

# C++ proqramlaşdırma dili

Əhməd Sadıxov



Buraxılış 1.1

19.12.2011

Önsöz

Hörmətli oxucular, müəllimlər, tələbələr ümumiyyətlə IT dünyası, proqramlaşdırma ilə maraqlanan hər kəs. C++ proqramlaşdırma dili kitabının 1.1 buraxılışında hamınızı salamlayıram. Bu buraxılış ilkin buraxılışdan o qədər də çox fərqlənmir, sadəcə bəzi sintaktik səhvlər aradan qaldırılıb, bəzi başlıqlara nümunə proqramlar əlavə olunub. Kitab sərbəst örgənmək üçün nəzərdə tutulub. Hər bir başlıqda və bu başlığa uyğun həllər bölməsində kifayət qədər nümunə proqram kodları nəticə ilə birlikdə verilib. C++ dilini örgənməkdə hamınıza uğurlar arzulayıram.

Əhməd Sadıxov

Müəllif hüquqları:

Kitabda daxil olunan materialın və proqram nümunələrinin sizin hansısa işinizə yarayacağına müəllif tərəfindən heç bir təminat verilmir. Bu proqramlardan istifadə nəticəsində yaranan istənilən ziyana görə məsuliyyəti oxucu özü daşıyır, müəllif heç bir məsuliyyət daşımır. Siz bu kitabda daxil olunan material, proqram nümunələri və şəkilləri çap etmək, başqa şəxsə ötürmək, öz saytınızda yerləşdirmək kimi hüquqlara sahibsiniz. Kitabda verilən məlumatlardan istifadə üçün müəllifə istinad vermək zəruri deyil. Əgər sizin C++ proqramlaşdırma təcrübəniz varsa və kitabın gələcək versiyalarının hazırlanmasında köməklik göstərmək istəyirsinizsə müəlliflə aşağıdakı ünvandan əlaqə saxlaya bilərsiniz.

Bu ünvandan ancaq zəruri hallarda istifadə etməyiniz rica olunur.

ahmed.sadikhov@gmail.com

## Mündəricat

1 Giriş.....	4
2 Dəyişənlər.....	9
3 Operatorlar.....	27
4 Ünvan dəyişənləri.....	37
5 Funksiyalar.....	44
6 Cərgələr.....	53
7 Sətirlər.....	62
8 Strukt tiplər.....	68
9 Siyahılar.....	73
10 Siniflər.....	101
11 Makroslar və başlıq fayllar .....	111
Əlavələr.....	115
Bəzi çalışmaların həlləri.....	117
Qeydlər.....	191

## \$1 Giriş.

Bu mətndə C++ dilində proqram tərtibindən bəhs olunur. Bu mətndən istifadə edə bilmək üçün ilkin olaraq heç bir proqramlaşdırma dilini bilmək tələb olunmur.

Hər bir paraqrafın sonunda verilmiş çalışmaları mütləq yerinə yetirilməlidir. Bu sizə materialı daha da aydın başa düşməyə kömək etməklə yanaşı, sizdə gələcək inkişaf üçün əvəzəlməz olan proqramlaşdırma təcrübəsi yaradacaq və artıracaq.

Yadda saxlayın ki, proqramlaşdırmanı örgənmənin yeganə yolu ancaq və ancaq sərbəst proqram yazmaqdır.

Çətinliyi artırılmış məsələlər \* simvolu ilə qeyd edilir.

Gələcəkdə sistem proqramlaşdırmanı örgənmək istəyənlər \$9 – Siyahılar bölməsinə xüsusi ilə diqqət yetirməlidirlər.

### 1.1 Kompüterin işlək vəziyyətə gətirilməsi.

Mətndə daxil olunan bütün proqramlar **Windows** sistemlərində test edilib, lakin bütün **Unix** tipli sistemlərdə , o cümlədən **GNU/Linux** sistemlərində də **g++** və digər kompilyatorlar vasitəsilə asanlıqla kodda heç bir dəyişiklik etmədən kompilyasiya və icra oluna bilər.

### 1.2 Kompilyatorun installasiyası

Kompilyator olaraq **Ms Visual Studio** - dan istifadə edilir.

**Ms Visual Studio** kompilyatorunun quraşdırma faylını

<http://www.microsoft.com/express/Downloads> keçidindən endirə bilərsiniz.

### 1.3 İlk test proqram

Sistem və kompilyator problemlərini həll etdikdən sonra hər şeyin qaydasında olduğunu yoxlamaq üçün test proqramı yerinə yetirək.

**Ms Visual Studio** proqramını yükləyin.

Daha sonra

**File -> New -> Project**

Seçimini edirik.

New Project pəncərəsi açılacaq.

**Project types:** panelindən **Win32** , **Templates:** panelindən isə **Win32 Console Application** seçimini edirik. Daha sonra **Name:** pəncərəsindən yeni yaradacağımız proqramın adını daxil edirik. Bu pəncərəyə **prog1** yazıb **Ok** düyməsini basırıq. Açılan yeni pəncərədə **Finish** düyməsini basırıq. Bu əməliyyatlar bizə ilk proqramımızı tərtib etmək üçün bütün lazımı faylları yaradacaq və proqramın mətn faylı redaktoru pəncərəsi - **prog1.cpp** aktivləşdirəcək. Bu fayl təqribən aşağıdakına bənzər formada olur.

```
// prg_1_1.cpp : Defines the entry point for the console
//application.
```

```
#include "stdafx.h"
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}
```

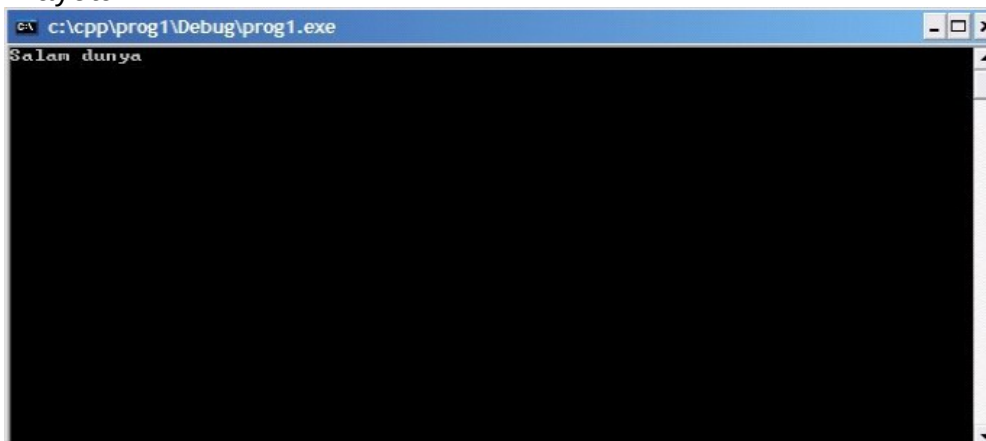
Bu mətn kompilyator tərəfindən avtomatik yaradılıb. Proqramın mətn faylında aşağıdakı kimi dəyişikliklər edirik. `#include "stdafx.h"` sətirindən sonra `#include <iostream>` ,

```
{  
mötərəzəsindən sonra isə  
std::cout<<"Salam dunya \n"; ,  
return 0; sətirindən əvvəl isə  
int x;  
std::cin>>x;  
sətrlərini daxil edirik.  
Aşağıdakı kimi:
```

```
// prg_1_2.cpp : Defines the entry point for the console  
//application.  
  
#include "stdafx.h"  
#include <iostream>  
  
int _tmain(int argc, _TCHAR* argv[])  
{  
std::cout<<"Salam dunya \n";  
int;  
std::cin>>x;  
return 0;  
}
```

Etdiyimiz dəyişiklikləri yadda saxlamaq üçün **File -> Save all** düyməsini basırıq. Artıq proqramımızın mətn faylı hazırdır və biz onu kompilyasiya edə bilərik. Kompilyasiya nəticəsində kompilyator bizim mətn faylından prosessor tərəfindən icraolunabilən ikili proqram alacaq.

