborotlar markazining serveri.
16. http:G’G’www.cnit.msu.ru - M.V.Lomonosov nomli MDUning yangi axborot texnologiyalari markazi.
17. http:G’G’www.gov.uz - O’zbekiston Respublikasi portali.
18. http:G’G’www.edu.uz - O’zbekiston Respublikasi OO’YU Vazirligining sayti.
19. http:G’G’www.tsue.fan.uz - Toshkent Davlat Iqtisodiyot Universitetining saytlari.
20. http:G’G’www.mgapi.edu - Moskva davlat asbobsozlik va informatika akademiyasi.
21. http:G’G’www.rea.ru - G.V.Plexanov nomli Rossiya iqtisodiyot akademiyasi.
22. http:G’G’www.fa.ru - Rossiya Federatsisi Xukumatining moliyaviy akademiyasi.
23. http:G’G’www.msu.ru - M.V.Lomonosov nomli Moskva Davlat universiteti..
24. http:G’G’www.spbu.ru - Sankt-Peterburg Davlat universiteti (SPbGU).
25. http:G’G’www.uef.ru - Sankt-Peterburg Davlat iqtisodiyot va moliya universiteti.

GLOSSARIY

Axborot – so’zi lotincha «informatio» co’zidan kelib chiqqan bo’lib «tushuntirish, tanishtirish, bayon etish» - degan ma’nolarni anglatadi.

Axborot texnologiyalari – axborotni yig’ish, saqlash, uzatish, o’zgartirish, qayta ishlash usul va vositalari yig’indisidan iborat.

Axborotlashtirish - axborot resurslari, axborot texnologiyalari va axborot tizimlaridan foydalanilgan holda yuridik va jismoniy shaxslarning axborotga bo’lgan ehtiyojlarini qondirish uchun shart-sharoitlar yaratishning tashkiliy ijtimoiy-iqtisodiy va ilmiy-texnikaviy jarayoni.

Axborot resurslari – insonlarning u yoki bu axborotlarga bo’lgan extiyojlarini qondirish uchun to’plangan barcha axborotlar tushuniladi.

Axborot tizimi – axborot resurslari, axborot texnologiyalari va aloqa vositalarining axborotni to’plash, saqlash, izlash, unga ishlov berish va undan foydalanish imkonini beradigan tashkiliy jihatdan tartibga solingan majmui.

Virtual auditoriya – masofali o’qitish sharoitida o’quvchilar guruhi o’quv - tarbiya jarayonini tashkil etish.

Virtual universitet – axborot – ta’lim muxiti, boshqarish bloklari va virtual ta’lim muassasasini boshqarish yig’indisi.

Virtual server – bu provayder bilan kelishgan holda kompyuterning imkoniyati darajasiga qarab alohida server ko’rinishda ishlashga mo’ljallangan qo’shimcha plata (elektron sxemalar) o’rnatilgan kompyuter hisoblanadi.

Veb sahifa - oddiy xat yoki kitob sahifasidan farqli narsa tushuniladi. Veb sahifa - Internet - provayderga xizmat qiluvchi va qandaydir firma, tashkilot yoki shaxsga tegishli xizmat qiluvchining uzoqlashtirilgan kompyuterida qandaydir nom bilan birlashtirilib, joylashgan axborotlar to’plamidir.

Veb sayt – bir nechta sahifalar turkumi sayt (site) deb ataladi. Veb sayt sahifalari o’lchamlari shaxsiy kompterning ekrani o’lchamlariga avtomatik ravishda moslashtiriladi.

Veb portal - so’zi inglizcha so’z bo’lib, umumiy ma’noda biror narsani birlashtirish, biriktirish nuqtasi ma’nolarida qo’llaniladi. Internet texnologiyasida esa turli xil universal servislarni birlashtiruvchi yirik sayt ma’nosida qo’llaniladi.

Global tarmoqlar – turli mamlakatlar yoki qit’alarda, er yuzining barcha joylarida joylashgan tarmoq foydalanuvchilarini birlashtiradi.

Gipermedia – matndan tashqari boshqa shakldagi ma’lumotlarni ham tavsiya qiluvchi hujjatlar hisoblanadi.

Gipermatn - Gipermatn ixtiyoriy joyda joylashgan o’zaro aloqadagi hujjatlar orasidagi aloqa. Ajratilgan jumla yoki so’zni "sichqoncha" bilan bosilganda foydalanuvchi tezgina ushbu mavzu yoritilgan fayllarga o’tishi mumkin.

Lokal (mahalliy) tarmoqlar (LAN - Local Agea Network)-bir xonadagi, binodagi, uncha katta bo’lmagan xududdagi kompyuter tarmoqlari.

