«O’zbekiston temir yo’llari» AJ
Toshkent temir yo’l muhandislari instituti

“Nashrga ruxsat etaman”

O`quv ishlari bo`yicha prorektor

dotsent_____ F.F.Karimova

«___»_____20__ yil

Rasulmuxamedov Mahamadaziz Mahamadaminovich
Boltayev Avaz Xudoyberdievich

«Dasturlash tillari va texnologiyalari (C#, Python)»

(2-qism)

5330200-“ Informatika va axborot texnologiyalari (temir yo’l transportida)” ta`lim yo'nalishi 1-bosqich bakalavriat talabalari va professor-o’qituvchilar uchun uslubiy qo’llanma

Toshkent -2017

“O'zbekiston temir yo'llari” AJ
Toshkent temir yo'l muhandislari instituti

Dasturlash tillari va texnologiyalari (C#, Python)

(2-qism)

5330200-“ Informatika va axborot texnologiyalari (temir yo’l transportida)” ta`lim yo'nalishi 1-bosqich bakalavriat talabalari va professor-o’qituvchilar uchun uslubiy qo’llanma

Toshkent — 2017

UDK 681.31

Dasturlash tillari va texnologiyalari (C#, Python). Uslubiy qo’llanma. M.M. Rasulmuxamedov, A.X. Boltayev ToshTYMI, T. : 20__, ___ bet.

Uslubiy qo’llanma 5330200 – “Informatika va axborot texnologiyalari (temir yo‘l transportida)” ta`lim yo'nalishida tahsil olayotgan bakalavriat talabalari uchun “Dasturlash tillari va texnologiyalari (C#, Python)” fani bo‘yicha reja va ko‘rsatilgan 5 ta laboratoriya ishlaridan iborat bo‘lib, unda axborot tizimlari bo‘yicha barcha muhim tushuncha va ularni ishlab chiqish hamda yaratish tamoyillari keltirilgan. Laboratoriya ishlariga mos ravishda nazariy qismi bevosita keltirib o‘tilgan va topshiriq, vazifalar berilgan. Talabalarning Visual Studio muhitda ishlashi, S# tilida dastur yaratish qoidalari rasmiylashtirilgan. Har bir laboratoriya ishiga muhim mavzu, ishning maqsadi, laboratoriya ishiga vazifalar, nazariy qism (asosiy matn va obyektlar ko‘rinishi), o‘zlashtirish uchun topshiriqlar va foydalangan adabiyotlar ro‘yxati berilgan.

Institutning Ilmiy-uslubiy kengashi tomonidan nashrga tavsiya etildi

Taqrizchilar :

S.A.Xudoyberdiyev — kafedra mudiri(O‘zDSMI)

T.R.Nurmuxamedov — t.f.d., professor.

6-laboratoriya ishi

Uslublar

Yuqorida qayd etilganidek, nusxa o`zgaruvchilari va uslublar – sinflarning ikki asosiy tarkibiy qismidir. Bizning **Building** sinfimiz o`z ichiga hozircha faqat ma`lumotlarni olgan. Bunday sinflarga (uslublarsiz) yo`l qo`yilishi mumkin bo`lishiga qaramay, aksariyat sinflar o`z uslublariga ega. Uslublar – qism dasturlar bo`lib, ular sinfda belgilangan ma`lumotlar bilan ish bajaradilar va ko`p hollarda ushbu ma`lumotlarni olish imkonini beradi. Odatda dasturning turli qismlari sinf bilan uning uslublari orqali aloqaga kirishadi.

Har qanday uslub bir yoki bir necha operatorga ega. Yaxshi qo`shaloq dasturda bir uslub faqat bir vazifani bajaradi. Har bir uslub o`z nomiga ega bo`lib, aynan shu nom uni “chaqirish” (ishga tushirish) uchun qo`llaniladi. Umumiy holda uslubga istalgan nom berilishi mumkin. Biroq **Main**() nomi dastur bajarila boshlanadigan uslub uchun zaxiralanganini unutmang. Bundan tashqari, uslub nomlari sifatida C# kalit so`zlaridan foydalanish mumkin emas.

Uslub nomlari matnda bir juft qavs bilan birga qo`llanadi. Masalan, uslub nomi getval bo`lsa, u holda matnda **getval**() deb yoziladi. Bu o`zgaruvchilar nomlarini uslub nomlaridan farqlash imkonini beradi.

Uslubni qayd etish formati quyidagicha:

ruxsat_modifikatori qaytuvchi_qiymat_tipi nom (parametrlar_ro`yxati) {
// uslub_tanasi
}

Bu yerda *ruxsat_modifikatori* elementi dostup modifikatorini anglatib, u dasturning qaysi qismlari uslubga kirishiga yo`l qo`yilishi mumkinligini belgilaydi. Yuqorida aytilganidek, dostup modifikatori zaruriy emas, hatto u ko`rsatilmagan bo`lsa ham, uslub o`zi belgilangan sinf doirasida yopiq (private) ekanligi nazarda tutiladi. Hozircha biz barcha uslublarni **public** – a`zolar sifatida e`lon qilib, bu dastur kodining barcha qolgan, hatto sinfdan tashqaridan joy olgan tarkibiy qismlari uchun ham ularni chaqirish imkoniyatini beradi.

qaytuvchi_qiymat_tipi elementi yordamida uslub orqali qaytarilayotgan qiymat tipi ko`rsatiladi. Bu istalgan yo`l qo`yiladigan, shu jumladan dasturchi tomonidan yaratilayotgan sinf tiplaridan biri bo`lishi mumkin. Uslub bironta qiymatni ham ortga qaytarmasa, void tipi ko`rsatib o`tiladi.

Uslub nomi *nom* elementida beriladi. Uslub nomi sifatida joriy ko`rinish sohasi doirasidagi boshqa dastur elementlari uchun qo`llanganlaridan farqli istalgan yo`l qo`yiladigan identifikatordan foydalanish mumkin. ***parametrlar_ro`yxati*** elementi o`zaro vergullar bilan ajratilgan parametrlar (ma`lumotlar tipi va identifikatordan tuzilgan) ketma-ketligidan iborat. Parametrlar – uslubga chaqirish vaqtida beriladigan argumentlar qiymatini oladigan o`zgaruvchilardir. Uslub parametrlariga ega bo`lmasa, ***parametrlar_ro`yxati*** bo`shligicha qolaveradi.

Sinfga usullarni qo`shish

Sizga ma`lumki, sinf usullari, odatda, sinfda belgilangan ma`lumotlar yordamida manipulyatsiya qilib, ushbu ma`lumotlarni olish imkoniyatini yaratadilar. Buni bilgan holda, avvalgi dasturda Main() uslubi yordamida binoning umumiy maydonini unda yashovchilar miqdoriga bo`lish yo`li bilan bir odamga to`g`ri keladigan maydon hisoblab topilganini yodga tushiramiz.

Ushbu hisob-kitoblarni ko`p jihatdan maqbul ekanligiga qaramay, juda to`g`ri deb aytish qiyin. Bir odamga to`g`ri keladigan maydonni topish vazifasini **Building** sinfi ham bajara oladi, chunki bu qiymat miqdori faqat **Building** sinfida birlashgan **area** va **occupants** o`zgaruvchilari qiymatiga bog`liq. Bundan tashqari, agar u boshqa sinfga “mahkamlab” qo`yilgan holida undan foydalanayotgan boshqa dastur bu harakatni “yangidan” bajarishiga to`g`ri kelmaydi. Bu “boshqa” dasturlar uchun qulaylikning o`zi emas, balki bunda dastur matnini nomaqbul ravishda qayta takrorlashning oldi olinadi. Nihoyat, **Building** sinfiga bir kishiga to`g`ri keladigan maydonni hisoblab topadigan uslubni kiritish bilan Siz ko`rib chiqilayotgan sinf ichida bevosita bino bilan bog`liq qiymatlarni biriktirish bilan uning ob`ektga mo`ljallangan tuzilmasini yaxshilaysiz.

Building sinfiga uslubni qo`shib qo`yish uchun uni sinf e`loni ichida aniqlash zarur. Masalan, **Building** sinfi **areaPerPerson**() nomli uslubga ega bo`lib, u muayyan binoning bir kishiga to`g`ri keladigan maydoni qiymatini aks ettiradi.

```
//Building sinfiga uslubni qo`shish.
using System; class Building{
public int floors; // qavatlar miqdori
public int area; // binoning umumiy maydoni
public int occupants; // yashovchilar soni
//Bir kishiga to`g`ri keladigan maydon qiymatini aks ettiramiz,
public void areaPerPerson(){ Console.WriteLine(" "+area/occupants+
" bir odamga to`g`ri keladi ");}}//areaPerPerson()uslubini qo`llaymiz.
class BuildingDemo{ public static void Main(){
```

```
Building house = new Building();Building office = new Building();
//house ob`ektidagi maydonlarga qiymat beramiz
house.occupants = 4;house.area = 2500;
house.floors = 2; // office ob`ektidagi maydonlarga qiymat beramiz,
office.occupants = 25;office.area = 4200;office.floors = 3;
Console.WriteLine("Uyda bor:\n " + house.floors + " qavat \n " +
house.occupants + " yashovchi \n " +
house.area + " kvadrat futga teng umumiy maydon, shundan ");
house.areaPerPerson();Console.WriteLine();
Console.WriteLine("Ofisda mavjud:\n " + office.floors + " qavat \n " +
office.occupants + " xodim \n " +office.area + " kvadrat futga teng
 umumiy maydon, shundan ");office.areaPerPerson();}}
```

Ushbu dastur avvalgilari bilan mos tushadigan natijalarni generatsiyalaydi:

Unda quyidagilar mavjud:

2 qavat

4 yashovchi

2500 kvadrat fut umumiy maydon, undan

Bir kishiga 625 fut to`g`ri keladi

Ofisda mavjud:

3 qavat

25 ishchi-xodim

4200 kvadrat fut umumiy maydon, undan

bir kishiga 168 fut to`g`ri keladi

Endi bevosita **areaPerPerson**() uslubidan boshlab, ushbu dasturning kalit elementlarini ko`rib chiqamiz. Bu uslubning birinchi satrini ko`rinishi quyidagicha: public void areaPerPerson() { }

Mazkur satrda parametrlarga ega bo`lmagan **areaPerPerson**() nomli uslub e`lon qilinadi. Ushbu uslub **public** kirish modifikatoridan foydalangan holda aniqlangan, shuning uchun undan dasturning boshqa barcha qismlari foydalanishi mumkin.

areaPerPerson() uslubi void kabilarning qiymatini qaytarib beradi, ya`ni hech qanday qiymatni qaytarmaydi. Bu satr orqasidan uslub tanasi joylashgan ochadigan figurali qavs bilan tugatiladi.

Xuddi shunday **areaPerPerson**() uslub tanasi yagona **Console.WriteLine** operatoridan tashkil topadi; (" " + area / occupants + " bir odamga to`g`ri keladi").

Bu operatorda **area** o`zgaruvchining qiymatini **occupants** o`zgaruvchining qiymatiga nisbatini hisoblash orqali bir odamga to`g`ri keladigan bino

maydoni ko`rsatiladi. **Building** turidagi har bir ob`ekt o`zining **area** va **occupants** qiymati nusxasiga ega bo`lganligi uchun, bir odamga to`g`ri keladigan bino maydonini hisoblash jarayonida **areaPerPerson**() uslubni chaqirish uchun, aniq chaqiruvchi ob`ektga tegishli bo`lgan ushbu o`zgaruvchilarning nusxasidan foydalaniladi.

areaPerPerson() uslubi yopuvchi figurali qavs bilan tugallanadi, ya`ni yopuvchi figurali qavs topilganda, dasturni boshqarish chaqiruvchi ob`ektga uzatiladi. Endi e`tibor bilan Main() uslubidagi kod satrini ko`rib chiqamiz:

house.areaPerPerson();

Ushbu operator house ob`ekti uchun **areaPerPerson**() uslubini chaqirib beradi. Ko`rayapsizki, buning uchun ketidan "nuqta" operatori keladigan ob`ekt nomidan foydalaniladi. Uslub chaqirilganda, dastur boshqarilishini amalga oshirish uslub tanasiga uzatiladi, u tugatilgandan so`ng esa boshqaruv chaqiruv muallifiga qaytariladi hamda bevosita uslubni chaqirishdan keyin joylashgan dastur matni satridan boshlab dasturni bajarish qaytadan boshlanadi.

Bu holda **house.areaPerPerson**() chaqirish natijasida, **house** ob`ekti tomonidan belgilangan bino uchun, bir odamga to`g`ri keladigan maydon miqdori ko`rsatiladi.

Xuddi shunday **office.areaPerPerson**() ni chaqirish natijasida **office** ob`ekti ko`rsatgan bino uchun bir odamga mos keluvchi maydon miqdori ko`rsatiladi. Boshqacha qillib aytganda, har gal, **areaPerPerson**() uslubi chaqirilganda, berilgan ob`ekt tomonidan berilgan bino uchun odamga mos keluvchi maydon miqdori ko`rsatiladi.

Shunga e`tibor bering. **area** va **occupants** nusxaning o`zgaruvchilari **areaPerPerson**() uslub ichida hech qanday atributlarsiz qo`llaniladi, ya`ni ulardan oldin na ob`ekt nomi, na "nuqta" operatori bo`lmaydi. Bu juda muhim: agar uslub o`z sinfidagi nusxaning o`zgaruvchisini ishga solsa, u buni to`g`ridan-to`g`ri "nuqta" operatorisiz hamda ob`ektni bahona qilmasdan amalga oshiradi. Bu esa mantiqan to`g`ri. Axir uslub doimo aniq bir sinfga mansub bo`lgan ob`ekt uchun chaqiriladi. Demak, chaqiruv amalga oshirilgan bo`lsa, ob`ekt ma`lum bo’ladi. Shunday qilib, uslub ichida ob`ektni ikkinchi marotaba ko`rsatishning hojati bo`lmaydi. Demak, **areaPerPerson**() uslubi ichidagi **area** va **occupants** larning qiymatlari **areaPerPerson**() uslubi keltirib chiqaradigan va ob`ektga tegishli bo`lgan ushbu o`zgaruvchilarning nusxalariga aniq ishora qilmaydilar.