Proqramı kompilyasiya etmək üçün **Build -> Build Solution** əmrini daxil edirik. Proqramımızın kompilyasiyası başlayacaq və nəticə **Output** pəncərəsinə ötürüləcək. Əgər sonda **Build: 1 Succeeded ...** sətiri çap olunursa deməli proqramımız uğurla kompilyasiya olunmuşdur. İndi isə proqramımızı yerinə yetirək. Bunun üçün **Debug -> Start Debugging** düyməsini basırıq. Nəticədə kansol pəncərəsi açılacaq və "Salam dunya" mətni çap olunacaq. Proqramı söndürmək üçün klaviaturadan hər-hansı simvol daxil edib enter düyməsinə basmağımız kifayətdir.



Əgər bu nəticəni almırsınızsa deməli bütün hazırlıq işləri tamamdır, siz növbəti paragraflarda verilən bütün proqram nümunələrini də eyni qayda ilə yerinə yetirə bilərsiniz.

Əgər bu nəticəni almırsınızsa onda üzülməyin, düzün desək hətta professional proqramçılar da hansısa yeni dili örgənəndə ilk proqramı heç də həmişə uğurla yerinə yetirmirlər. Ancaq bütün çətinlik elə bu ilk addımdadır.

### **Proqramın izahı:**

Proqram icra olunduqda `std::cout<<"Salam dünya \n";` sətirində verilən **Salam dünya** ifadəsini ekranda çap edir. Əgər **Salam dünya** əvəzinə istənilən digər ifadə yazsanız onda ekranda həmin ifadə çap olunur.

Proqramın yerdə qalan detalları barədə irəlidə müvafiq bölmələrlə tanış olduqdan sonra məlumat veriləcək. Hələlik isə onu deyə bilərik ki, `#include<iostream>` sətiri istifadə edəcəyimiz bütün proqramlarda daxil olunur. Proqrama bu sətiri daxil etmək bizə `std::cout`, `std::cin` funksiyalarından istifadə etməyə imkan verir.

Növbəti sətir `int _tmain(int argc, _TCHAR* argv[])` sətiridir.

Bu proqramın əsas funksiyasıdır. C++ dilində yazılmış proqramlarda müxtəlif işlər görəndə standart(`std::cout`) və proqramçı tərəfindən yaradılan funksiyalardan (`$5`) istifadə olunur. `main` funksiyası (`_tmain` unikodu dəstəkləmək üçün MS Vstudio-nun əlavəsidir) isə xüsusi funksiyadır. Bütün C++ proqramlarında bu funksiya mütləq olmalıdır və bütün C++ proqramları icra olunmağa `main` funksiyasından başlayır.

Beləliklə sizə C++ dilində yazılmış istənilən proqramın mətn kodu verilərsə onda siz `main` funksiyasını tapmaqla proqramın icra olunmağa başladığı yeri asanlıqla müəyyənləşdirə bilərsiniz.

Ən sonda yerləşən `}` simvolu isə `main` funksiyasının bitdiyini göstərir.

### **Çalışmalar:**

1. `prog1.cpp` proqramında `std::cout<<"Salam dünya \n";` sətirindən sonra `std::cout<<"Hey bu yeni proqramdır \n";` sətirini yerləşdirin, faylı yadda saxlayın, kompilyasiya və icra edin.
2. C++ dilində "Mən C++ dilini örgənirəm" sətirini çap edən proqram tərtib edin.
3. (\*) C++ dilində Səməd Vurğunun Azərbaycan şerinin ilk bəndini ekranda çap edən proqram tərtib edin.

## \$2 Dəyişənlər.

### 2.1 Dəyişənlərin tipləri.

Əvvəlki paraqrafda biz C++ dilində necə proqram yerinə yetirməyi örgəndik. Bu paraqrafda biz C++ dilində yazılmış proqramın ən vacib elementlərindən biri – dəyişənlərlə tanış olacağıq. Hər bir proqram yerinə yetirilərkən müxtəlif məlumatları yadda saxlamalı (yaddaşa yerləşdirməli) olur. Tutaq ki, iki ədədin cəmini hesablayan proqram yazmaq istəyirik. Bu zaman biz yaddaşa 3 məlumat üçün yer ayırmalıyıq. İki toplanan və cəm.

Proqramlaşdırmada hər hansı məlumatı qəbul etmək, yadda saxlamaq və bu məlumatın qiymətinə müraciət etmək üçün dəyişənlərdən istifadə olunur. Konkret olaraq dəyişən adı olan müəyyən bir yaddaş sahəsidir. Bu yaddaşın sahəsinin həcmi və adını dəyişəni elan edərkən biz özümüz (proqramçılar) müəyyən edirik.

Proqramda dəyişən elan etmək üçün aşağıdakı sintaksisdən istifadə olunur:

**tip ad;**

Burada **tip** dəyişənin tipi, **ad** isə dəyişənə verdiyimiz addır.

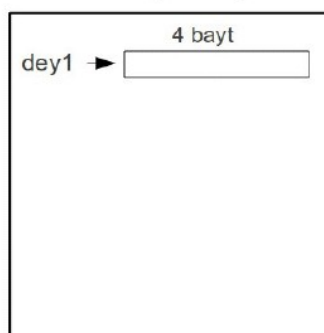
Misal üçün aşağıdakı kimi:

**int dey1;**

Yuxarıdakı kod hissəsində biz **int** tipindən olan **dey1** adlı dəyişən elan etdik. Fiziki yaddaşa (RAM) 4 bayt yer ayrılır və həmin yerə **dey1** adı ilə müraciət olunur. Artıq bundan sonra proqramda **dey1** üzərində apardığımız bütün əməliyyatlar birbaşa **dey1** - ə aid yaddaş sahəsi üzərində aparılacaq (şəkil1).

**int dey1;**

Fiziki yaddaş



şəkil 1

Yadda saxladığı məlumatın növünə və yaddaşa tutduğu yerin həcminə görə dəyişənlər tiplərə ayrılır. Misal üçün tam ədədlər tipi – **int** (4 bayt), kəsr ədədlər tipi – **double** (8 bayt) , simvol tipi – **char** (1 bayt), sətir tipi – **char []**, **char \* v.s.**

Bu tiplərə standart tiplər deyilir.

Bundan əlavə C++ dilində ünvan dəyişənləri tipi, struktur tiplər, siyahılar və siniflərdən də çox geniş istifadə olunur ki, bunlarla da uyğun olaraq 4, 8, 9, 10-cu paraqraflarda məşğul olacağıq.