Regional (mintaqaviy) tarmoqlar - bir-biridan ancha uzoqda joylashgan kompyuterlar va mahalliy tarmoqlarni o’zaro bog’laydi.

Masofali o’qitish – axborot-kommunikatsion texnologiya (kompyuterlar, telekommunikatsiya, multimedia) vositalari va ilmiy asoslangan o’qitish usullarini qo’llab ta’lim (kunduzgi, sirqi, eksternat) olish shaklidir.

Masofali ta’lim (distant education) – axborot texnologiyasini foydalangan holda masofadan turib ta’lim muxiti yordamida o’quv axborotlarini almashinishni ta’minlaydigan va o’quv jarayonini olib borish hamda boshqarish tizimini amalga oshiradigan bilim va ko’nikmalarni egallash jarayoni.

Elektron pochta (E mail)- Jo’natish uchun mo’ljallangan elektron xat, ya’ni matnli fayl foydalanuvchining manzili va xat mazmunidan tashkil topadi.

Elektron darslik - kompyuter texnologiyasiga asoslangan o’quv uslubini qo’llashga, mustaqil ta’lim olishga hamda fanga oid o’quv materiallar, ilmiy ma’lumotlarning har tomonlama samarador o’zlashtirilishiga mo’ljallangan.

INTERNET - International Network (xalqaro kompyuter tarmog’i) - butun dunyo kompyuter tarmog’i. Internet – WWW gipermatnlarga asoslanib axborot tashkil, uzatish va qabul qilish tizimini amalga oshiradigan xalqaro kompyuter tarmoqidir.

WWW - World Wide Web ("Umumjahon o’rgimchak to’ri") - gipermuhitga asoslanib Internetda tashkil qilingan axborotlar tizimi.

URL (Uniform Resource Locator - ashyolarni unifikasiya qilingan ko’rsatuvchisi) - Web-tugunning manzili. URL ko’rsatkichi odatda hujjatning transport bayonnomasini (masalan, HTTP yoki FTR) va u joylashgan xost - kompyuterning nomini ifodalaydi. Bundan tashqari URL ko’rsatkichlari o’zlarida ushbu kompyuterdagi hujjatga kirish marshrutini ham saqlashi mumkin. Ushbu marshrutlar URL satrining oxirida ko’rsatiladi.

Internet portali - deganda internetdan foydalanuvchilarga boshqa saytlardan axborotlar olishga imkon beruvchi, o’zida ko’plab o’quv-uslubiy, ilmiy materiallarni, me’yoriy-huquqiy hujjatlarni mujassamlashtirish imkoniyati mavjud veb resursdir.

Intranet (Intranet -intratarmoq) - dasturiy mahsulotlari va Internet texnologiyalarini foydalaniladigan biror tashkilotning mahalliy korporativ tarmog’idir, masalan, Web-server Intertarmoqlar brandmauerlar deb ataluvchi maxsus dasturlar yordamida Internetning tashqi foydalanuvchilaridan ajratilib qo’yilishi yoki tashqaridan aloqa qilaolmaydigan avtonom tarmoq sifatida ishlatilishi mumkin.

Tarmoq abonentlari – tarmoqda axborotlarni yuzaga keltiruvchi yoki iste’mol qiluvchi ob’ektlardir. Ularga alohida kompyuterlar, kompyuter komplekslari, sanoat inshootlari, dastur orqali boshqariladigan stanoklar va boshqalar misol bo’lishi mumkin. Har qanday abonent tarmog’i stantsiyaga ulangan bo’ladi.

Stantsiya – axborot uzatish va qabul qilish bilan bog’liq vazifalarni bajaruvchi texnik jihozlar yig’indisidir.

Uzel – tarmoqning uzatish vositasiga ulangan har qanday qurilmalar.

Server - bu tarmoqdagi kompyuterlar va ularga xizmat ko’rsatadigan tashkilotning kompyuterlaridir.

Servis–provayder – kompyuter tarmoqlari xizmatini tashkil qiluvchi

tashkilotlar hisoblanadi.

HTML formati (Huper Text Markup Language – gipermatnni belgilash tili) – Web sahifalarni yaratishda qo'llaniladigan til hisoblanadi.

Multimedia vositalari - informatikaning dasturiy va texnikaviy vositalari asosida audio, video, matn, grafika va animatsiya (ob'ektlarning fazodagi harakati) effektlari asosida o'quv materiallarini o'quvchilarga etkazib berishning mujassamlangan holdagi ko'rinishidir.

Zamonaviy axborot texnologiyalari – shaxsiy kompyuterlar va telekommunikatsiya vositalaridan foydalangan holda foydalanuvchi ishining do'stona interfeysli axborot texnologiyasidir.