Uslubdan qaytish

Umumiy holda uslubdan qaytish shartlarining ikki varianti mavjud. Birinchisi uslub tanasini tugaganini belgilovchi figurali qavsni topilishi

bilan bog`liq (**areaPerPerson**() uslubi misolida namoyish qilinganidek). Ikkinchi variant **return** operatorini bajarishdan iborat. **return** operatoridan foydalanishning ikki shakli mavjud: birinchisi void-uslublar uchun mo`ljallangan (qiymat qaytarmaydiganlari), boshqalari esa qiymatini qaytarish uchun. Ushbu qismda biz birinchi shaklni ko`rib chiqamiz, keyingisida esa ikkinchisini.

void - uslubni zudlik bilan tugatishni return operatorining quyidagi mojno shakli yordamida amalga oshirish mumkin: **return;**

Bu operatorni bajarishda dasturni boshqaruvi uslub chaqiruvi muallifiga o`tkaziladi, qolgan kod esa o`tkazib yuboriladi. Masalan, quyidagi usulni ko`rib chiqamiz:

```
public void myMeth(){
int i; for (i = 0; i < 10; i++){
if (i == 5) return; // i = 5da usulni amalga oshirilishini to`xtatilishi.
Console.WriteLine();}}
```

Bunda sikl for i ning faqat 0 dan 5 gacha diapazonidagi qiymatlarida ishlaydi, chunki ***i*** ning qiymati 5ga teng bo`lishi bilan, **myMeth**() uslubidan chiqish malga oshiriladi. Uslub bir nechta **return** operatorlariga ega bo`lishi muimkin. Masalan,

```
public void myMeth() { if(done) return; //... if(error) return; }
```

uslubidan chiqish uni aniq tugatilishi, yoki xato paydo bo`lgan hollarda sodir bo`ladi. Ammo uslubdan chiqish nuqtalarining haddan ziyod ko`p bo`lgan hollarda dastur matni destrukturalanishi mumkin. Shuning uchun, ularning ko`p sonidan foydalanishga yo`l qo`yilgan bo`lsa ham, bu imkoniyatni katta ehtiyotlik bilan qo`llash ma`qul. Shunday qilib, void-uslubdan chiqish ikki usulda amalga oshirilishi mumkin: yopadigan figurali qavsga yetganda yoki return operatori bajarilganda, buni xotiramizda saqlaymiz.

Qiymatini qaytarish

Garchi void-uslublar kamyob bo`lmasa ham, ko`pchilik usullar qiymatini qaytaradi. Haqiqiatdan ham, qiymatni qaytara olish xususiyati - uslubning eng foydali xususiyatlaridan biridir.

Uslublar yordamida qaytarilgan qiymatlardan dasturlashda turlicha foydalaniladi. Ayrim hollarda (**Math.Sqrt**() uslubidagi kabi) qaytariladigan qiymat hisoblash natijasi bo`ladi, boshqasida u faqatgina amallar muvaffaqiyatli yoki muvaffaqiyatsiz bajarilganligini bildiradi , uchinchisida esa u holat kodini ifodalashi mumkin. Ammo, nima maqsadda fodalanilishidan qat`iy nazar, uslublar bilan qaytarilgan qiymatlardan foydalanish C# dasturlash tilining ajralmas qismidir. Uslublar “**return**

qiymat;” shaklidan foydalanib, ularni chaqiruvchi protseduralarga ***qiymat***ni qaytaradilar.

Bunda ***qiymat*** elementi uslub qaytargan qiymatni bildiradi.

Uslublarning qiymatni qaytarish imkoniyatlarini **areaPerPerson**() uslubidan foydalanishni yaxshilashda qo`llash mumkin. Bir odamga to`g`ri keladigan maydonni ifodalash o`rniga **areaPerPerson**() uslubi endilikda boshqa hisob kitoblarda foydalanish mumkin bo`lgan qiymatni qaytaradi.

Quyidagi misolda **areaPerPerson**() uslubining modifikatsiyalashgan varianti keltirilgan, qaysiki bir odamga to`g`ri keladigan maydonni ifodalamaydi (avvalgi variantdagi kabi), balki maydon qiymatini qaytaradi. **areaPerPerson**() uslubida qiymatni qaytarishni namoyish qilish.

```
class Building{
public int floors; // qavatlar soni
public int area; // binoning umumiy maydoni
public int occupants; // yashovchilar soni
//Bir odamga to`g`ri keluvchi maydon miqdorini qaytarish
public int areaPerPerson(){
return area / occupants;
}}//areaPerPerson()uslubi bo`yicha topilgan qiymatdan foydalanish.
class BuildingDemo{
public static void Main(){
Building house = new Building();
Building office = new Building();
int areaPP; G`G` Bir odamga to`g`ri keluvchi maydon miqdorini.
//Ob`ektdagi hoshiyalarga qiymat berish
house.occupants = 4;
house.area = 2500;
house.floors = 2;
//office ob`ektdagi hoshiyalarga qiymat berish,
office.occupants = 25;office.area = 4200;office.floors = 3;
//house ob`ekti uchun bir odamga to`g`ri keluvchi maydon olamiz
areaPP = house.areaPerPerson();
Console.WriteLine("Uy ega:\n " +
house.floors + " qavat \n " +
house.occupants + " yashovchi \n " +
house.area + " kvadrat fut umumiy maydon,ulardan \n" +
areaPP + " bir odamga to`g`ri keladi ");
Console.WriteLine(); // Bir odamga to`g`ri keluvchi office uchun
//maydon miqdorini olamiz.
areaPP = office.areaPerPerson();
```

```
Console.WriteLine("Ofis ega:\n " + office.floors +
" etaja \n " + office.occupants + " ishchilar \n " +
office.area + " kvadrat fut umumiy maydon,ulardan \n " +
areaPP + " bir odamga to`g`ri keladi ");}}
```

Dasturning bu variantini bajarish avvalgilariniki kabi.

Dasturning ushbu variantini bajarish natijalari avvalgilari bilan bir xil. **areaPerPerson**() uslubini chaqiruviga e`tibor qarating: uning nomi o`zlashtirish operatoridan o'ng tarafda joylashgan. Chap tomonida **areaPerPerson**() uslubi yordamida qaytariladigan qiymatga ega bo`ladigan o`zgaruvchi joylashadi.

Shunday qilib, **areaPP = house.areaPerPerson()** operatorga amal qilingandan so`ng **house** ob`ekti uchun har bir odamga to`g`ri keladigan maydon miqdori o`zgaruvchi **areaPP** da saqlanadi. **areaPerPerson**() uslubi mazkur misolda qaytarilayotgan qiymatning boshqa tipiga ega ekanligiga e`tibor qiling, xususan, **int** tipiga. Bu uslub chaqiruv muallifiga butun son qaytaradi degani. Uslub qaytarayotgan qiymatning tipi uslubning eng muhim tavsifidir, chunki uslub qaytaradigan ma`lumotlar tipi uslub ta`rifi boshida keltirilgan qaytarilayotgan qiymat tipiga mos bo`lishi kerak.

Shunday qilib, agar uslub **double** tipidagi ma`lumotlarni qaytarishini istasangiz, uni aniqlashda qaytarilayotgan qiymat sifatida **double** ni ko`rsatish zarur.

Avvalgi dasturning maqbulligidan qat`iy nazar, uning samaradorligi yetarli darajada emas. Xususan, o`zgaruvchi **areaPP** qo`llashga xojat bo`lmaydi.

areaPerPerson() uslubini chaqiruvini bevosita WriteLine() chaqiruv operatorida amalga oshirish mumkin:

```
Console.WriteLine ("Uy ega:\n "+house.floors+" qavatga \n "+
house.occupants+"yashovchining \n "+house.area+" umumiy maydonini,
shundan \n "+house.areaPerPerson()+" bir odamga to`g`ri keladi ");
```

Bu holda **WriteLine**() uslubini chaqirganda, avtomatik ravishda **house**. **areaPerPerson**() uslubi chaqiriladi va uning yordamida chaqirilgan qiymat **WriteLine**() uslubiga uzatiladi. Bundan tashqari, endi **house. areaPerPerson**() uslubiga murojaatdan bir odamga to`g`ri keladigan maydon miqdori **Building** sinfidagi ob`ekt uchun zarur bo`lgan barcha hollarda foydalanish mumkin. Masalan, quyidagi operatorda ikkita binoning xuddi shunday maydonlari miqdori solishtiriladi.

```
if (bl.areaPerPerson() > b2.areaPerPerson())
Console.WriteLine(" b1 binosida har bir odam uchun joy ko`proq.");
```

Parametrlardan foydalanish

Chaqirish vaqtida uslubga bir yoki bir nechta miqdor uzatish mumkin. Yuqorida aytib o`tilganidek, uslubga uzatiladigan miqdor argument deb nomlanadi. Argument miqdorini qabul qiladigan uslub ichidagi o`zgaruvchi parametr deb nomlanadi. Parametrlar uslub nomidan keyin turgan dumaloq qavslar ichida paydo bo`ladi. Parametrlarni chiqaruvchi Sintaksis o`zgaruvchilar uchun qo`llaniladigan sintaksis bilan bir xil bo`ladi. Parametr o`z uslubining ko`rish doirasida joylashadi hamda argument olish kabi maxsus vazifadan tashqari ixtiyoriy lokal o`zgaruvchiga o`xshab harakat qiladi.

Quyida parametrli uslubdan foydalanishga doir oddiy misol. **ChkNum** sinfida **isPrime** uslubi aniqlanadi, **true** qiymatini qaytaradi, agar unga uzatilgan qiymat oddiy bo`lsa va aksincha, qiymat murakkab hollarda **false** qiymati qaytariladi. Demak, **isPrime** uslubi **bool** tipidagi qiymatni qaytaradi.

```
//Parametrdan foydalanishga doir oddiy misol.
using System;
class ChkNum{
//Uslub true ni qaytaradi, agar x – oddiy son bo’lsa,
public bool isPrime(int x){
for (int i = 2; i < x / 2 + 1; i++)
if ((x % i) == 0) return false;
return true;}}
class ParmDemo{
public static void Main(){
ChkNum ob = new ChkNum();
for (int i = l; i < 10; i++)
if (ob.isPrime(i)) Console.WriteLine(i + " oddiy son.");
else Console.WriteLine(i + " oddiymas son.");}}
```

Bu dastur quyidagi natijalarni generatsiya qiladi:

1 oddiy son.
2 oddiy son.
3 oddiy son.
4 oddiymas son.
5 oddiy son.
6 oddiymas son.
7 oddiy son.
8 oddiymas son.
9 oddiymas son.

Ushbu dasturda **isPrime** uslubi to`qqiz marotaba chaqiriladi va har gal unga yangi qiymat uzatiladi. Bu jarayonni alohida e`tibor bilan ko`rib chiqamiz. Birinchidan, **isPrime** uslubiga murojaat qanday amalga oshirilishiga e`tibor qiling. Uzatiladigan argument dumaloq qavslar orasida ko`rsatiladi. **isPrime** uslubini birinchi chaqirilishida unga 1 miqdori beriladi. Bu uslubni bajarishdan oldin ***x*** parametriga 1 miqdor beriladi. Ikkinchi chaqiruvda argument 2ga teng bo`ladi, demak parametr ham 2 miqdoriga ega bo`ladi va h.k. Muhimi, **isPrime** funksiyasi chaqirilganda argument sifatida uzatilgan miqdor ***x*** parametri oladigan miqdorning o`zi bo`ladi.

Uslub bittadan ortiq parametrga ega bo`lishi mumkin. Bunday hollarda har bir parametrni keyingisidan vergul bilan ajratib turib e`lon qilishni o`zi yetarli bo`ladi. Masalan, o`zimizga avvalgi dasturdan ma`lum unga uzatiladigan miqdorlarga (***east common denominator***) eng kichik umumiy maxrajni qaytarib beruvchi **led**() uslubini qo`shib **ChkNum** sinfni kengaytiramiz.

```
// Ikkita argument qabul qiluvchi uslubni qo`shamiz.
using System;
class ChkNum{
// Uslub true ni qaytaradi, agar x – oddiy son bo`lsa,
public bool isPrime(int x){
for (int i = 2; i < x / 2 + 1; i++)
if ((x % i) == 0) return false;
return true;}
// Uslub eng kichik umumiy maxrajni qaytaradi,
public int led(int a, int b){
int max;
if (isPrime(a) | isPrime(b)) return 1;
max = a < b ? a : b;
for (int i = 2; i < max / 2 + 1; i++)
if (((a % i) == 0) & ((b % i) == 0)) return i;
return 1;}}
class ParmDemo{
public static void Main(){
ChkNum ob = new ChkNum();
for (int i = 1; i < 10; i++)
if (ob.isPrime(i)) Console.WriteLine(i + " oddiy son.");
else Console.WriteLine(i + " oddiymas son.");
int a, b; a = 7; b = 8;
Console.WriteLine("uchun eng kichik umumiy maxraj " + a +
```

" va " + b + " teng " + ob.led(a, b)); a = 100; b = 8;
Console.WriteLine("uchun eng kichik umumiy maxraj " + a +
" va " + b + " teng " + ob.led(a, b)); a = 100; b = 75;
Console.WriteLine("uchun eng kichik umumiy maxraj " + a +
" va " + b + " teng " + ob.led(a, b));
Console.ReadKey();}}

led() uslubini chaqirganda ham argumentlar vergullar bilan ajratib qo`yilishiga e`tibor bering. Ushbu dasturni bajarilishi natijalari quyidagicha:

1 oddiy son.
2 oddiy son.
3 oddiy son.
4 oddiymas son.
5 oddiy son.
6 oddiymas son.
7 oddiy son.
8 oddiymas son.
9 oddiymas son.
7 va 8 uchun eng kichik umumiy maxraj 1 ga teng
100 va 8 uchun eng kichik umumiy maxraj 2 ga teng
100 va 75 uchun eng kichik umumiy maxraj 5 ga teng.