Dəyişənlərə istədiyimiz kimi ad verə bilərik yalnız və yalnız hərflərdən (ingilis əlifbasının) , '\_' simvolundan və rəqəmlərdən istifadə etməklə. Dəyişənin adı mütləq hərflə başlamalıdır və operator, tip v.s. adlarından da dəyişən adı kimi istifadə etmək olmaz. Operatorlarla gələn mövzularda tanış olacağıq.

Beləliklə cəm proqramında hər iki toplananı və onların cəmini yerləşdirmək üçün biz tam tipli 3 dəyişən təyin etməliyik. Gəlin bu dəyişənləri uyğun olaraq **top1**, **top2** və **cem** kimi adlandıraraq.

Bu dəyişənləri təyin etmək üçün proqram kodu aşağıdakı kimi olacaq.

```
int top1;  
int top2;  
int cem;
```

Qeyd edək ki, eyni tiptən olan dəyişənləri vergüllə ayırmaqla bir sətirdə də elan edə bilərik.

Aşağıdakı kimi :

```
int top1, top2, cem;
```

Burada ; simvoluna diqqət yetirməyinizi istəyirəm. Bu işarə kompilyatora hər hansısa bir əməliyyatın (indiki halda dəyişənlərin elanının) bitməsini göstərir.

Gəlin proqramımızı tərtib edək. Hələlik 3 dəyişən elan etmişik(dəyişən elan etmək və ya təyin etmək eyni mənə bildirir).

**Proqramımız belə olacaq:**

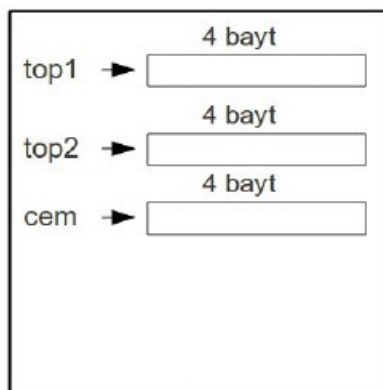
```
// prg_2_1.cpp  
#include "stdafx.h"  
#include <iostream>  
int _tmain(int argc, _TCHAR* argv[])  
{  
int top1, top2, cem;  
top1 = 4;  
top2 = 6;  
cem = top1 + top2;  
std::cout<<" 4 ile 6 -nin cemi ="<<x;  
int x;  
std::cin>>x;  
return 0;  
}
```

Proqramı yadda saxlayaq , kompilyasiya edək və yerinə yetirək.

**Proqramın izahı:**

Program icra olunmağa `int _tmain(int argc, _TCHAR* argv[])` sətirindən başlayır. Daha sonra proqramda `int top1, top2, cem;` sətiri gəlir. Burada biz `int` tipli `top1, top2` və `cem` dəyişənlərini elan edirik. Yaddaşın vəziyyəti bu zaman aşağıdakı kimi olar(şəkil 2):

Fiziki yaddaş



şəkil 2

Növbəti sətir aşağıdakı kimidir:

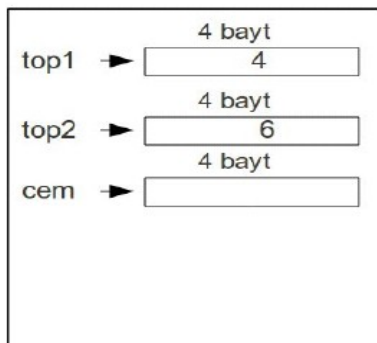
```
top1 = 4;
```

Burada biz mənimsətmə operatorundan ('=') istifadə edirik. Mənimsətmə operatoru - '=' riyaziyyatdan yaxşı bildiyimiz bərabərlik simvolu kimi işarə olunur, amma proqramlaşdırmada, daha doğrusu C++ dilində o ayrı funksiya daşıyır. Mənimsətmə operatoru ilə biz operatorun

sol tərəfində göstərilən yaddaş sahəsinə sağ tərəfdə verilən məlumatı yerləşdiririk. Yuxarıdakı kod sətirində mənimsətmə operatorundan solda **top1** dəyişəni, sağda isə 4 qiyməti durur. Beləliklə bu kod icra olunduqda yaddaşın **top1** - ə aid hissəsinə 4 qiyməti yazılır. Diqqət yetirilməli mühüm məqamlardan biri də odur ki, bu zaman **top1** -də əvvəl nə məlumat varsa o silinir. Uyğun olaraq **top2 = 6**; proqram sətiri də yaddaşın **top2** - yə aid olan hissəsinə 6 qiymətini yazır.

Bu zaman yaddaşın vəziyyəti aşağıdakı kimi olar:

Fiziki yaddaş



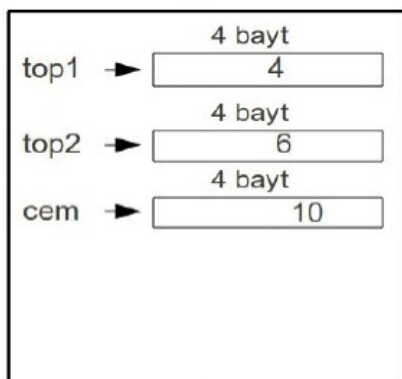
şəkil 3

Proqramın növbəti sətirinə keçək:

**cem = top1 + top2;**

Burada biz yenə də = mənimsətmə operatorundan istifadə edərək. Bu zaman əvvəl operatorun sağ tərəfində dayanan riyazi ifadənin qiyməti hesablanır **top1 + top2**. CPU **top1** və **top2** -nin yaddaş sahəsinə yazılmış qiymətləri (4,6) oxuyur və onların cəmini hesablayır (10) və yekun nəticəni **cem** dəyişəninə aid yaddaş sahəsinə yazır:

Fiziki yaddaş



şəkil 4

Proqramın növbəti sətiri belədir:

**std::cout<<" 4 ile 6 -nin cemi ="<<cem;**

Bu proqram kodu icra olunduqda ekranda **4 ile 6 -nin cemi = 10** ifadəsi çap olunur. Burada **std::cout** çap funksiyasından istifadə olunur. Funksiyalar barədə daha ətraflı \$5 -də danışacağıq. **std::cout** funksiyasının geniş izahı **Əlavə A** -da verilib.

**Proqram nümunələri:**

1. 3 ədədin cəminə hesablayan proqram tərtib edin.

**Həlli:**

Bu proqramda bizə 3 ədədi və onların cəmini yadda saxlamaq üçün **int** tipli 4 dəyişən lazım olacaq. Bu dəyişənlərin elanı aşağıdakı kimi olacaq:

**int x,y,z,h;**



Daha sonra  $x, y, z$  - ə müxtəlif qiymətlər mənimsədək:  $x=8; y=45; z=3;$   
İndi isə bu 3 dəyişənin cəmini  $h$  dəyişəninə mənimsədək.  $h = x + y + z;$   
Nəticəni çapa verək: `std::cout<<x<<" , "<<y<<" , "<<" , "<<z<<"ededlərinin cemi ="<<h;`  
Program aşağıdakı kimi olar:

```
// prg_2_2.cpp
```

```
#include "stdafx.h"
#include <iostream>

int main()
{
    int x,y,z,h;
    x=8;
    y=45;
    z=3;
    h = x + y + z;
    std::cout<<x<<" , "<<y<<" , "<<" , "<<z<<"ededlərinin cemi ="<<h;
    int x;
    std::cin>>x;
    return 0;
}
```

## 2.2 Dəyişənlərin elanı qaydaları.

C++ dilində dəyişənlər programın istənilən yerində elan oluna bilər. Dəyişənlərə müraciət isə yalnız dəyişənlərin elanından aşağı hissədə edilə bilər. Eyni tiptən olan bir neçə dəyişən ayrı-ayrı sətirlərdə və ya eyni elan sətirində elan oluna bilər.