Uslubga bir nechta parametrni uzatishda har birining tipi ko`rsatilishi kerak, buning ustiga parametrlar tipi turli bo`lishi mumkin.

Masalan, quyidagi yozuvga bemalol yo`l qo`yiladi:

int myMeth(int a, double b, float s) {
// ... Building sinfiga parametrlashtirilgan uslubni qo`shish
}

Sinfga qo`shish

Yangi vositani (binoda yashovchilarni yo`l qo`yiladigan eng ko`p sonini hisoblash) sinfga qo`shish uchun parametrlashtirilgan uslubdan foydalanish mumkin. Bu yangi uslubni **maxOccupant**() deb nomlaymiz va uning ta`rifini keltiramiz.

/* Uslub odamlar sonining maksimal miqdorini qaytaradi, agar har biriga maydonning minimal miqdori to`g`ri kelsa. */

public int maxOccupant (int minArea){ return area / minArea; }

maxOccupant() uslubini chaqirganda, parametr **minArea** har bir odamning hayotiy faoliyati uchun zarur bo`lgan eng kichik maydon miqdoriga ega bo`ladi. **maxOccupant**() uslub tomonidan qaytariladigan natija, bino umumiy maydonini shu miqdorga nisbatidan kelib chiqadi.

Tarkibida **maxOccupant**() uslubi mavjud bo`lgan **Building** sinfning to`liq ta`rifini keltiramiz.

/* Har biriga maydonning belgilangan eng kam miqdori to`g`ri keladi deb taxmin qilgan holda binoda joylasha oladigan odamlarning maksimal sonini hisoblaydigan parametrlashtirilgan uslubni qo`shamiz. */

```
using System;
class Building{
public int floors; // qavatlar soni
public int area; // binoning umumiy maydoni
public int occupants; // yashovchilar soni
// Uslub har bir odamga to`g`ri keladigan maydonni qaytaradi,
public int areaPerPerson(){
return area / occupants;
}
```

/* Uslub binodagi odamlar sonini yo`l qo`yarli maksimal qiymati **r** ni qaytaradi, agar har biriga belgilangan maydonning minimal qiymati to`g`ri kelsa. */

```
public int maxOccupant(int minArea){
return area / minArea;
}}
// maxOccupant()uslubidan foydalanish.
class BuildingDemo{
public static void Main(){
Building house = new Building();
Building office = new Building();
// house ob`ektida hoshiyalarga qiymat beramiz
house.occupants = 4;
house.area = 2500;
house.floors = 2; // office ob`ektida hoshiyalarga qiymat beramiz,
office.occupants = 25;
office.area = 4200;office.floors = 3;
Console.WriteLine("Uy uchun odamlar sonining maksimal qiymati, \n"
+" agar har biriga to`g`ri keladigan bo`lsa " + 300 +
" kvadrat fut: " + house.maxOccupant(300));
Console.WriteLine("Ofis uchun odamlarning maksimal soni, \n" +
" agar har biriga to`g`ri keladigan bo`lsa " + 300 +
" kvadrat fut: " + office.maxOccupant(300));
Console.ReadKey();
}}
```

Ushbu dasturni bajarilishi natijalari quyidagi ko`rinishga ega:

Uy uchun odamlarning maksimal soni,
Agar har biriga 300 kvadrat fut to`g`ri keladigan bo`lsa: 8
Ofis uchun odamlarning maksimal soni,
Agar har biriga 300 kvadrat futdan to`g`ri keladigan bo`lsa: 14

Konstruktorlar

Avval keltirilgan misollarda har bir **Building** ob`ekt uchun o`zgaruv-chilar operatorlarning quyidagi ketma- ketligi yordamida "qo`lda" belgilanar edi:

house.occupants = 4;
house.area = 2500;
house.floors = 2;

Mutaxassis hech qachon bunday yondashmaydi. Gap faqat shu yo`l bilan bir yoki bir necha ma`lumotlarni osongina "esdan chiqarish" mumkinligida emas, balki buni amalga oshirishning bundan qulayroq usullarni mavjudligidadir. Bu usul – konstruktordan foydalanishdan iborat.

Konstruktor ob`ektni hosil qilishda uni initsializatsiya qiladi. U sinf bilan bir xil nomga ega bo`lib, sintaksis jihatidan uslubga o`xshash. Ammo konstruktorlar ta`rifida qaytariladigan qiymat tipi ko`rsatilmaydi. Konstruktor yozilishi formati shunday:

ruxsat_modifikatori sinf_nomi () {
// konstruktor tanasi
}

Odatda konstruktor sinfda belgilangan o`zgaruvchilariga boshlang`ich qiymatlarni berish uchun yoki butunlay shakllangan ob`ektni hosil qilishga zarur bo`lgan boshlang`ich amallarni bajarish uchun qo`llaniladi. Bundan tashqari, *ruxsat_modifikatori* elementi sifatida yaqinlashuv modifikatori “public” dan foydalaniladi, chunki odatda, konstruktorlar ularning sinfidan tashqari chaqiriladi.

Barcha sinflar ularni aniqlashingizdan yoki aniqlamasligingizdan qat`iy nazar konstruktorlarga ega, chunki C# avtomatik ravishda bo`sh konstruktorni taqdim etadi, qaysiki qiymat tipiga ega barcha o`zgaruvchi a`zolarni nollar bilan initsializatsiya qiladi. Ammo, agar Siz shaxsiy konstruktoringizni aniqlasangiz, bo`sh konstruktori boshqa ishlatilmaydi.

Mana konstruktordan foydalanishga doir misol:

```
// Oddiy konstruktordan foydalnish.
using System;
class MyClass{
public int x;
public MyClass(){
x = 10;}}
```

```
class ConsDemo{
public static void Main(){
MyClass tl = new MyClass();
MyClass t2 = new MyClass();
Console.WriteLine(tl.x + " " + t2.x);
Console.ReadKey();}}
```

Bu dastur misolida MyClass sinfi konstruktori quyidagi ko`rinishga ega:

```
public MyClass(){ x = 10; }
```

MyClass sinfidan tashqarida belgilangan koddan chaqirish imkonini beruvchi konstruktor **public** ta`rifiga e`tibor bering. Bu konstruktor ekzemplyar o`zgaruvchisi ***X***ga 10 sonini beradi. **MyClass**() konstruktori **new** operatori tomonidan **MyClass** sinfi ob`ektini hosil qilishda chaqiriladi. Masalan:

```
MyClass t1=new MyClass();
```

satrini **t1** ob`ekti uchun bajarganda konstruktori chaqiriladi, qaysiki **tl.x** nusxasining o`zgaruvchisiga 10 sonini beradi. Shuning o`zi **t2.x** ob`ekti uchun ham to`g`ridir, ya`ni **t2** ob`ekti hosil qilinishi natijasida **t2.x** nusxa o`zgaruvchisining qiymati ham 10 ga teng bo`lib qoladi. Shunday qilib, ushbu dastur bajarilgandan so`ng quyidagi natijani olamiz:

10 10.

Parametrlashtirilgan konstruktorlar

Avvalgi misolda parametrsiz konstruktordan foydalanildi. Ammo ko`proq bir yoki bir nechta parametr qabul qiladigan konstruktorlar bilan ishlanadi. Parametrlar konstruktorlarga huddi uslubga kiritilgandek kiritiladi: buning uchun ularni konstruktor nomidan keyin dumaloq qavs ichida e`lon qilishni o`zi yetarli. Masalan, keyingi dasturda parametrlash-tirilgan konstruktordan foydalaniladi.

```
// Parametrlashtirilgan konstruktordan foydalanish.
using System;
class MyClass{
public int x;
public MyClass(int i) { x = i; }
}
class ParmConsDemo{
public static void Main(){
MyClass tl = new MyClass(10);
MyClass t2 = new MyClass(88);
Console.WriteLine(tl.x + " " + t2.x);
Console.ReadKey();}}
```

MyClass() sinfidan tashqarida belgilangan koddan chaqirish imkonini beruvchi konstruktor **public** ta`rifiga e`tibor bering. Bu konstruktor ekzemplyar o`zgaruvchisi **X** ga 10 sonini beradi. **MyClass**() konstruktori **new** operatori tomonidan **MyClass**() sinfi ob`ektini hosil qilishda chaqiriladi. Masalan:

MyClass tl = new MyClass(10);

satrini **t1** ob`ekti uchun bajarganda konstruktori chaqiriladi, qaysiki **tl.x** nusxasining o`zgaruvchisiga 10 sonini beradi. Shuning o`zi **t2.x** ob`ekti uchun ham to`g`ridir, ya`ni **t2** ob`ekti hosil qilinishi natijasida **t2.x** nusxa o`zgaruvchisining qiymati ham 10 ga teng bo`lib qoladi. Shunday qilib, ushbu dastur bajarilgandan so`ng quyidagi natijani olamiz.

10 10

Sinfga konstruktor qo`shish

Biz **Building** sinfini unga ob`ekt hosil qilinganda **floors**, **area** va **occupants** nusxa o`zgaruvchilarini avtomatik ravishda initsiallashtiradigan konstruktorni qo`shib **Building** sinfini yaxshilashimiz mumkin, **Building** sinfi ob`ektlari qanday hosil qilinishiga alohida e`tiboringizni qarating.

```
// Konstruktorni Building sinfiga.
using System;
class Building{
public int floors; // qavatlar soni
public int area; // bino asosining umumiy maydoni
public int occupants; // yashovchilar soni
public Building(int f, int a, int o){
floors = f; area = a; occupants = o;
// uslub bir odamga to`g`ri keladigan
// maydon qiymatini qaytaradi
public int areaPerPerson(){
return area / occupants;}
/* Uslub binodagi odamlarning maksimal mumkin bo`lgan sonini qaytaradi, agar har biriga maydonning minimal belgilangan maydoni to`g`ri keladigan bo`lsa. */
public int maxOccupant(int minArea){
return area / minArea;}}
// parametrlashtirilgan konstruktor Building()dan foydalanamiz.
class BuildingDemo{
public static void Main(){
Building house = new Building(2, 2500, 4);
Building office = new Building(3, 4200, 25);
Console.WriteLine("Uy uchun odamlar sonining maksimal miqdori,
```

```
\n" + " agar har biriga to`g`ri keladigan bo`lsa " + 300 +
" kvadrat futlar: " + house.maxOccupant(300));
Console.WriteLine("ofis uchun odamlar sonining maksimal miqdori,
\n" + " agar har biriga to`g`ri keladigan bo`lsa " + 300 +
" kvadrat futlar: " + office.maxOccupant(300));
Console.ReadKey();
}}
```

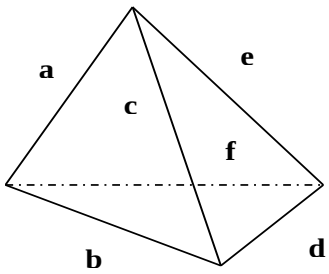
Bu bajarilishi natijalari uning avvalgi versiyasi natijalariga mos keladi.

house va **office** ob`ektlarning ikkalasi ham hosil qilinish vaqtida dasturda **Building**() konstruktori tomonidan initsializatsiyalanadi. Har bir ob`ekt konstruktorga uzatiladigan parametrlar qanday ko`rinishda berilganga qarab initsializatsiyalashtiriladi. Masalan, 1 chi satrni bajarishda:

Building house = new Building(2, 2500, 4).

Building() konstruktoriga **new** operatori tomonidan **Building** sinfi ob`ekti hosil qilinayotgan vaqtda 2, 2500 va 4 qiymatlari uzatiladi. Buning natijasida **house** ob`ektiga tegishli **floors**, **area** va occupants o`zgaruvchilar nusxalari tegishlicha 2, 2500 va 4 qiymatga ega bo`ladi.

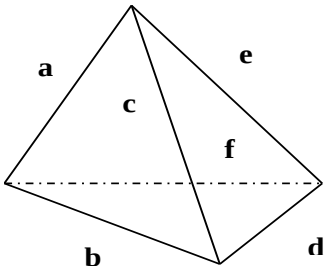
Misol	Uslubdan foydalanib to`rtqirralik maydonini toping, agar qirralari uzunligi berilgan bo`lsa.



Geron formulasi bo`yicha uchburchak maydonini hisoblash protsedura yordmida rasmiylashtiriladi

```
using System;
namespace ConsoleApplication1{
class Class1{
/* Geron formulasi bo`yicha uchburchak maydonini hisoblash protsedurasini bayoni */
public static void Sq(double x,double y, double z, out double s){
double p=(x+y+z)/2;
s=Math.Sqrt(p*(p-x)*(p-y)*(p-z));}
public static void Main(string[]args){
double a=double.Parse(Console.ReadLine());
double b=double.Parse(Console.ReadLine());
double c=double.Parse(Console.ReadLine());
double d=double.Parse(Console.ReadLine());
double e=double.Parse(Console.ReadLine());
double f=double.Parse(Console.ReadLine());
double s,s1,s2,s3,s4;
Sq(a,b,c,out s1); Sq(c,d,e,out s2); Sq(a,f,e,out s3);
Sq(b,d,f,out s4); s=s1+s2+s3+s4;
Console.WriteLine("s="+s);}}}
```

Misol.	Uslubdan foydalanib to`rtqirralik maydonini toping, agar qirralari uzunligi funksiyadan foydalanib berilgan bo`lsa.



Geron formulasi bo`yicha uchburchak maydonini hisoblang.

```
using System;
namespace ConsoleApplication1{
class Class1{
/* Geron formulasi bo`yicha uchburchak maydonini hisoblash funksiyasi bayoni*/
public static double Sq(double x, double y, double z){
double p=(x+y+z)/2;
double s=Math.Sqrt(p*(p-x)*(p-y)*(p-z));
return s;}
public static void Main(string[]args){
double a=double.Parse(Console.ReadLine());
double b=double.Parse(Console.ReadLine());
double c=double.Parse(Console.ReadLine());
double d=double.Parse(Console.ReadLine());
double e=double.Parse(Console.ReadLine());
double f=double.Parse(Console.ReadLine());
double s,s1,s2,s3,s4;
s= Sq(a,b,c) + Sq(c,d,e)+Sq(a,f,e )+ Sq(b,d,f);
Console.WriteLine("s="+s);}}}
```

«Rekursiya» so`zi lotincha «recursion» so`zi – qaytish dan kelib chiqqan.