Aşağıdakı kimi:

```
int x,y,z;
və ya
int x;
int y;
int z;
```

Dəyişənlərə birbaşa elan sətirində qiymət mənimsədə bilərik, aşağıdakı kimi:

```
int x, y=20, z=0;
```

Yuxarıdakı elnada biz  $y$  və  $z$  dəyişənlərini elan edirik və bu dəyişənlərə başlanğıc qiymətlər mənimsədirik. Bundan əlavə C++ dilində aşağıdakı kimi də, mənimsətmə qaydalarından istifadə olunur. **deyishen += qiymet;**

Bu yazılış aşağıdakına ekvivalentdir .

```
deyishen = deyishen + qiymet;
```

Buradan görürük ki, mənimsətmə operatorunun sağ tərəfindəki ifadə onun öz qiymətindən istifadə olunur. Programlaşdırma nöqtəyi nəzərdən burada hər şey qaydasındadır, belə ki, bu zaman dəyişənin əvvəlki qiyməti sağ tərəfin qiymətinin hesablanmasında istifadə olunacaq və yekun qiymət dəyişənə mənimsədiləcək.

Misal üçün:

$x += 5;$  əməliyyatı  $x$  -in qiymətin 5 vahid artırır.

və ya

$x += y;$  əməliyyatı  $x$  -in qiyməti üzərinə  $y$  əlavə edir.

Eyni qayda vurma və çıxma əməliyyatlarına da aiddir.

$x *= 5;$  əməliyyatı  $x$  -in qiymətin 5 dəfə artırır.

## 2.3 İnkrement və Dekrement.

C++ dilində **İnkrement** və **Dekrement** adlandırılan xüsusi operatorlar var ki, onlar dəyişənlərin qiymətin müvafiq olaraq 1 vahid artırmaq və azaltmaq üçün istifadə olunur. Bunlar uyğun olaraq aşağıdakılardır:

**İnkrement** - artırma ++, **Dekrement** azaltma -- .

Misal üçün inkrementdən istifadə edərək x -in qiymətin 1 vahid artırmaq istəsək aşağıdakı kimi yazı bilərik.

```
x++; və ya ++x;
```

Eyni qayda ilə dekrement x -in qiymətin 1 vahid azaldır, aşağıdakı kimi:

```
x--; və ya --x;
```

Bəs toplama, çıxma işarələrinin dəyişəndən əvvəl və ya sonra olmasının fərqi varmı? Əgər bu operatorun məqsədi sadəcə qiyməti dəyişməkdirsə onda işarəni sağda və ya solda yazmağın fərqi yoxdur. Lakin əgər inkrement və ya dekrement hansısa ifadənin daxilindədirsə onda sağ,solun fərqi var. Belə ki, işrə solda olanda ifadədə dəyişənin ilkin qiyməti, sağda olanda isə 1 vahid dəyişdirilmiş yeni qiyməti hesablanır.

İndi isə biraz daha maraqlı, interaktiv proqramlar tərtibi ilə məşğul olaq. Baxdığımız proqramlarda biz hesab əməlləri üçün əvvəlcədən təyin olunmuş qiymətlərdən istifadə etdik, indi isə elə proqramlar yazaq ki, dəyişənlərin qiymətlərinin istifadəçi tərəfindən daxil edilməsi mümkün olsun. C++ dilində istifadəçinin daxil etdiyi məlumatı proqramdakı dəyişənlərə mənimsətmək üçün əsasən **std::cin** funksiyasından istifadə olunur.

**std::cin** funksiyası barədə **Əlavə A** -da geniş izah verilir.

Tutaq ki, biz proqramda **int** tipli x dəyişəni elan etmişik.

```
int x;
```

Əgər biz istəyiriksə x - dəyişənin yaddaş sahəsinə (qısa olaraq x dəyişəninə deyəcəyik) istifadəçi tərəfindən daxil olunan qiymət yazaq, onda proqrama aşağıdakı sətiri yerləşdirməliyik.

```
std::cin>>x;
```

Bütün bu dediklərimizi proqram nümunəsində test edək. Elə bir sadə proqram tərtib edək ki, istifadəçidən hər hansı ədəd daxil etməsini istəsin, daha sonra isə bu ədədin kvadratın ekranda çap eləsin.

Qısa bir qeyd: Gəlin yaratdığımız proqramları icra etmənin yeni metodu ilə tanış olaq. Əvvəlcə C diskində yeni bir qovluq yaradırıq. Bu yeni yaratdığımız qovluq **cpp** adlandıraraq. Daha sonra ilk test proqramında olduğu kimi yeni bir proyekt yaradırıq. **New Project** pəncərəsinin **Name** alt pəncərəsinə yeni proqramımızın adını daxil edək, **prog2**. **Location** pəncərəsinə isə ünvan olaraq yeni yaratdığımız qovluğun ünvanını daxil edək, **C:\cpp**. Proqramımızın mətnində aşağıdakı kimi dəyişiklik edək.

```
// prog2.cpp : Defines the entry point for the console application.
```

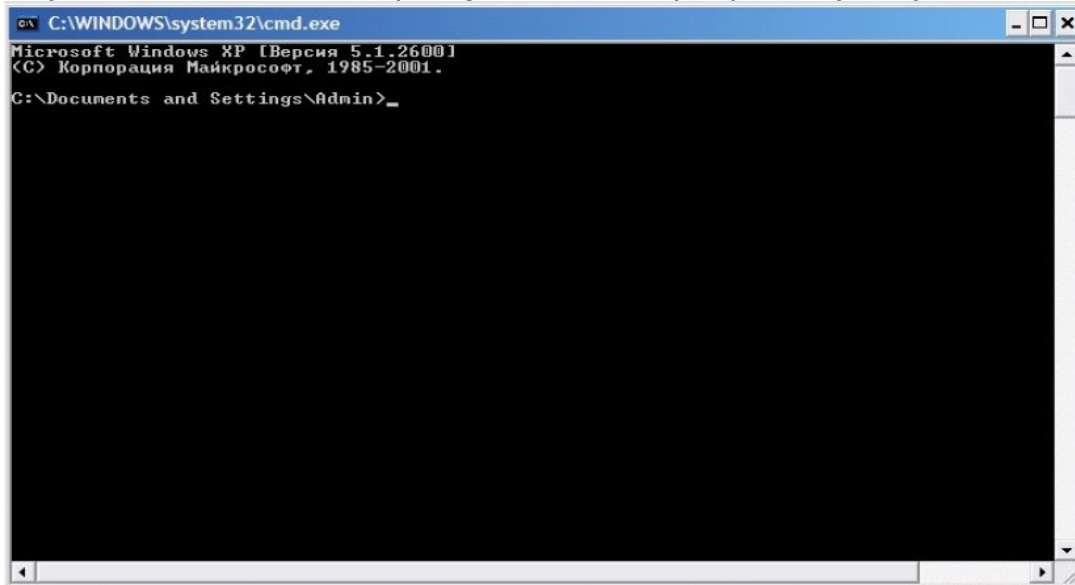
```
#include "stdafx.h"
#include <iostream>
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    std::cout<<"Salam dunya \n";
    return 0;
}
```

Yadda saxlayıb, kompilyasiya edirik. Əgər proqramımızı əvvəlki qayda ilə icra eləsək, (Debug -> Start Debugging) onda kansol pəncərəsi bir anlığa açılıb bağlanacaq. Kansol (cmd) proqramını özümüz yükləyək.