Agar protsedura –funksiya o`ziga o`zi protsedura yoki funksiya sifatida bevosita yoki nimdasturlar zanjiri orqali murojaat qilsa, bunga rekursiya deb nomlanadi.

Shunga o`xshash tipdagi dasturlar qaytarilavermasligi uchun (bu haqiqatga yaqin), birinchi navbatda, rekursiyadan chiqish ta`minlanishi kerak.

Misol. Rekursiya qo`llanadigan uslub yordamida berilgan son faktoriali hisoblansin.

Berilgan son faktoriali – bu 1 dan berilgan songacha natural sonlar ko`paytmasi. Masalan, ***n***- faktorial bu ***n!=1*2*3*…*n***

```
using System;
namespace ConsoleApplication1{
class Class1{
/* Faktorialni hisoblash funksiyasi bayoni */
public static int Fact(int n){
if(n==1)return 1;     // Rekursiya tugatilishi shartini tekshirish
else return n*Fact(n-1);    // n! rekursiv hisoblash}
public static void Main(string[]args){
int n=int.Parse(Console.ReadLine());
// Haqiqiy  parametr n funksiyasini chaqirish
int f= Fact(n);Console.WriteLine("f="+f);}}}
```

6-laboratoriya ishi yuzasidan nazorat savollari

1. Nimdastur nima?
2. Funksiya va protsedura deb nimaga aytiladi?
3. Funksiya va protsedura orasidagi farqni tavsiflang.
4. Protsedura va funksiya qaysi bo`limda tavsiflangan?
5. Rasmiy parametrlar deb nimaga aytiladi?
6. Haqiqiy parametrlar deb nimaga aytiladi?
7. Parametr-miqdorlar va parametr-o`zgaruvchilar orasida qanday farq bor?
8. Funksiya va o`zgaruvchialarga qanday murojaat etiladi?

7-laboratoriya ishi

Satrlar bilan ishlash

Bir vaqtlar komp`yuterlardan faqat soniy miqdorlarni boshqarish uchun foydalanilardi. Birinchi kompyuterlar faqat raketalar trayektoriyasini hisoblash uchun kerak bo`lgan hamda dasturlash faqat yetakchi universitetlarning matematik fakultetlarida o`qitilar edi.

Bugungi kunda ko`pchilik dasturlar sonlarga nisbatan satrlar bilan ko`proq ishlaydi. Odatda, satrlardan matnlarni qayta ishlashda, hujjatlar bilan manipulyatsiya qilish va Web-sahifalarni yaratish uchun foydalaniladi.

Satrlar bilan ishlashni ichki tomondan qo`llanishini S# ta`minalydi. Bundan tashqari, S# satrlarni ob`ektlar kabi qayta ishlaydi, bu esa odatda simvollar satri uchun qo`llaniladigan barcha manipulyatsiya, sortlarga ajratish va qidiruv uslublarini inkapsulyatsiya qiladi.

System.String tipining o`ziga xosliklari

S# satrlarni qayishqoq, baquvvat va qulay bo`lgan ichki qurilma tiplari kabi qayta ishlaydi. Satrning har bir ob`ekti - bu Unicode simvollarining o`zgarmas ketma-ketligidir. Boshqacha qilib aytganda, satrlarni o`zgartiruvchi uslublar haqiqatdan o`zgartirilgan nusxani qaytaradi, boshlang`ich satr esa shikastlantirilmasdan qoldiriladi.

String kalit so`zi yordamida S# da satrni e`lon qilib, siz haqiqatdan .NET Framework sinflar kutubxonasi tomonidan ta`minlanadigan ichki

qurilma tiplardan biri bo`lgan System.String tipidagi ob`ektni e`lon qilasiz. S# satri - bu System.String tipidagi ob`ekt va bu nomlardan galma-galiga butun bob davomida foydalanamiz.

System.String sinfi e`loni quyidagicha:

public sealed class String: IComparable, ICloneable, IConvertible, IEnumerable.

Bunday e`lon sinf muhrlanganligini, String sinfidan o`zining sinfini me`ros qabul qilish mumkin emasligini bildiradi. Bundan tashqari sinf to`rtta tizimli - IComparable, ICloneable, IConvertible va IEnumerable interfeysni amalga oshiradi, qaysilar System.Stringning **.NET Framework**dagi boshqa sinflar bilan birga qo`shimcha foydalanishning funksional imkoniyatlarini belgilab beradilar.

IComparable interfeysi uni qiymatlari tartiblasha oladigan tip sifatida amalga oshiruvchi tipni belgilaydi. Masalan, satrlar alfavit tartibida joylashgan bo`lishi mumkin; qaysi biri tartibga solingan ro`yxatda birinchi bo`lib turishi kerakligini aniqlash uchun ixtiyoriy satrni boshqa satrlar bilan solishtirish mumkin. IComparable sinflari CompareTo uslubini amalga oshiradi.

IEnumerable interfeysi satrlar elementlarini simvollar to`plami kabi bittadan ko`rib chiqish uchun **foreach** operatoridan foydalanishingizga yo`l qo`yishadi.

ICloneable ob`ektlari xuddi dastlabki variantdagi kabi qiymatlarga ega yangi ob`ektlar ekzemplyarlarini yaratishi mumkin. Bu holda xuddi aslidagidek simvollar to`plamiga ega yangi satrni klonlashtirish mumkin.

IConvertible sinf ob`ektlarini ichiga o`rnatilgan Tolnt32(), ToDouble(), ToDecimal() va h. k. kabi boshqa tiplarga o`zgartirishni osonlashtirish usullarini amalga oshiradilar.

Satrlar yaratish

Satrlarni yaratishning eng umumlashtirilgan usul string tipidagi ***string newString = "Yangi satr";*** ni foydalanuvchi tomonidan belgilanadigan satrli literal sifatida ma`lum bo`lgan simvollar satrini o`rnatishdan iborat.

Ko`rsatilgan satr «\n» yoki «\t» tipidagi xizmat simvollaridan tashkil topgan bo`lishi mumkin. Ular qiya chiziq (\)dan boshlanadi hamda satrni o`tkazish yoki tabulyatsiya simvolini kiritishni ko`rsatish uchun qo`llaniladi. Chapga og`gan chiziq **URL** yoki katalog yo`llari tipidagi satrlarning ayrim sintaksislarida mustaqil ishlatilganligi sababli, bunday satrda chapga og`gan chiziqdan oldin boshqa chapga og`gan chiziq simvoli turishi kerak.

Bundan tashqari satrlar so`zma-so`z yozish yo`li bilan ham yaratilishi mumkin. Bunday satrlar (@) simvolidan boshlanishi mumkin. Bu simvol String konstruktoriga satrda xizmat simvollari mavjud bo`lsa ham undan

so`zma-so`z foydalanish kerakligini ma`lum qiladi. Satrni so`zma-so`z ta`rifida chapga og`gan chiziqlar va ulardan keyin turgan simvollariga satrning qo`shimcha simvollari sifatida qaraladi. Shunday qilib, quyidagi ikkita ta`rif bir-biriga ekvivalentdir:

string stringOne = "\\\\MySystem\\MyDirectory\\MyFile.txt";

string stringTwo = @"\\MySystem\MyDirectoru \MyFile.txt";

Birinchi satrda satrning odatiy literali ishlatiladi, shuning uchun chapga og`gan chiziq (\) simvollari takrorlanishi kerak. Demak (\)ni ifodalash uchun (\\) yozib qo`yiladi. Ikkinchi satrda so`zma-so`z literalli satr ishlatiladi, shu bois chapga og`gan qo`shimcha chiziq kerak bo`lmaydi.

Keyingi misol ko`psatrli satrlarni ifodalaydi:

string stringOne ="Line One\nLine Two";

string stringTwo =@"Line One
Line Two";

Satrlarning bunday e`lonlari o`zaro almashinuvchandir. Qaysi variantdan foydalanishingiz uning qulayligiga va shaxsiy uslubingizga bog`liq.

System. Object.ToString()

Satr tuzishning boshqa uslubi - ob`ektda **ToString**() uslubini chaqirish va **string** tipidagi o`zgaruvchining natijasini o`rnatishdan iborat. Barcha ichki o`rnatiluvchi tiplarda shunday uslub mavjud, bu esa sonlarni satrli ko`rinishga o`zgartirish masalasini osonlashtiradi. Keyingi misolda **int** tipi uchun uning qiymatini satrda saqlash maqsadida **ToString**() uslubi chaqirtiriladi.

int mylnt = 10; string intString = mylnt.ToString();

myInt ob'ektidan **ToString**() uslubini chaqirish 10 sonini satrdagi ko`rinishini qaytaradi.

System.String sinfi .NET da satrli qiymatlarni turli tiplar bilan initsializatsiya qilishning xilma-xil uslublari bilan ta`minlovchi o`ta yuklangan konstruktorlarning ko`pchiligini qo`llab-quvvatlaydi. Bunday konstruktorlar-ning ayrimlari simvollar massivi yoki simvollarga ko`rsatkich ko`rinishida satrlarni hosil qilish imkoniyatini beradi. Simvollar massiv ko`rinishidagi satrni yaratishda **CLR** xavfsiz koddan foydalanib, yangi satr nusxasi hosil qilinadi. Ko`rsatkich asosida satrni hosil qilishda «xavfli» kod qo`llaniladi, buning esa .NET ilovalarini ishlab chiqishda keragi yo`q.

Satrlar bilan manipulyatsiya qilish.

String sinfi satrli qiymatlarni qidirish, taqqoslash va boshqarish uchun ko`plab ichki o`rnatilgan uslublarni ta`minlab beradi. Bunday sinfning barcha imkoniyatlarini qisqacha ro`yxati quyidagicha:

Empty() — satr bo`sh yoki bo`sh emasligini aniqlovchi xossasi;

Compare() — ikki satrni taqqoslash funksiyasi;

CompareOrdinal() — satrlarni mintaqaviy sozlashga bog`liq bo`lmagan holda taqqoslaydi;

Concat() — boshlang`ich berilgan ikki yoki undan ko`proq satrlardan yangi satr yaratadi;

Soru() — boshlang`ich berilgan satrning dublikatini hosil qiladi;

Equals()-ikki satr bir xil qiymatlarga ega yoki ega emasligini aniqlaydi;

Format() — aniq berilgan formatdan foydalanib, satrni shakllantiradi;

Intern() — satrning mavjud ekzemplyariga dalil ko`rsatmani qaytaradi;

Join() — mavjud satrning ixtiyoriy joyiga yangi satr qo`shadi;

Chars — satrlar simvollari indeksatori;

Length — satrdagi simvollar soni;

Clone() — mavjud satrga dalil ko`rsatmani qaytaradi;

CompareTo() — bir satrni ikkinchisi bilan taqqoslaydi;

CopyTo()— **Unicode** simvollar massiviga ma`lum miqdorda simvollar nusxasini ko`chiradi;

EndsWith()— satr simvollarning ma`lum ketma-ketligi bilan tugashini aniqlaydi;

Insert() — mavjud satrga yangisini kiritadi;

LastlndexOf()— elementni satrga oxirgi kirishi indeksini qaytaradi;

PadLeft() — barcha probelli yoki boshqa simvollarni o`tkazib yuborgan holda, satrni o`ng tarafidan tekislaydi;

PadRight()— barcha probelli yoki boshqa simvollarni o`tkazib yuborgan holda, satrni chap tarafidan tekislaydi;

Remove() — satrdan simvollarning kerakli sonini olib tashlaydi;

Split()— asosiy massivdan ma`lum simvol orqali ajratilgan ostsatrni qaytaradi;

StartsWith() — satr simvollarning ma`lum ketma-ketligi bilan boshlanishini aniqlaydi;

Substring() — ostsatrni simvollarning umumiy massividan qaytaradi;

ToCharArray()— satrdan simvollar massiviga ko`chiradi;

ToLower() — satrni pastki registrga o`zgartiradi;

ToUpper() — satrni yuqorigi registrga o`zgartiradi;

Trim() — satrning boshi va oxiridagi aniqlangan probellarning barchasini olib tashlaydi;

TrimEnd() — satrning oxiridagi aniqlangan probellarning barchasini olib tashlaydi;

TrimStart() — satrning boshidagi aniqlangan probellarning barchasini olib tashlaydi.

Endi satrlardan foydalanishga misol ko`rib chiqamiz. Buning uchun **Compare**(), **Concat**(), **Copy**(), **Insert**() va boshqa uslublardan foydalanadigan ilovani yozamiz.

```
using System;
namespace prinstr1{
class Program{
static void Main(){//bir necha satr yaratamiz
string strl = " abvg "; string str2 = " ABVG ";
string str3 = @"C# .NET platforma uchun ilovalarni tezlik bilan
yaratish uchun " + " asbobini ifodalaydi"; int result;
//satrlarni taqqoslash uslublari
//taqqoslash uchun Compare statik funksiyasidan foydalanamiz
result = string.Compare(strl, str2);
Console.WriteLine(@"strlni taqqoslaymiz; {0} va str2: {1}, " +
" natija;{2}\n", strl, str2, result);
// qo`shimcha parametrli Compare funksiyasidan foydalanamiz.
//satr registrini e`tiborga olmaslik uchun
result= string.Compare (strl, str2, true) ;
Console.WriteLine("Registrni hisobga olmasdan taqqoslaymiz ");
Console.WriteLine(@"strl: {0}, str2: {1}, natija: {2}\n", strl, str2,
result);
//satrlarni birlashtirish uslubi
//satrlarni birlashtirish uchun Concat funksiyasidan foydalanamiz
string str4 = string.Concat(strl, str2);
Console.WriteLine(@"strl va str2 birlashtirish orqali
str4 ni yaratamiz: \n{0}", str4);
//o`ta yuklangan operatordan foydalanamiz satrlarni birlashtirish uchun
string str5 = strl + str2;
Console.WriteLine("\nstr5 = strl + snr2  =: {0}\n", str5);
// satr nusxasini ko`chirish uchun Copy uslubidan foydalanamiz
String str6 = string.Copy(str5);
Console.WriteLine("\nstr6 nusxalandi str5: {0} dan\n", str6);
// o`ta yuklangan nusxa ko`chirish operatoridan foydalanamiz
string str7 = str6; Console.WriteLine("str7 = str6: {0}", str7);
// taqqoslashning bir necha usullari
// ob`ektning Equals uslubidan foydalanib
Console.WriteLine("\nstr7.Equals(str6)?: {0}", str7.Equals(str6));
// Equals statik uslubidan foydalanib
Console.WriteLine("\nsnr7 va strb tengmi?; {0}", string.Equals(str7,
 str6));// == operatoridan foydalanib
```

```
Console.WriteLine("\nstr7 == str6 ?; {0}", str7 == str6);
// satr uzunligini aniqlash
Console.WriteLine("\nsatr str7 {0} simvollardan iborat uzunlikka
ega. ", str7.Length);// satr simvolini uning indeksi bo`yicha aniqlash
Console.WriteLine("{0}\n simvoli str7 satrining beshinchi
elementi ", str7[4]); // satr oxirini kirish ekzemplyari bilan taqqoslash
Console.WriteLine("str3:{0}\n\"asbobini ifodalaydi\" so`zi bilan
satr tugaydimi"?: {1}\n", str3, str3.EndsWith("asbobini ifodalaydi"));
// satr oxirini kirish ekzemplyari bilan taqqoslash
Console.WriteLine("str3:{0}n\"platforma\" so`zi bilan satr tugaydimi?:
{1}\n", str3, str3.EndsWith("platformalar "));
//ostsatrni satrga birinchi kirishini qidirish
Console.WriteLine(@"asbob " + " so`zini  str3 satriga birinchi kirishi
{0}\n indeksga ega ", str3.IndexOf("asbob "));
//satrga yangi so`z kiritamiz
string str8 = str3.Insert(str3.IndexOf("ilovalar"), " quvvatli");
Console.WriteLine("str8: {0}\n", str8);Console.ReadKey();}}}
```

Dastur ishining natijasi quyidagicha bo`ladi:

strl: abvg va str2 ni taqqoslaymiz: ABVG, natija: -1

Registrni hisobga olmasdan taqqrslaymiz strl: abvg, str2: ABVG, natija: 0

strl va str2ni birlashtirib, str4 yaratamiz: abvgABVG

str5 = strl Q str2 =: abvgABVG s

tr6 nusxasi str5 dan ko`chirilgan: abvgABVG

str7 = str6: abvgABVG

str7.Eguals(str6)?: True

str7 va str6 tengmi?: True

str7 qq str6 ?: True

str7 satr uzunligi 8 simvollardan iborat.

str7 satridagi beshinchi element A simvoli bo`ladi

str3:C# .NET platforma uchun ilovalarni tezlik bilan yaratish asbobi bo`lib xizmat qiladi

Bu satr "asbob" so`zi bilan tugaydimi?: False

Bu satr "platforma" so`zi bilan tugaydimi "?: True

Asbob so`zini str3 satriga birinchi kirishi 22 indeksga ega

str8: C# .NET platforma uchun quvvtli ilovalarni tezlik bilan yaratish asbobi bo`ladi

Satrlarni e`lon qilishning turli usullaridan foydalanishingiz mumkin. **str3** satrini e`lon qilish uchun men satrni so`zma-so`z ko`rsatish usulidan

foydalandim. Shuni aytish kerakki, satrni boshqa qatorga ko`chirishda dastur kodida bo`lishingiz mumkin. Bunda (+) operatori yordamida satrning uzilgan qismlarini birlashtirish zarur. Bunday satr bir butun deb qabul qilinadi.

string str3 =@"S#. NET platformaga tezlik bilan ilovalarni yaratish asbobi" + " bo`ladi "; result = string.Compare(strl, str2);

Console.WriteLine(@"taqqoslaymiz strl: {0} i str2: {1}, "+ "natija: {2}\n", strl, str2, result);

Bu holda registrga bog`liq bo`lgan ikki satrni taqqoslash funksiyasi qo`llaniladi. Taqqoslash funksiyasi doimo taqqoslash natijalariga ko`ra turli qiymatlarni qaytaradi:

• no`ldan kichik qiymatni, agar birinchi satr ikkinchiga nisbatan kichik bo`lsa;

• 0, agar satrlar teng bo`lsa;

• no`ldan katta qiymatni, agar birinchi satr ikkinchiga nisbatan katta bo`lsa.

Bizning misolda natija quyidagicha bo`ladi: **strl** ni va **str2** ni taqqoslaymiz: **abvg** va **ABVG**, natija: -1

Quyi registr xarflari yuqoridagiga nisbatan kichikroq qiymatga ega, natija shundan kelib chiqadi.

Compare ning keyingi funksiyasida registrni hisobga olmasdan taqqoslaymiz. Bundan funksiyaning uchinchi qo`shimcha parametri true bundan dalolat beradi.

result =string.Compare(strl, str2, true);

Console.WriteLine ("Registrni hisobga olmasdan");

Console.WriteLine(@"strl: {0}, str2: {1}, " + "natija: {2}\n", strl, str2, result); Tegishlicha natija ham quyidagicha bo`ladi:

strl ni hisobga olmasdan taqqoslaymiz: abvg, str2: ABVG, natija: 0

Registrni hisobga olmasdan taqqoslash funksiyasi dastlab ikkala satrni umumiy registrga olib, keyin esa satrlarni simvollab taqqoslashni amalga oshiradi. Natijada amallar ketma-ketligiga erishamiz:

abvg -> ABVG, ABVG = ABVG ? -» DA -» natija 0

Satrlarni birlashtirish uchun string sinfining ikki xil imkoniyatlaridan foydalandik. Bulardan bittasi – statik funksiya Concat() dan foydalanish.

string str4 = string.Concat(strl, str2);

Ikkinchi usul - (+)operatorini qo`llash.

string str5 =strl + str2;

string sinfining (+)operatori shunday yuklanganki, Concat() funksiya-siga o`xshash amalni bajaradi. Ammo strl+str2 yozuvdan foydalanishda

yaxshiroq o`qiladi, shuning uchun dasturchilar odatda funksiyalarni chaqirish operatorlaridan foydalanishni afzal ko`radilar.

Xuddi shunday funksiya Copy() va operator (=) larni ham taqqoslash mumkin. Ular bir xil amalni bajaradilar – bitta satrdagilarni nusxasini ikkinchisiga ko`chiradilar. Farq faqat dastur kodini yozishda:

string str6 = string.Copy(str5);

string str7 = str6;

str6 va **str7** natijada str5 satrida yozilgan qiymatga ega bo`ladilar. **string** S# sinfida ikki satr tengligini nazorat qilishning uchta usulini ta`minlaydi. Birinchisi- bu **Equals**() uslubidan foydalanish. str7.Equals(str6)

Bunday holda **str7** ob`ekti uchun unga ob`ekt **str6** ni tengligi tekshiriladi. Satrlar tengligini tekshirishning ikkinchi usuli-bu statik funksiya stringni qo`llash.

string.Equals (str7, srr6)

Uchinchi variant - bu taqqoslash operatori (==)ni qo`llash.

str7 == str6

Bu chaqirishlarning hammasi **True** ni qaytaradi, agar satrlar teng bo`lsa. Agar satrlar teng bo`lmasa **False** ni qaytaradi.

Length ni xossasi satrlardagi simvollar sonini qaytaradi. Operator ([]) esa–tegishli indeksga ega satr simvolini qaytaradi.

Console.WriteLine("str7 satrining 5 chi elementi bo`lib simvol {0}\n xizmat qiladi.", str7[4]);

Bu yerda biz ([]) operatordagi 4 soni yordamida satrning beshinchi elementini olishga harakat qilamiz. Chunki S# dagi satr elementlari, C++ dagi kabi, no`l indeksidan boshlanadi.

Nimsatr qidiruvi

String tipi satrdan nimsatrni chiqarish uchun o`ta yuklangan Substring() uslubiga ega. Uslublardan biri qaysi elementdan boshlab nimsatr chiqarilishi kerak bo`lsa, shu element indeksi parametr sifatida qabul qilinadi. Ikkinchi uslub qidirishni qayerdan boshlab, qayerda tugatishni ko`rsatish uchun ham boshlang`ich, ham oxirgi indeksni qabul qiladi. Substring uslubini keyingi misolda ko`rish mumkin. Dastur satr so`zlarini yozilish tartibiga teskari tartibda chiqaradi:

```
using System;namespace primstr2{
class Program{ static void Main(){
// qayta ishlash uchun satrni e`lon qilamiz
string s1 = " Bir Ikki Uch To`rt ";
int ix = s1.LastIndexOf(" "); //oxirgi probel indeksi kelib chiqadi
string s2 = s1.Substring(ix + 1); //satrda oxirgi so`z kelib chiqadi
```

```
//0-chi indeksdan boshlanayotgan va
//oxirgi probel bilan tugaydigan  nimsatrni s1 ga o`rnatamiz
s1 = s1.Substring(0, ix);
ix = s1.LastIndexOf(" "); //oxirgi probel indeksini qayta qidiramiz
string s3=s1.Substring(ix + 1);//s3 ni satrning oxirgi so`ziga o`rnatamiz
// s1 ni nimsatrga tashlaymiz no`l simvolidan to ix gacha
s1 = s1.Substring(0, ix);
ix=s1.LastIndexOf(" ");//ix ni "bir" va "ikki" probel orasiga tashlaymiz
string s4=s1.Substring(ix +1);//ix dan keyin nimsatrga s4 ni o`rnatamiz
// s1 ni nimsatrga 0 dan ix gacha o`rnatamiz
// faqat "bir" so`zini olamiz
s1 = s1.Substring(0, ix);
// oxirgi probel indeksini olishga  harakat qilamiz
// ammo bu gal funksiya LastlndexOf -1 ni qaytaradi
ix = s1.LastIndexOf(" ");
// ix-M dan boshlab nim satrga s5 ni o`rnatamiz
// ix = 1 bo`lgani uchun, s5 s1 boshiga o`rnatiladi
string s5 = s1.Substring(ix + 1);
// Natijani chiqaramiz
Console.WriteLine("s2: {0}\ns3: {1}", s2, s3);
Console.WriteLine("s4: {0}\ns5: {1}\n", s4, s5);
Console.WriteLine("s1: {0}\n", s1);
Console.ReadKey();}}}
```

Birinchi bo`lib satrni e`lon qilamiz va uni zarur parametrlar bilan initsiallashtiramiz.

```
string s1 = " Bir Ikki Uch To`rt ";
```

So`ng satrdagi oxirgi probel pozitsiyasini hisoblaymiz. Bu **int ix = s1.LastlndexOf(" ");** satri oxirgi so`zining boshini aniqlash uchun zarur.

Bu holda **ix** qiymati 12 ga teng bo`ladi. «To`rt» so`zi pozitsiya 13 dan boshlanadi. Endi, satrning oxirgi so`zi boshini bilganimizda, uni chiqarish mumkin.

```
string s2 = s1.Substring(ix+1);
```

Natijada, **s2** «To`rt» ga teng bo`ladi. Shundan so`ng biz **s1** bosh-lang`ich satrdagi «To`rt» so`ziga kesamiz. Buning uchun Substring funksiyasini ikkala parametri – nimsatrning boshlanishi va oxirini chaqira-miz. Bizda satr boshi bo`lib boshlang`ich satr boshi xizmat qiladi, oxirgi probel indeksi esa - satr oxiri bo`lib.

```
s1 = s1.Substring(0, ix);
```

Yangi satr «Bir Ikki Uch» ko`rinishiga ega bo`ladi. Endi, avvalroq, butun satrdagi kabi xuddi shu amallar ketma-ketligini qaytaramiz. Oxirgi

probel indeksini olamiz, oxirgi so`zni tanlaymiz, satrni kesamiz. Buni satrda bitta «Bir» so`zi qolguncha bajaraveramiz. Bu satrdan probel simvoli indeksini olishga harakat qilganimizda, funksiya -1 qiymatini qaytaradi.

ix = si.LastlndexOf(" ");

Shuning uchun, funksiya **Substring**() va unga (-1 + 1 = 0) qiymatini bersak, bizga to`liq boshlang`ich satr qaytariladi, ya`ni «Bir» so`zi.

string s5 = si.Substring(ix+1);

Dastur ishining natijasi quyidagicha:

s2: To`rt
s3: uch
s4: ikki
s5: Bir

s1: Bir

Satrlarni parchalash

Aslida, avvalgi misol satrni so`zlarga ajratar edi, topilgan so`zlarni saqlab, ularni ekranga chiqaradi. Avvalgi misolda namoyish etilgan muammoning samaraliroq yechimi **string** sinfidagi **Split**() uslubidan foydalanishdan iborat. **Split**() uslubi satrni nim satrlarga ajratadi. Boshlang`ich satrdan nimsatrlarni ajratib olish uchun **Split**() uslubiga parametr sifatida ajratish simvolini uzatish zarur. Bunda natija satrlar massivi ko`rinishida qaytariladi. Keling, **Split**() uslubi ishini misolda ko`rib chiqamiz:

```
using System;
namespace primstr3{
class Program{
static void Main(string[] args){
// taxlil uchun satr
string s1 = " Bir, Ikki, Uch, ajratish uchun satr ";
// tahlil uchun ajratish simvollarini beramiz
const char cSpace = ' '; const char cComma = ',';
// sozdayem ajratish simvollari massivini yaratamiz
// va uni ikki element bilan initsiallashtiramiz
char[] delimiters = new char[] { cSpace, cComma };
string output = "";int ctr = 1;
// nimsatrlarni ajratish satr asosida ajratamiz
// va natijani saqlab qolamiz
foreach (string substring in s1.Split(delimiters)){
output += ctr++; // nimsatr raqami
```

```
output += ": ";output += substring; // nimsatr
output += "\n"; // yangi liniyaga o`tish}
// natijani chiqarish
Console.WriteLine(output);
Console.ReadKey();
}}}
```

Matn dastur ishining natijasi bo`ladi:

1: Bir
2: Ikki
3: Uch
4:
5: Satr
6: ajratish
7: uchun

4 chi raqamli satr bo`shligiga e`tibor bering - bu xato emas. Uni o`ziga xosligini hozir ko`rib chiqamiz.

```
string s1 = " Bir,Ikki,Uch, Tahlil uchun satr ";
```

6 so`zli **s1** satrni e`lon qildik. Satrda ajratish simvollarning ikki tipi ishlatiladi. Ulardan biri - probel simvoli, ikkinchisi - (,) simvoli. «Uch» va «Satr» So`zlari oralig`ida birdaniga ikkita simvol, vergul va probel joylashadi. Buni ta`kidlab o`tish juda muhimdir!