**Start -> Run -> CMD**

və ya C:\Windows\system32 qovluğunda cmd.exe proqramını yükləyirik.



Daha sonra kansoldan aşağıdakı əmrləri daxil edirik:

```
cd C:\cpp\prog2\Debug  
prog2.exe
```

Nəticədə proqramımız icra olunacaq.

```
C:\Documents and Settings\Admin>
```

```
C:\Documents and Settings\Admin>
```

```
C:\Documents and Settings\Admin>cd C:\cpp\prog2\debug
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
Salam Dunya
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

Bundan sonra bütün proqramlarımızı bu qayda ilə icra edəcəyik. İndi isə yuxarıda daxil etdiyimiz ədədin kvadratı proqramı ilə məşğul olaq. Əvvəl proqramı daxil edək daha sonra isə izahı ilə tanış olarıq.

Proqramın mətn kodu aşağıdakı kimi olacaq:

```
// prg_2_3.cpp
```

```
#include "stdafx.h"
```

```
#include <iostream>
```

```
int main()
```

```
{
```

```
int x,y;
```

```
std::cout<<"Zehmet olmasa her hansı eded daxil edin \n";
```

```
std::cin>>x;
```

```
y = x*x;
```

```
std::cout<<x<<" in kvadrati = "<<y<<"\n"
```

```
return 0;
```

```
}
```

Proqramı yerinə yetirək, unutmayaq ki, proqramın mətn faylında hər - hansı dəyişiklik etdikdən sonra bu dəyişikliklərin ikili faylda (**prog2.exe**) oturması üçün mütləq kompilyasiya etməliyik .

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Zehmet olmasa her hansı eded daxil edin
67
67 in kvadrati = 4489
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### Proqramın izahı:

Proqramda int tipli **x** və **y** dəyişənləri elan edirik. Daha sonra **std::cin** funksiyası ilə istifadəçinin daxil etdiyi qiyməti **x** dəyişəninə mənimsədirik (**x** -in yaddaş sahəsinə yazırıq). **y** -ə **x** -in kvadratını mənimsədirik və çap edirik.

Başqa proqrama baxaq:

Elə proqram tərtib edin ki, istifadəçidən düzbucaqlının enini və uzunluğunu daxil etməsini istəsin. Daha sonra proqram düzbucaqlının sahəsini ekranda çap etsin.

Əvvəlcə proqramı sərbəst yazmağa cəhd edin.

Proqram aşağıdakı kimi olacaq:

```
// prg_2_4.cpp

#include <iostream>
int main(){
int en, uz, sahe;
std::cout<<"Zehmet olmasa duzbucaqlnn enini daxil edin \n";
std::cin>>en;
std::cout<<"Zehmet olmasa duzbucaqlnn uzunlugunu daxil edin \n";
std::cin>>uz;
sahe = en*uz;
std::cout<<"Duzbucaqlinin sahəsi = "<<sahe<<" \n";
return 0;
}
```

Proqramı kompilyasiya edib yerinə yetirək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Zəhmət olmasa düzbucaqlının enini daxil edin
56
Zəhmət olmasa düzbucaqlının uzunluğunu daxil edin
23
Düzbucaqlının sahəsi = 1288
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## 2.4 Char tipi və ya Simvol tipi

Biz qeyd elədik ki, simvol tipi C++ dilində **char** kimi elan olunur. Gəlin simvol tipindən olan bir neçə dəyişən elan edək və onlara qiymət mənimsədək.

```
char x,y,z;
x = 'a'; y = 'B';
```

Yuxarıdakı proqram kodunda biz **x** dəyişəninə **a** qiymətini, **y** dəyişəninə **B** qiymətini mənimsətdik. Sadə proqram nümunəsinə baxaq:

Program nümunəsi.

Aşağıdakı programı kompilyasiya edib yerinə yetirin:

```
// prg_2_5.cpp
#include <iostream>

int main(){
char x,y;
x = 'a';
y = 'B';
std::cout<<"x ve y -in qiymetleri: "<<x<<" , "<<y<<"\n";
return 0;
}
```

Programı kompilyasiya edib yerinə yetirək

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
x ve y -in qiymetleri: a , B
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

C++ dilində bir prinsip mövcuddur: hər şey ədəddir. Yəni bizim simvol, sətir, kəsr v.s. kimi qəbul etdiyimiz hər şey aşağı səviyyədə ədədlərlə ifadə olunur. Sadəcə olaraq ədədlərə müxtəlif tipdən olan dəyişən kimi yanaşmaq bizə istifadə rahatlığı yaradır. Başqa sözlə char tipindən olan dəyişəni ədədə mənimsədə bilirik .

Yuxarıda baxdığımız programın bir qədər dəyişilmiş variantına baxaq:

```
//prg_2_6.cpp
#include <iostream>

int main()
{
char x,y,z;
int f,h;
x = 'a'; y = 'B';
f = x; h = y;
std::cout<<"x ve y -in qiymetleri simvol sheklinde: " <<x<<" , "<<y<<"\n";
std::cout<<"x ve y -in qiymetleri eded sheklinde: "<<f <<" , "<<h<<"\n";
z = 99;
std::cout<<"z -in qiymeti simvol sheklinde: "<<z<<"\n";
return 0;
}
```

Programı kompilyasiya edib yerinə yetirək

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
x ve y -in qiymetleri simvol sheklinde: a , B
x ve y -in qiymetleri eded sheklinde: 97 , 66
z -in qiymeti simvol sheklinde: c
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

C++ dilində hər hansı simvola müraciət etmək üçün bu simvolu tək dırnaq işarəsi ilə göstəririk- ' '. Misal üçün əgər programda 'x' , 'y' yazırıqsa kompiyator bunu simvo kimi qəbul edəcək, əks halda(dırnaqsız) isə dəyişən kimi. Günümüzdə simvolların sayı həddən artıq çoxdur, İngilis dilinin simvolları, kiril əlifbasının simvolları, ərəb, çin, yapon və digər əlifbanın

10 000 -lə simvolu mövcuddur. Bu simvollar müxtəlif cür təsnif olunur(unicode v.s. ) və ədəd qarşılığı göstərilir. Proqramçının bilməli olduğu ən əsas simvollar cədvəli **ASCII** simvollar cədvəlidir. **ASCII** cədvəli 128 simvoldan ibarətdir. Onların siyahısı və ədəd qarşılığı Əlavə B -də göstərilib. Bu siyahıdan olan bəzi **ASCII** simvolları daha çox istifadə olunur.

Bunlar aşağıdakılardır:

'\n' yeni sətir simvolu.

'\t' tabulyasiya simvolu.

'\0' sətirin sonunu bildirmək üçün istifadə olunan somvol.

## 2.5 Proqramda Şərhlər

C++ dilində yazılmış hər – hansı proqramı şərhə təsvir etmək olmaz. Şərh (comment, statement ... ) proqramın bu və ya digər hissəsinin hansı iş gördüyünü bildirmək üçün proqramın mətn koduna əlavə olunur . Şərhlər ancaq proqramın işini başa düşmək istəyənlər üçündür. Proqramın real yerinə yetirilən koduna şərhlərin heç bir aidiyyəti yoxdur. Belə ki, kompilyator proqramı kompilyasiya edərkən birinci gördüyü iş şərhləri proqram kodundan silməkdir.

C++ dilində 2 cür şərhərdən istifadə olunur: çoxsətirli - /\* və \*/ və təksətirli - // . Çoxsətirli şərhərdən istifadə etdikdə kompilyator /\* və \*/ arasında qalan bütün proqram kodun şərh kimi qəbul edəcək . Təksətirli şərhərdən istifadə edən zaman kompilyator // simvollarından həmin sətirin sonuna kimi olan hissəni şərh kimi qəbul edəcək. Bu zaman kompilyator bu sətirləri nəzərə almayacaq. Onu da deyim ki, çox vaxt bu qaydadan proqramdakı səhvləri tapmada istifadə olunur (proqramın müəyyən hissəsini şərh kimi verib nəticəni yoxlamaqla). Misal üçün, tutaq ki, biz düzbucaqlının enini və uzunluğunu yadda saxlamaq üçün int tipli en və uz adlı dəyişənlər elan etmək istəyirik .