Keyin dasturda ajratish simvollarini aniqlovchi ikkita konstanta belgilanadi

```
const char cSpace = ' ' ;
const char cComma = ', ' ;
```

Bu konstantlar bitta massivga birlashadi.

```
char[] delimiters = new char[] ( cSpace, cComma };
```

Split() uslubi parametr sifatida bitta simvolni ham, ajratish simvollarini ham birdek qabul qilishi mumkin. Misolimizda parametr sifatida ikki element -probel va vergul simvollaridan tashkil topgan ajratish simvollaridan foydalanamiz. Shunday qilib, Split() uslubi boshlang`ich satrda bu simvollardan birini topsa, zudlik bilan chiqish massiviga avvalgi ajratuvchidan joriy pozitsiyagacha simvollar ketma-ketligini joylashtiradi.

```
foreach (string substring in si.Split(delimiters))
```

Bu holda funksiya **Split**() qaytaradigan yakunlovchi satrlar massivining barcha elementlariga **foreach** operatori qo`llaniladi. Yakunlovchi massiv yettita elementdan tashkil topadi va ulardan bittasi bo`sh satr bo`ladi. Bo`sh satr tahlil qilinuvchi satrning o`ziga xos xususiyatlaridan kelib chiqqan. Yuqorida qayd qilinganidek «Uch» va «Satr» so`zlari oralig`ida bir vaqtda ikkita ajratuvchi simvollar vergul hamda probel joylashadi.

Split() uslubi «Uch» va «Satr» oralig`idagi probel simvoli tahliligacha yetib borganda, u probelni ajratuvchi simvol sifatida aniqlaydi. Ammo bevosita uning oldida yana bitta ajratuvchi simvol turadi. Demak, ikkita ajratuvchi simvollar orasida uzunligi 0 simvollarga teng bo`sh satr joylashadi, uni **Split**() uslubi chiqish massiviga joylaydi.

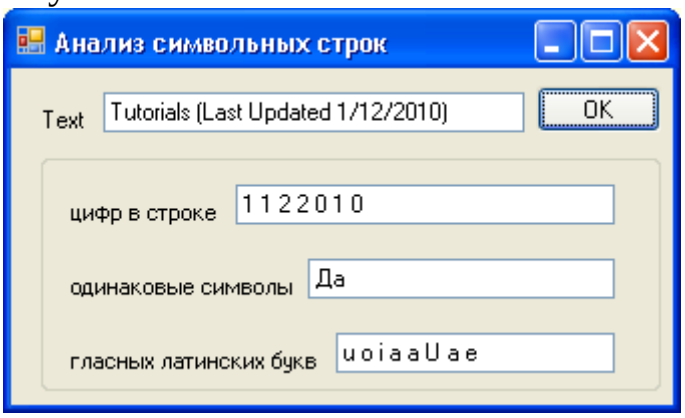
Simvolli satrlarni tahlili

Misol	Simvollar satrini kiritish. Aniqlang: 1. Satrda nechta son borligini. 2. Berilgan satrda bir xil simvollar borligini. 3. Barcha unsiz lotin xarflari ro`yxatini tuzilishini.

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text; /* Kiritilgan satr */
int n = s.Length;
textBox2.Clear();
/* Sonlarni qidirish uchun satrning hamma simvollari bo`yicha sikl */
for (int i = 0; i < n; i++)
if (char.IsDigit(s[i])) textBox2.Text += s[i] + " ";
textBox3.Text = "Net";
/* Bir xil simvollarni  qidirish uchun satrning hamma simvollari
 bo`yicha takrorlanuvchi jarayon */
for (int i = 0; i < n; i++)
if (s.IndexOf(s[i]) != s.LastIndexOf(s[i])){
textBox3.Text = " Ha ";
break;
}
textBox4.Clear();
/* Unsiz lotin xarflari qidirish uchun satrning hamma simvollari
bo`yicha takrorlanuvchi jarayon */
for (int i = 0; i < n; i++)
switch (s[i]){
case 'A':case 'E': case 'I': case 'O': case 'U': case 'Y':
textBox4.Text += s[i] + " "; break;
case 'a': case 'e': case 'i': case 'o': case 'u': case 'y':
textBox4.Text += s[i] + " "; break;
}
if (textBox4.Text == "") textBox4.Text = " Lotin unsizlari yo`q ";
}
```

Loyiha shakli darchasi



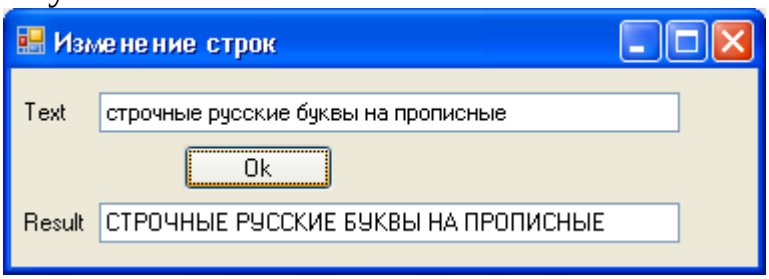
Satrlarni o`zgartirish (almashtirish, olib tashlash, simvollarni kiritish)

Misol	Kiritilgan satrda kichik kirill alifbosidagi harflarni bosh harflarga almashtirish.

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text; textBox2.Text = s.ToUpper();}
```

Loyiha shakli darchasi



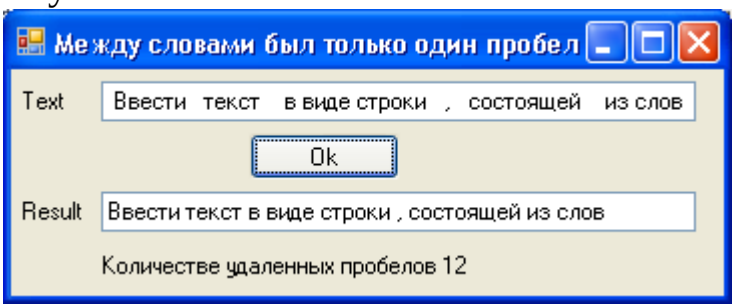
Misol	Bir yoki undan ko`proq probellar bilan ajratilgan so`zlardan iborat satr ko`rinishida matn kiriting. Satrni shunday o`zgartirish kerakki, so`zlar orasida faqat bitta probel bo`lsin. Olib tashlangan probellar haqida ma`lumotni chiqaring.

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text;
s = s.Trim();
string s1 = ""; int n = s.Length;
for (int i = 0; i < n; i++)
if (s[i] != ' ') s1 += s[i];
else if (s[i + 1] != ' ') s1 += ' ';
int k = n - s1.Length;
```

```
textBox2.Text = s1;
label3.Text = " Olib tashlangan probellar soni " + k.ToString();}
```

Loyiha shakli darchasi

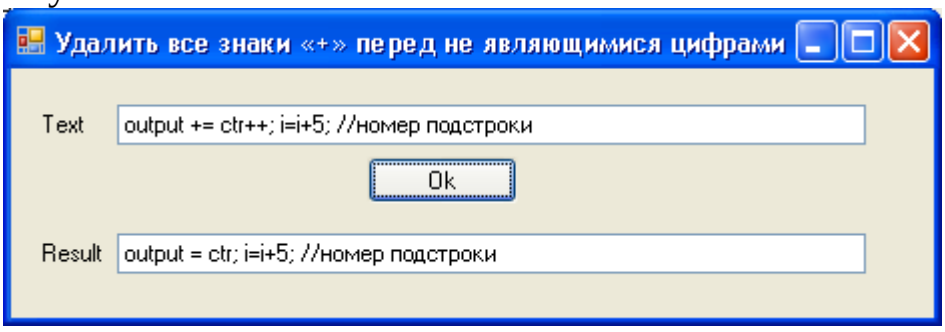


Misol	254 simvol uzunligidagi satr. Son bo`lmagan barcha simvollar oldidan «+» belgilarni olib tas'halang.

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text;
int n = s.Length; s = s + " ";
string s1 = "";
for (int i = 0; i < n; i++)
if (s[i] == '+'){
if (char.IsDigit(s[i+1]))s1 += s[i];/*Raqam oldidan ko`chirib yozish */}
else s1 += s[i]; /* agar u bo`lmasa ixtiyoriy simvolni ko`chirib yozish*/
textBox2.Text = s1;}
```

Loyiha shakli darchasi



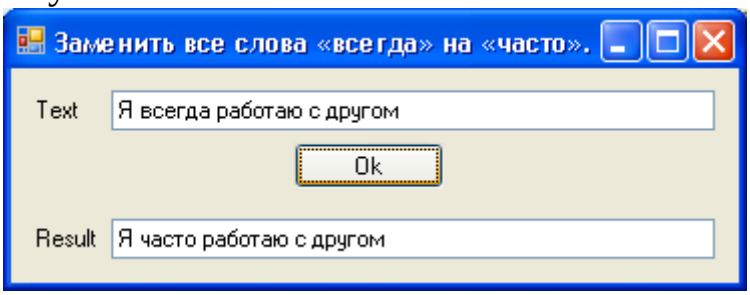
Misol.	Berilgan satrdagi "doim" so`zlarini "tez- tez" ga almashtirish

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text;
string s2 = s.Replace("doim ", " tez- tez ");
```

textBox2.Text = s2;}

Loyiha shakli darchasi

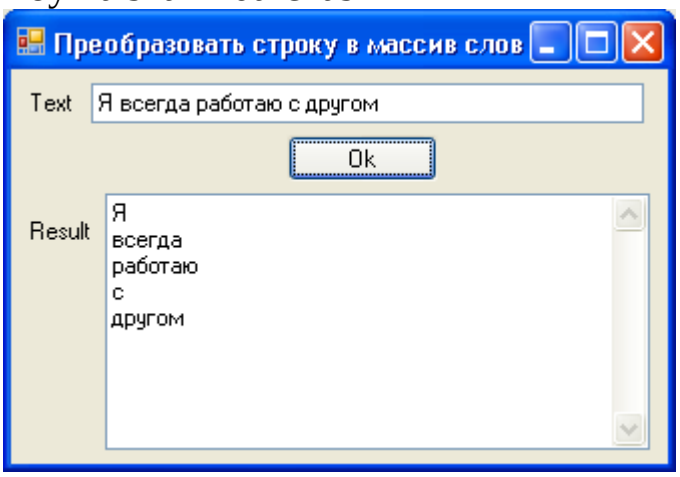


Misol	Probellar bilan ajratilgan so`zlardan iborat satr berilgan. Satrni so`zlar massiviga o`zgartirish.

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text;
string[] s1 = s.Split(' '); textBox2.Lines = s1;}
```

Loyiha shakli darchasi

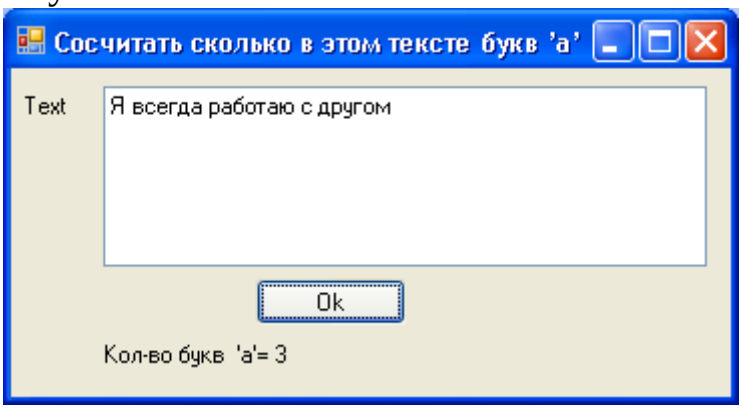


Misol	TextBox komponenti yordamida (Multilineqtrue) bir necha satrdan iborat kirill imlosidagi matnni kiritish. Matnda nechta ’a’ harflar borligini hisoblang.

Dastur matni lavhasi

```
private void button1_Click(object sender, EventArgs e){
string s = textBox1.Text;
int n = s.Length;int k = 0;
for (int i = 0; i < n; i++)
if ((s[i] == 'a')||(s[i] == 'a')) /*kirill yoki lotin alifbosi harfi 'a'*/
k++; label2.Text = " 'a' harflari soni = "+ k.ToString();}
```

Loyiha shakli darchasi



8-laboratoriya ishi

Fayllar

Kutubxona sinflari fayl, katalog va operativ xotira xududlari bilan turli rejimlarda ishlashga imkoniyat yaratadi.

1-jadvalda sinflarning qisqa bayoni keltirilgan.

1-Jadval

Sinf	Bayon
BinaryReader, BinaryWriter	Taqdimotning ichki shaklida oddiy ichki qurilma tiplar (butun sonli, logik, satrli va sh. o`. larni) qiymatini o`qish va yozish
BufferedStream	Baytlar oqimini vaqtincha saqlash (masalan, keyinchalik doimiy saqlash joyiga o`tkazish uchun).
Directory, DirectoryInfo, File, FileInfo	Fizik fayl va kataloglar bilan ishlash: yaratish, o`chirish xossalarni qabul qilish. **File** va **Directory** imkoniyatlari asosan statik uslublar ko`rinishida amalga oshiriladi. Shularga o`xshash **Directorylnfo** va **FileInfo** sinflari odatiy uslublardan foydalanadilar
FileStream	Baytlar oqimi ko`rinishidagi faylga ixtiyoriy (to`g`ridan-to`g`ri) kirish
MemoryStream	Operativ xotiradagi baytlar oqimiga ixtiyoriy kirish
StreamWriter, StreamReader	Matnli axborotni fayldan o`qish va faylga yozish (ixtyoriy kirishga yo`l qo`yilmaydi)
StringWriter, StringReader	Operativ xotirada matnli axborot bilan ishlashga yo`l qo`yilmaydi

Jadvaldan ko`rinib turibdiki tashqi qurilma bilan almashinuvni quyidagilarda amalga oshirish mumkin:

• berilgan ma`lumotlarni ikkilangan taqdimoti (**BinaryReader**, **BinaryWriter**);

• baytlarda (**FileStream**);

• matnda, ya`ni simvollarda (**StreamWriter**, **StreamReader**).

.NET da **Unicode** kodlashdan foydalaniladi, qaysinda har bir simvol ikkita bayt bilan kodlanadi. Matn bilan ishlaydigan sinflar baytlardan foydalanadigan sinflarning qobig`i bo`lib, avtomatik ravishda baytlardan simvollarga va aksincha qayta kodlashni bajaradilar.

Ikkilangan va bayt oqimlari ma`lumotlarni operativ xotiradagidek ko`rinishda saqlashadi, ya`ni fayl bilan almashinuv jarayonida axborotlarni bitlar bo`yicha nusxalash amalga oshiriladi. Ikkilangan fayllar odam ko`rib chiqishi uchun emas, balki dasturlarda foydalanish uchun qo`llaniladi.

Matnli fayllar faqat ketma-ketlikda kirishni amalga oshirishga imkoniyat beradi, ikkilangan va bayt oqimlarida ikkala uslubdan foydalanish mumkin.

To`g`ridan-to`g`ri kirish o`zgartirishlarni bo`lmasligi bilan birga kerakli axborot olishda yuqori tezlikni ta`minlaydi.

Fayl oqimlari bilan ishlashni eng oddiy usullarini ko`rib chiqamiz. Dasturda fayl oqimlari sinflaridan foydalanish quyidagi operatsiyalarni ko`zda tutadi:

1. Oqimni yaratish va uni fizik fayl bilan bog`lash.
2. Almashuv (kiritish-chiqarish).
3. Faylni yopish.

Fayl oqimlarining har bir sinfida konstruktorlarning bir nechta varianti mavjud, qaysilar yordamida turli usullar bilan va turli rejimlarda shu sinflar ob`ektlarini yaratish mumkin.

Masalan, fayllarni faqat o`qish uchun, faqat yozish uchun yoki o`qish va yozish uchun ochish mumkin. Fayllarga kirishning rejimlari **System.IO** nomlar makonida belgilangan **FileAccess** ro`yxatida mavjud.

Ro`yxat doimiylari 2- jadvalda keltirilgan.

2-jadval

Qiymatlar	Bayoni
Read	Faylni faqat o`qish uchun ochish
ReadWrite	Faylni o`qish va yozish uchun ochish
Write	Faylni faqat yozish uchun ochish

Faylni ochish imkoniyatiga ega rejimlar **FileMode** ro`yxatida belgilangan (3- jadvalda).

3-jadval

Qiymatlar	Bayoni
Append	Faylni ochish, agar u mavjud bo`lsa hamda fayl oxirida joriy ko`rsatkichni o`rnatish. Agar fayl mavjud bo`lmasa, yangi fayl yaratish
Create	Yangi fayl yaratish. Agar katalogda xuddi shunday nomli fayl mavjud

	bo`lsa, u o`chiriladi
CreateNew	Yangi fayl yaratish. Agar katalogda xuddi shunday nomli fayl mavjud bo`lsa, IOException istisnosi paydo bo`ladi
Open	Mavjud faylni ochish
OpenOrCreate	Faylni ochish, agar u mavjud bo`lsa. Agar yo`q bo`lsa, shunday nomli fayl yaratish
Truncate	Mavjud faylni ochish. Ochilgandan so`ng u nol uzunlikkacha kesib tashlanishi kerak

4- jadval

Qiymat	Bayon
None	Ochiq faylni birgalikda ishlatish man etiladi. Ushbu faylni ochishga talab xato haqida ma`lumot berish bilan tugatiladi
Read	Bir vaqtni o`zida bir nechta foydalanuvchi faylni o`qish uchun ochishiga imkon beradi. Agar bunday bayroq o`rnatilmagan bo`lsa, faylni o`qish uchun ochishga talablar xato haqida ma`lumot berish bilan tugatiladi
ReadWrite	Bir vaqtning o`zida bir nechta foydalanuvchiga faylni o`qish va yozish uchun ochishga imkon beradi
Write	Bir vaqtning o`zida bir nechta foydalanuvchiga faylni yozish uchun ochishga imkon beradi

Baytlar oqimi

Baytlar darajasida faylga kiritish-chiqarish oqimlar bilan operatsiya-larni amalga oshiruvchi **Stream** abstrakt sinfi vorisi bo`lgan **FileStream** sinfi yordamida bajariladi.

Stream sinfi elementlari 5-jadvalda keltirilgan.

5-jadval

Element	Bayon
BeginRead, BeginWrite	Asinxronkiritish yoki chiqarish
CanRead, CanSeek, CanWrite	Oqim qo`llaydigan operatsiyalar turini belgilovchi xususiyatlar: o`qish, to`g`ridan-to`g`ri kirish va yoki yozish
Close	Joriy oqimni yopish va u bilan bog`liq resurslarni ozod etish (soketlar, fayllarga ko`rsatuvchilar va h. k.)
EndRead, EndWrite	Asinxron kiritishni tugashini kutish; asinxron chiqarishni tugatish
Flush	Oqim bilan bog`liq berilgan ma`lumotlar manbasiga ma`lumotlarni buferdan yozib olish va buferni tozalash. Agar shu oqim uchun buferdan foydalanilmasa, bu uslub ishlamaydi
Length	Oqim uzunligini baytlarda qaytarish
Position	Oqimdagi joriy pozitsiyani qaytarish
Read, ReadByte	Joriy oqimdagi baytlar ketma-ketligini hisoblash va oqimdagi ko`rsatkichni hisoblangan baytlar soniga ko`chirish
Seek	Oqimning joriy ko`rsatkichini berilgan pozitsiyaga ko`chirish
SetLength	Joriy oqim uzunligini belgilash
Write, WriteByte	Joriy oqimdagi baytlar (yoki bitta bayt) ketma-ketligini hisoblash va oqimdagi ko`rsatkichni yozilgan baytlar soniga ko`chirish

FileStream sinfi bu elementlarni diskli fayllar bilan ishlash uchun amalga oshiradi. Fayl bilan ish rejimlarini belgilash uchun **FileMode**, **FileAccess** va **FileShare** standart ro`yxatlardan foydalaniladi. Bu ro`yxatlarning qiymatlari jadvalda keltirilgan.

1- lavhada fayl bilan ishlashga misol keltirilgan. Misolda bir bayt va baytlar massivi o`qish va yozish hamda oqimdagi pozitsionlash namoyish etilgan.

1-lavha. Baytlar oqimidan foydalanishga misol

```
using System;using System.IO;
namespace ConsoleApplication1{
class Classl{
static void Main(){
FileStream f=new FileStream("test.txt",
FileMode.Create, FileAccess.ReadWrite);
f.WriteByte(100); //  fayl boshiga 100 soni yoziladi
byte[] x = new byte[10];
for (byte i = 0; i < 10; ++i){
x[i] = (byte)(10 - i);f.WriteByte(i); //0 dan 9 gacha 10 son yoziladi}
f.Write(x, 0, 5); // massivning 5 elementi yoziladi
byte[] y = new byte[20];
f.Seek(0, SeekOrigin.Begin);// boshini ko`rsatuvchi joriy ko`rsatkich
f.Read(y, 0, 20); // fayldan massivga o`qish
foreach (byte elem in y) Console.Write(" " + elem);
Console.WriteLine();
f.Seek(5, SeekOrigin.Begin);// 5-elementga joriy ko`rsatkich
int a = f.ReadByte(); //5-elementni o`qish
Console.WriteLine(a); a = f.ReadByte(); // 6- elementni o`qish
Console.WriteLine(a);
Console.WriteLine("Oqimdagi joriy holati " + f.Position);
f.Close();Console.ReadKey();}}}
```

Dastur ishi natijasi:

100 0 1 2 3 4 5 6 7 8 9 10 9 8 7 6 0 0 0 0

4

5

oqimdagi joriy holati7

Oqimdagi joriy pozitsiya dastlab fayl boshiga o`rnatiladi (**Append** dan tashqari ixtiyoriy ochish rejimi uchun) va har bir bayt yozilganida bitta pozitsiyaga suriladi.

Muddaodagi o`qish pozitsiyasini o`rnatish uchun ikki parametr: ikkinchisi tomonidan belgilangan hisob boshiga nisbatan birinchisi bergan baytlardagi surilishdan iborat Seek uslubidan foydalaniladi. Sanoq boshi SeekOrigin ro`yxati konstantalari tomonidan belgilanadi:

Fayl boshi - Begin, joriy pozitsiya - Current va fayl oxiri - End.

Bu misolda fayl joriy katalogi yaratilgan.

Masalan: FileStream f=new FileStream("test.txt", FileMode.Create, FileAccess.ReadWrite);

So`zma-so`z literallarda teskari qiya chiziqni qaytarishni keragi bo`lmaydi.

Fayllarni ochish operatsiyalari muvaffaqiyatsiz tugashi mumkin. Masalan, mavjud fayl nomidagi xato yoki diskda bo`sh joy bo`lmaganida, shuning uchun har doim bu operatsiyalar natijasini nazorat qilish tavsiya etiladi.

Matnli fayllar

StreamWriter va **StreamReader** simvol oqimlari **Unicode**-simvollari bilan ishlaydilar, demak, odam o`zlashtirishi uchun mo`ljallangan fayllar bilan ishlashda ulardan foydalanish qulayroq. Bu oqimlar ularning ishchanligini ko`p qismini ta`minlovchi, tegishlicha, **TextWriter** va **TextReader** sinflarning vorisidir. 6 va 7-jadvallarda bu sinflarning eng muhim elementlari keltirilgan. Ko`rib turganingizdek, matnli fayllarga ixtiyoriy kirish qo`llab quvvatlanmaydi.

6-jadval

Element	Bayoni
Close	Fayllarni yopish va bu bilan bog`liq resurslarni ozod etish. Agar yozish jarayonida buferdan foydalanilsa, u avtomatik ravishda tozalanadi.
Flush	Barcha buferlarni joriy fayl uchun fayl tozalash va ularda to`plangan ma`lumotlarni doimiy saqlash joyiga yozish. Faylning o`zi shunda yopilmaydi.
NewLine	Yangi satr boshini anglatuvchi simvollar ketma-ketligini berish uchun ishlatiladi. Indalmaganda (\r\n) ketma-ketligi ishlatiladi.
Write	Matnning bir qismini oqimga yozish.
WriteLine	Satrni oqimga yozish va boshqa satrga o`tish.

7-jadval

Element	Bayoni
Peek	Fayldagi ko`rsatkich pozitsiyasini o`zgartirmasdan keyingi simvolni qaytarish
Read	Kirish oqimidagi ma`lumotlarni hisoblash
ReadBlock	Kirish oqimidan foydalanuvchi tomonidan ko`rsatilgan simvollar sonini hisoblash (o`qish) va birinchi pozitsiyadan boshlab, buferga yozish
ReadLine	Joriy oqimdagi satrni o`qish va **string** tipidagi qiymat sifatida qaytarish. (**null**) bo`sh satr fayl tugaganini anglatadi
ReadToEnd	Joriy pozitsiyadan boshlab oqim oxirigacha barcha simvollarni o`qish va o`qilgan ma`lumotlarni **string** tipidagi satr sifatida qaytarish

Ushbu jadvallarda keltirilgan ayrim uslublar bilan tanishsiz: kitobning boshidan uzluksiz matnli oqimlardan o`qish va bu oqimlarga yozish uslublaridan foydalanildi, ammo diskli fayllar uchun emas, balki ularning alohida bir ko`rinishi bo`lgan konsol uchun.

3 - lavhada ikki satr yozilgan matnli fayl yaratiladi.

Ikkinchi satr qayta o`zgartirilgan o`zgaruvchilarning soniy qiymatlari va tushuntiruvchi matnlardan shakllantiriladi. Fayl tarkibini ixtiyoriy matn muharririda ko`rib chiqish mumkin. Fayl muhitida bajariladigan faylni yozib qo`yadigan katalogda yaratiladi. Indalmaganda bu ...\ConsoleApplication1\bin\ Debug katalogidir.

3-lavha. Matnli faylga chiqarish.

```
using System;
using System.IO;
namespace ConsoleApplication1{
class Classl{static void Main(){
try{StreamWriter f = new StreamWriter("text.txt");
f.WriteLine("Matnli faylga chiqarish:");
double a = 12.234;
int b = 29;
f.WriteLine("a = {0:6:C} b = {1:2:X}", a, b);
f.Close();
}catch (Exception e){
Console.WriteLine("Error: " + e.Message);
return;}}}}
```

4 - lavhada oldingi lavhada yaratilgan fayl ekranga chiqariladi.

4 - lavha. Matnli faylni o`qish.

```
using System;
using System.IO;
namespace ConsoleApplication1{
class Class1 {
static void Main(){
try{StreamReader f = new StreamReader("text.txt");
string s = f.ReadToEnd();
Console.WriteLine(s);
f.Close();}catch (FileNotFoundException e){
Console.WriteLine(e.Message);
Console.WriteLine("Fayl nomi to`g`riligini tekshirish ");
return; }}}}
```

Bu dasturda butun fayl **ReadToEnd** uslubi bo`yicha bitta matnli o`zgaruvchiga o`qib olinadi. Ko`pincha faylni satrma-satr o`qish zarurati tug`iladi, bunga misol 11.5–lavhada keltirilgan. Har bir satr chiqishda raqamlanadi.