```
int en, uz;
```

Əgər biz bu dəyişənlərin hansı məqsəd üçün elan olunduqlarını proqramda şərh kimi daxil etmək istəyiriksə aşağıdakı kod hissəsini proqrama daxil edirik.

```
/* Burada en düzbucaqlının enini, uz isə uzunluğunu bildirir. */
```

Kompilyator proqram kodunda /\* ifadəsinə rast gəldikdə bu ifadə də daxil olmaqla \*/ -yə kimi hissəni şərh kimi qəbul edir. Şərhlər proqramçıya çox kömək edir və biz də şərhərdən tez-tez istifadə edəcəyik.

## 2.6 Dəyişənlər üzərində əməllər.

Biz dəyişənlər üzərində onların tipindən asılı olmayaraq aşağıdakı hesab və müqaisə əməllərini apara bilərik:

\*, -, +, /, %

Uyğun olaraq vurma, çıxma, toplama, bölmə və qalıq əməllərini bildirir.

### Dəyişənlər üzərində hesab əməlləri

Tutaq ki, bizə int tipindən olan x,y,z dəyişənləri verilib.

```
int x,y,z;
```

Onlara müxtəlif qiymətlər mənimsədək:

```
x = 786;
```

```
y = 93;
```

İndi tutaq ki, mən istəyirəm z -ə x -in y -ə nisbətini mənimsədim.Yəni x -i y -ə böləndə alınan tam ədəd.

Kod aşağıdakı kimi olar:

```
z = x/y ;
```

Qeyd edək ki, bu əməliyyatda istifadə olunan bütün dəyişənlər tam tipli ( `int` ) olduğundan, bölmənin qalıq hissəsi atılır. Əgər mən istəsəydim `x` -in `y` -ə nisbətinin qalıq hissəsini tapım onda qalıq - `'%'` operatorundan istifadə etməliyəm.

```
z = x%y ;
```

Bu zaman `z` tam tipli dəyişən olduğundan qalığın 0-dan kiçik hissəsi atılır. Əgər mən bölmə əməliyyatının nəticəsini heç bir ixtisarsız, tam şəkildə almaq istəyirəmsə onda `int` əvəzinə `double` və ya `float` tipli dəyişənlərdən istifadə etməliyəm.

```
double = q;
```

```
q = x/z;
```

### Program nümunəsi.

Aşağıdakı proqramı kompilyasiya edib yerinə yetirin:

```
//prg_2_7.cpp
```

```
#include <iostream>
```

```
int main()
{
int x,y,z,h;
double q,p,f;
// x,y -e bezi qiymetler menimsedek
x=238;
y=45;

// z -te x-in y-e nisbetinin tam hissesini menimsedek
z=x/y;
std::cout<<" 238 / 45 in tam hissesi = "<<z<<"\n";
// h -a x-in y-e nisbetinin qaliq hissesini menimsedek
h = x%y;
std::cout<<" 238 / 45 in qaliq hissesi = "<<h<<"\n";
// Indi ise q,p -ye bezi qiymetler menimsedek
q = 132.258;
p = 43.91;
// f-e q/p -ni menimsedek
f = q/p;
std::cout<<q<<" / "<<p<<" = "<<f<<"\n";
}
```

Proqramı kompilyasiya edib yerinə yetirək

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>./prog2.exe
```

```
238 / 45 in tam hissesi = 5
```

```
238 / 45 in qaliq hissesi = 13
```

```
132.258 / 43.91 = 3.01202
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

### Çalışmalar

1. Elə proqram yazın ki, istifadəçidən 5 ədəd daxil etməsini istəsin, daha sonra proqram bu ədədlərin cəmini ekranda çap etsin.
2. Elə proqram tərtib edin ki, istifadəçidən üçbucağın tərəflərinin uzunluğunu daxil etməsini istəsin və üçbucağın perimetrini çap etsin.
3. Elə proqram tərtib edin ki, istifadəçidən dairənin radiusunu daxil etməsini istəsin və dairənin sahəsini çap etsin. `pi` -nin qiymətini 3.14 götürün.

## \$3 Operatorlar.

Əvvəlki paragrafda biz proqramlaşdırmanın əsas elementlərindən biri - DƏYİŞƏNLƏR , onların tipləri və dəyişənlər üzərində hesab əməlləri ilə tanış olduq. Bu paragrafda isə biz C++ dilində yazılmış proqramların digər əsas elementləri - OPERATORLAR ilə tanış olacağıq. OPERATORLAR proqramlaşdırmada hər hansı şərtdən asılı olaraq proqramın növbəti icra istiqamətini müəyyənləşdirir. Məsələn üçün hər hansı şərtdən asılı olaraq bu və ya digər kod hissəsi icra olunur, hər hansı kod hissəsi bir neçə dəfə təkrar olunur və ya bir neçə kod hissəsindən biri icra olunur.

### 3.1 Şərt operatorları

Biz 3 qrup operatorlarla tanış olacağıq : **Şərt, Dövr, Seçim**. Şərt operatoru çox sadədir. Əvvəlki proqram nümunələrində biz müxtəlif proqram kodları icra etdik.

Məsələn üçün:

```
std::cout<<" x = "<<x<<"\n";
```

kod hissəsi ilə biz x -in qiymətini ekranda çap edirdik və bu zaman heç bir şərtdən istifadə etməyirdik. Şərt operatoru bizə bu imkanı verir, yəni biz istədiyimiz şərtdən asılı olaraq x -in qiymətini çap edərik və ya heç bir iş görmərik.

Tutaq ki, məsələn belədir: Əgər x -in qiyməti 234 -dən böyükdürsə onda onu çap et.

Kod aşağıdakı kimi olacaq:

```
if (x > 234)
```

```
std::cout<<" x = "<<x<<"\n";
```

Şərt operatorunun sintaksisi aşağıdakı kimidir:

```
if (şərt)
```

```
    { yerinə yetirilməli əməliyyatlar }
```

```
else
```

```
    { digər əməliyyatlar }
```

Qeyd edək ki, əgər cəmi bir əməliyyat yerinə yetirilirsə onda {} mötərizələrinə ehtiyac yoxdur.

**İzahı:**

Əvvəlcə şərt yoxlanılır , əgər doğrudursa onda {} arasında olan əməliyyatlar yerinə yetirilir əks halda else – dən sonrakı {} mötərizələri arasında olan əməliyyatlar yerinə yetirilir.

Digər misal:

```
int x;
```

```
if (x<5)
```

```
std::cout<<x<<" 5 -den kiçikdir "<<"\n";
```

```
else
```

```
std::cout<<x<<" 5 -den böyükdür "<<"\n";
```

Əgər verilmiş kod hissəsinin icra olunması üçün bir deyil bir neçə şərt ödənməlidirsə onda biz bu şərtlərin hamısını və (&&) operatoru ilə birləşdirə bilərik.

Aşağıdakı kimi:

```
if ((şərt1) && (şərt2) && (şərt3)) {
```

```
    emel1;
```

```
    emel2;
```

```
    ...
```

```
}
```

Əgər yuxarıdakı kod hissəsinin icra olunması üçün şərt1, şərt2, şərt3 -dən heç olmasa birinin ödənməsi kifayətdirsə onda biz bu şərtlərin hamısını və ya (||) operatoru ilə birləşdirə bilərik.



Aşağıdakı kimi:

```
if ((şərt1) || (şərt2) || (şərt3)) {  
    emel1;  
    emel2;  
    ...  
}
```

Qeyd edək ki, if operatorunun { } mötərəzələri arasında olan kod hissəsi ancaq və ancaq şərt ödəndiyi halda icra olunur. Lakin bir çox hallarda bizə lazım olur ki, bu şərtin əksi olan halda da hansısa əməliyyatları icra edək. Bu zaman biz **else** -dən istifadə edirik. **else** -dən proqramda sərbəst şəkildə istifadə edə bilmərik, **else** mütləq **if** -ə bitişik olmalıdır.

Şərt operatoruna aid proqram nümunələri:

```
//prg_3_1.cpp  
#include<iostream>  
  
int main()  
{  
    int x;  
    std::cout<<"zehmet olmasa x-in qiymetini daxil edin\n";  
    std::cin>>x;  
    if (x<5)  
        std::cout<<x<<" 5 -den kichikdir\n";  
    else  
        std::cout<<x<<" 5 -den boyukdur\n";  
    return 0;  
}
```

Proqramı kompilyasiya edək və hər dəfə müxtəlif qiymətlər daxil etməklə bir neçə dəfə yerinə yetirək.

```
C:\cpp\prog2\Debug>prog2.exe  
zehmet olmasa x-in qiymetini daxil edin  
6  
6 5 -den boyukdur  
C:\cpp\prog2\Debug>  
C:\cpp\prog2\Debug>prog2.exe  
zehmet olmasa x-in qiymetini daxil edin  
4  
4 5 -den kichikdir  
C:\cpp\prog2\Debug>  
C:\cpp\prog2\Debug>
```

### Müqaisə əməlləri

C++ dilində iki ifadənin qiymətini müqaisə etmək üçün aşağıdakı müqaisə əməllərdən istifadə olunur:

```
== bərabərdir  
!= fərqlidir  
> böyükdür  
< kiçikdir  
<= kiçikdir bərabərdir  
>= böyükdür bərabərdir
```

Misal üçün əgər mən x və y dəyişənlərinin qiymətlərinin bərabər olduğunu yoxlamaq istəsəm onda aşağıdakı şərtədən istifadə edəram:

```
if (x==y) {...}
```

## 3.2 Dövr operatorları

### for, while, do while

Dövr operatorları müəyyən əməliyyatların bir neçə dəfə təkrar yerinə yetirilməsinə imkan verir. Bu operatorlardan proqramlaşdırmada çox geniş istifadə olunur.

### for operatoru

for operatorunun sintaksisi aşağıdakı kimidir:

```
for (sayğacın ilkin qiyməti; dövrün başa çatma şərti; sayğacın dəyişmə qaydas) { əməliyyatlar; }
```

Nümunə proqram:

```
//prg_3_2.cpp
```

```
#include<iostream>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
int k;
```

```
for (k=0; k<10; k=k+1)
```

```
{
```

```
std::cout<<"salam dunya\n";
```

```
}
```

```
return 0;
```

```
}
```

proqramı kompilyasiya edib yerinə yetirək.

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
salam dunya
```

```
C:\cpp\prog2\Debug>
```

### Tapşırıq:

Yuxarıdakı proqramda for (k=0; k<10; k=k+1) şərtini for (k=5; k<10; k=k+1) ,for (k=0; k<3; k=k+1),for (k=0; k<10; k=k+7), for (k=0; k<10; k=2) sətirləri ilə əvəz edib proqramı icra edin. Hər dəfə müxtəlif nəticələr alacağıq. Sonuncu halda isə ekranda **Salam dunya** ifadəsi sonsuz çap olunacaq(proqramın icrasını dayandırmaq üçün **CTRL + Z** düyməsini daxil edin).

Birinci halı təhlil edək.

```
for (k=0; k<10; k=k+1)
```

Sayğac olaraq k dəyişənindən istifadə olunur və ona başlanğıc qiymət olaraq 0 mənimsədilir ( k=0;).

Dövrün sona çatması şərti kimi k<10; göstərilib. Bu o deməkdir ki nə qədər ki, k < 10 şərti ödənilir dövrə verilmiş əməliyyatlar təkrar icra olunacaq. Sayğacın qiymətinin dəyişmə qaydası kimi k=k+1 (! sonda ; simvolunun yoxluğuna diqqət yetirin) göstərilib, yəni dövr hər dəfə təkrar olunduqda sayğacın qiyməti 1 vahid artır.

Proses aşağıdakı şəkildə baş verir: Əvvəlcə k dəyişəni 0 qiyməti alır. Sonra dərhal k<10 şərti yoxlanılır. Şərt ödənilir, belə ki, 0 < 10 ifadəsi doğrudur və avtomatik olaraq dövr operatorunun

əməliyyatları icra olunur(ekranda "salam dünya" sətəri çap olunur). Daha sonra növbə sayğacın qiymətinin dəyişməsinə gəlir. Bunun üçün biz **for** -da  $k = k + 1$ ; yazmışıq. Bu əməliyyat  $k$  -in əvvəlki qiyməti nə idisə onu 1 vahid artırır. Baxdığımız hal üçün (  $k$  -nin qiyməti 0 olan hal )  $k$  -nin yeni qiyməti 1 olur. Şərt yoxlanılır. Bu proses  $k$  10 qiyməti alana kimi davam edir. Bu zaman  $k < 10$  şərti ödənmir və **for** operatoru sona çatır.