5-lavha. Matnli faylni satrma-satr o`qish

```
using System;
using System.IO;
namespace ConsoleApplication1{
class Classl {
static void Main(){
try {
StreamReader f = new StreamReader( "text.txt" );
string s;
long i = 0;
while ((s = f.ReadLine()) != null )
Console.WriteLine( "{0}: {1}", ++i, s );
f.Close();
}catch( FileNotFoundException e ){
Console.WriteLine( e.Message );
Console.WriteLine("Fayl nomi to`g`riligini tekshiring!" );
return;
} catch ( Exception e ){
Console.WriteLine( "Error: " + e.Message );
return;
}}}}
```

Matnli fayldagi sonlarni ularni ifodalashning ichki shakliga qayta o`zgartirish 6-lavhada keltirilgan. Dasturda har bir satr yig`indisi keltiriladi.

Fayl ichidagilarga ancha qat`iy cheklovlar qo`yiladi: sonlar bir dona probel bilan ajratilgan bo`lishi kerak, oxirgi sondan keyin probel qo`yilmaydi, fayl satrni ko`chirish simvoli bilan tugatilmasligi kerak.

Satrni bo`laklarga ajratish va butunsonli ko`rinishga qayta o`girish avval ko`rib chiqilgan.

6 - lavha. Satrlarni qayta o`zgartirish.

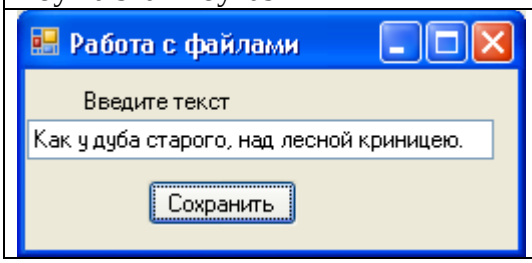
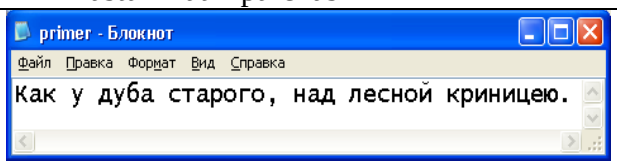
```
using System;
using System.IO;
namespace ConsoleApplication1{
class Class1{
static void Main(){
try{
```

```
StreamReader f = new StreamReader( "numbers.txt" );
string s;
const int n = 20;
int[] a = new int[n];
string[] buf;
while ( ( s = f.ReadLine() ) != null ){
buf = s.Split(' ');
long sum = 0;
for ( int i = 0; i < buf.Length; ++i ){
a[i] = Convert.ToInt32( buf[i] );
sum += a[i]; }
Console.WriteLine( "{0} summa; {1}", s, sum ); }
f.Close();
} catch( FileNotFoundException e ){
Console.WriteLine( e.Message );
Console.WriteLine( "Fayl nomi to`g`riligini tekshiring!" );
return;
}catch ( Exception e ){
Console.WriteLine( "Error; " + e.Message );
return;
}}}}
```

Dastur ishi natijasi:

12 4 summa: 7
3 44 -3 6 summa: 50
8 11 summa: 10

Misol	TextBox oynasiga kiritilgan matnni faylga yozish.

Loyiha shakli oynasi	Dastur matni parchasi
	

Dastur matni parchasi

```
private void button1_Click(object sender, EventArgs e){
FileStream f=File.Open("primer.txt",FileMode.Create,
FileAccess.Write);
```

```
StreamWriter sw = new StreamWriter(f);
sw.Write(textBox1.Text);
sw.Flush();
sw.Close(); f.Close();}
```

Misol	Matnli faylni o`qish uchun ochish, undan matnni TextBox oynasiga o`tkazish. Faylni ochishdan avval uni mavjudligini tekshirish, mavjud bo`lmasa, xabar berish

Dastur matni parchasi

```
private void button1_Click(object sender, EventArgs e){
try{
FileStream f = File.Open("primer.txt", FileMode.Open,
FileAccess.Read);
StreamReader sr = new StreamReader(f);
textBox1.Text = sr.ReadToEnd();
sr.Close();
f.Close();
}catch (IOException ex){
Console.WriteLine(ex.Message);
Console.WriteLine("Fayl nomi to`g`riligini tekshiring!");
}}
```

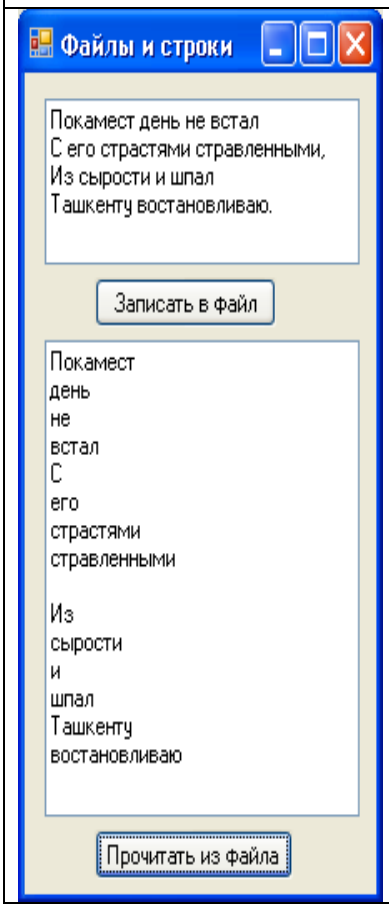
Misol.	Qo`shimcha uchun matnli fayl ochish, unga TextBox oynasidanmatn qo`shish.

Dastur matni parchasi

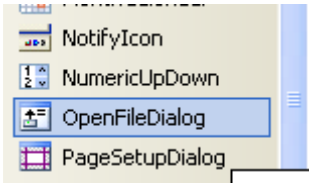
```
private void button1_Click(object sender, EventArgs e){
try{FileStream f=File.Open("primer.txt",FileMode.Append,
FileAccess.Write);
StreamWriter sw = new StreamWriter(f);
sw.WriteLine(textBox1.Text);
sw.Flush(); sw.Close();
f.Close();
}catch (IOException ex){
Console.WriteLine(ex.Message);
Console.WriteLine("Fayl nomi to`g`riligini tekshiring!");}}
```

Misol.	She`riy matn (satrda 80 simvollardan ortiq emas) «zinapoya» shaklida yozish (satrda bir so`zdan).

Loyiha shakli oynasi	Dastur matni parchasi

	using System; using System.Windows.Forms;
	using System.IO;
	namespace WindowsFormsApplication1{
	public partial class Form1 : Form{
	public Form1(){InitializeComponent();}
	private void button1_Click(object sender, EventArgs e){
	string s = textBox1.Text;
	char ch = new char[]{' ','.',',','!',';','(',')','-'};
	string[] ss = s.Split(ch); int k = ss.Length;
	FileStream f=File.Open("primer.txt",FileMode.Create,
	FileAccess.Write);
	StreamWriter sw = new StreamWriter(f);
	for (int i = 0; i < k; i++){
	sw.WriteLine(ss[i]);}
	sw.Flush(); sw.Close(); f.Close();}
	private void button2_Click(object sender, EventArgs e){
	try{FileStream f=File.Open("primer.txt",FileMode.Open,
	FileAccess.Read);
	StreamReader sr = new StreamReader(f);
	textBox2.Text = sr.ReadToEnd();
	sr.Close(); f.Close();}catch (IOException ex){
	Console.WriteLine(ex.Message);
	Console.WriteLine("Fayl nomi to`g`riligini tekshiring !");
	}}}}

OpenFileDialog Komponenti ochish faylini standart dialog oynasi Windows dan tanlab olishga imkon beradi.



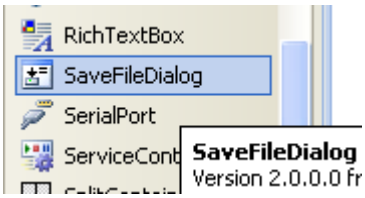
OpenFileDialog komponent xususiyatlari

Filter	Filtrlari muharriri oynasida ochiladigan fayllar nomi shablonini ko`rsatish imkonini beradi. Bir vaqtni o`zida bir nechta filtrni aniqlash (topish) mumkin.
FileName	Bu xususiyatda tanlangan fayl nomi saqlanadi.
Title	Panel sarlavhasi matnini beradi.

Dasturda foydalanishga misol

```
if (openFileDialog1.ShowDialog() == DialogResult.OK){
fileName = saveFileDialog1.FileName;}
```

SaveFileDialog Komponenti



Standart dialog oynasi Windows dan saqlash faylni tanlashga imkon beradi. Tashqi ko`rinishi va ish tartibi bo`yicha komponent OpenFileDialog bilan deyarli bir xil. Faqat yozuvlar va ayrim xususiyatlar bilan farqlanadi.

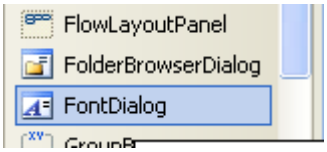
SaveFileDialog komponent xususiyatlari

Filter	Filtrlari muharriri oynasida saqlanadigan fayl nomi shablonini ko`rsatish imkonini beradi.
FileName	Bu xususiyatda saqlanadigan fayl nomi beriladi.
Title	Panel sarlavhasi matnini beradi.

Dasturda foydalanishga misol

```
if (saveFileDialog1.ShowDialog() == DialogResult.OK){
fileName = saveFileDialog1.FileName;
}
```

FontDialog Komponenti



Shrift va ularning tavsifini tanlash standart dialog panelini chaqirishga imkon beradi.

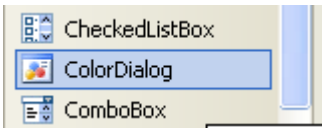
FontDialog komponent xususiyatlari

Font	Keyingi ish uchun qanday shrift tanlab olinganligini belgilaydi.

Dasturda foydalanishga misol

```
if (fontDialog1.ShowDialog() == DialogResult.OK){
textBox1.Font = fontDialog1.Font;
}
```

ColorDialog Komponenti



Ranglarni sozlash standart dialog panelini chaqirish imkonini beradi.

ColorDialog komponenti xossalari

Color	Tanlangan rangni saqlab qoladi

Dasturda foydalanishga misol

```
if (colorDialog1.ShowDialog() == DialogResult.OK){
BackColor = colorDialog1.Color;}
```

Misol	Fayldan ikki o`lchamli matritsani kiritish. Kerakli faylni ochish uchun OpenFileDialog komponentini qo`llash.

Dastur matni parchasi

```
private void button1_Click(object sender, EventArgs e){
dataGridView1.ColumnHeadersVisible = false;
dataGridView1.RowHeadersVisible = false;
dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
if(openFileDialog1.ShowDialog()==DialogResult.OK){
try{StreamReader sr = new StreamReader("primer.txt");
string[] s= sr.ReadLine().Split(' ');
int n = int.Parse(s[0]); int m = int.Parse(s[1]);
int[,] A = new int[n, m];
dataGridView1.RowCount = n;
dataGridView1.ColumnCount = m;
for (int i = 0; i < n; i++){
s = sr.ReadLine().Split(' ');
for (int j = 0; j < m; j++){
A[i, j] = int.Parse(s[j]);
dataGridView1.Rows[i].Cells[j].Value = A[i, j];}}
sr.Close(); }catch (IOException ex){
MessageBox.Show(ex.Message);
MessageBox.Show("Fayl nomini tug`ri kiriting!");
}}}
```

8-laboratoriya ishi yuzasidan nazorat savollari

1. Fayl o`zgaruvchisini qanday e`lon qilish kerak?
2. Fayl o`zgaruvchisini fizik fayl bilan qanday bog`lash mumkin?
3. Komponentlar palitrasidagi Dialogs betining OpenDialog, SaveDialog, FontDialog komponentlari vazifasini tushuntiring. Ularni izoh shaklida joylanishining o`ziga xosliklari nimadan iborat?
4. OpenDialog1.Filter xususiyatining vazifasi nimadan iborat? Qanday qilib bu xususiyat qiymatlarini o`zgartirish mumkin?
5. Dialog panelini chaqirishda qanday usul qo`llaniladi?
6. Fayl nomi qaysi xususiyatda joylashadi?
7. Matnli, tiplashtirilgan va tiplashtirilmagan fayllar orasidagi farqni ta`riflab bering.

Foydalanilgan adabiyotlar

1. Биллиг В. А. Основы программирования на C#. - М.: Изд-во «Интернет - университет информационных технологий - ИНТУИТ.ру», 2006. - 488 с.
2. Гуннерсон Э. Введение в C#. Библиотека программиста.-СПб.: Питер, 2001. -304 с.
3. Павловская Т. А. C/C++. Программирование на языке высокого уровня: Учебник для вузов. - СПб.: Питер, 2001. - 464 с.
4. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: Учебник для вузов. - СПб.: Питер, 2003. - 393 с.
5. Петцолъд Ч. Программирование для MS Windows на C#. Т. 2. - М.: Издательско-торговый дом «Русская Редакция», 2002. - 624 с.
6. Робисон У. C# без лишних слов. - М.: ДМК Пресс, 2002. - 352 с.
7. Секунов Н. Самоучитель C#. Серия "Самоучитель". - СПб.: БХВ-Петербург, 2001. - 576 с.
8. Фролов А. В., Фролов Г. В. Язык C#: Самоучитель. - М.: Диалог-МИФИ, 2003. - 560 с.
9. Шилдт Г. C#: Учебный курс. - СПб.: Питер, 2002. - 512 с: ил.
10. Шилдт Г. Полный справочник по C#. - М.: Издательский дом «Вильяме», 2004. - 752 с.

Mundarija

Rasulmuxamedov Mahamadaziz Mahamadaminovich
Boltayev Avaz Xudoyberdievich

Dasturlash tillari va texnologiyalari (C#, Python)

(2-qism)

Uslubiy qo’llanma

Muharrir:
Texnik muharrir va sahifalovchi:
Nashrga ruxsat etildi 20____ y.
Qog`oz bichimi 60×84/16. Hajmi __ b.t.
Adadi __ nusxa. Buyurtma №
ToshTYMI bosmaxonasida chop etildi
Toshkent sh., Odilxo`jaev ko`chasi,1

Toshkent temir yo`l muhandislari instituti, 20___y.