### **while** operatoru

**for** operatorunda biz sayğac təyin etdik, dövrün başa çatması şərtini və sayğacın dəyişmə qaydasını verdik. Bəzən isə elə olur ki, dövrün başa çatması şərtinin nə vaxt ödənəcəyi əvvəlcədən bilinmir. Bu zaman **while** operatorundan istifadə olunur.

**while** operatorunun sintaksisi aşağıdakı kimidir.

```
while(şərt){
```

```
  əməliyyatlar; }
```

nümunə:

```
char x;
```

```
x='b';
```

```
while (x!='a'){
```

```
  std::cout<<"Salam dünya\n";
```

```
  std::cout<<" yeni simvol daxil edin\n";
```

```
  std::cin>>x;
```

```
}
```

Bu kod icra olunduqda ekranda **Salam dünya** sətəri çap olunacaq və proqram istifadəçinin hər hansı simvol daxil etməsini gözləyəcək. Əgər bu simvol 'a' -dirsə dövr sona çatacaq əks halda dövr təkrar olunacaq .

### **Tapşırıq :**

Bu kodu yoxlamaq üçün proqram tərtib edib, icra edin.

**while** dövr operatorunun digər forması **do while** operatorudur. **do while** operatorunun sintaksisi belədir:

```
do{
```

```
  əməliyyatlar;
```

```
} while(şərt);
```

Bu operatorun **while** operatorundan yeganə fərqi odur ki, bu halda şərtin nə zaman ödənməsindən asılı olmayaraq əməliyyatlar ən azı 1 dəfə yerinə yetiriləcək.

### **3.3 switch operatoru**

Əgər müəyyən halda proqramın icra istiqaməti bir neçə şərtədən asılıdırsa bu zaman **if** operatoru ilə bu şərtlərin mürəkkəb konfigurasiyasından istifadə etmək əvəzinə **switch** operatorundan istifadə edirlər.

**switch** operatorunun sintaksisi aşağıdakı kimidir:

```
switch ( dəyişən ) {
```

```
  case qiymət1:
```

```
  /* əgər dəyişənin qiyməti == qiymət1 */
```

```
  yerinə yetirilməli proqram hissəsi
```

```
  break;
```

```
  case qiymət2:
```

```
  /* əgər dəyişənin qiyməti == qiymət2 */
```

```
  yerin yetirilməli proqram hissəsi
```

```
  break;
```

```
  ...
```

```
  default:
```

```
  /* yuxardakı şərtlərin heç biri ödənmədikdə */
```

```
  yerinə yetirilməli proqram hissəsi
```

```
}
```

**switch** operatoru dəyişənin qiymətini yuxarıdan aşağı **case** ifadəsinin qarşısında dayanan qiymətlə yoxlayır və bərabər olarsa onda iki nöqtə : - dən sonra gələn bütün operatorları yerinə yetirir. **break** rast gəlinən yerdə **switch** operatoru işini dayandırır və proqramda **switch** -dən sonra gələn operator yerinə yetirilir. **switch** operatoru ilə bağlı mühüm məqamlardan biri də odur ki, **case** ifadələrində qiymət kimi ancaq tam tipli dəyişənlərdən istifadə etmək olar (**int**) . **default** seçiminə istifadə etmək vacib deyil. Əgər qiymətlərdən heç biri ödənməsə onda **default**: seçiminə göstərilən operatorlar yerinə yetiriləcək.

nümunə proqram:

```
//prg_3_3.cpp
#include <iostream>

int main()
{
int color = 0;
std::cout<<"Her hansı reng sechin(qirmizi=1,yashil=2,qara=3):\n";
std::cin>>color;
switch(color){
case 1: std::cout<<"siz qirmizi rengi secdiniz\n";
break;
case 2: std::cout<<"siz yaşıl rengi secdiniz\n";
break;
case 3: std::cout<<"siz qara rengi secdiniz\n";
break;
default: std::cout<<"siz heç bir reng sechmediniz\n";
}
return 0;
}
```

Proqramı kompilyasiya edib yerinə yetirək.

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Her hansı reng sechin(qirmizi=1,yashil=2,qara=3):
2
siz yaşıl rengi secdiniz
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### 3.4 continue və break

**continue** və **break** operatorları dövr operatorlarının daxilində istifadə olunur.

**break** operatoru icra olunduqda dövr dərhal başa çatır.

**continue** operatoru icra olduqda dövr daxilində **continue** -dən sonra gələn əməliyyatlar icra olunmadan dövr yeni tsiklə keçir.

### 3.5 Şərtin qiyməti

Şərt və dövr operatorlarında diqqət yetirilməli digər bir məqam da şərtin qiyməti ilə bağlıdır. C++ dilində hər şey rəqəmlərlə göstərilir - ünvan, məlumat, sətir, simvol ... Şərtlər də istisna deyil: 0 - yanlış, 0 -dan böyük qiymətlər isə doğru kimi qəbul olunur. Misal üçün aşağıdakı yazılış doğrudur (Bu kod icra olunduqda heç nə yerinə yetirilməyəcək, çünki şərt ödənmir 0-dır)

```
if (0)
{
std::cout<<"Bakıda havalar yaxşı keçir \n";
}
```

Bu kod isə icra olunduqda sonsuz sayda "Bakida havalar soyuq keçir" sətri çap olunacaq(şərt həmişə ödənilir - 1).

```
while(1)
{
std::cout<<"Bakida havalar soyuq keçir \n";
}
```

## Çalışmalar

if operatoru

1. Elə proqram yazın ki, istifadəçidən 2 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.
2. Elə proqram yazın ki, istifadəçidən 3 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.
3. Elə proqram yazın ki, istifadəçidən 5 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.

for operatoru

4. Elə proqram qurun ki, istifadəçinin daxil etdiyi ədəd sayda ekranda 'a' simvolu çap etsin.
5. Elə proqram qurun ki, 1 ilə 100 arasında olan ədədlər içərisində 3-ə qalıqsız bölünən ədədləri çap etsin.
6. Elə proqram qurun ki, 1 ilə 1000 arasında istifadəçinin daxil etdiyi ədədə qalıqsız bölünən ədədləri çap etsin.

for və if operatorları

7. Elə proqram qurun ki, istifadəçidən hər-hansı ədəd qəbul etsin. Əgər bu ədəd 100-dən böyük olarsa onda ekranda 100 dəfə 'c' simvolu çap etsin, 50 ilə 100 arasında olarsa ekranda həmin ədəd sayda 'b' simvolu çap etsin, 50 -dən kiçik olarsa həmin ədəd sayda 'a' simvolu çap etsin.
- 8.(\*). for dövr operatorundan istifadə etməklə ekranda sonsuz olaraq "unix" kəlməsini çap edən proqram yazın.
9. while dövr operatorundan istifadə etməklə ekranda sonsuz olaraq "linux" kəlməsini çap edən proqram yazın.

## \$4 Ünvan dəyişənləri - Göstəricilər.

### 4.1 Dinamik və Adi dəyişənlər

Gələcəyin güclü proqramçıları ilə tələbələrin ayrıldığı yerə çatdıq.

Proqramda istifadə olunan dəyişənlər yaddaşda yer ayrılmasına görə iki cür olur, adi və dinamik . Bizim indiyə qədər istifadə etdiyimiz dəyişənlər hamısı adi dəyişənlərdir. Aşağıda onların müqaisəsi verilir.

Adi dəyişənlər :

ancaq proqramın əvvəlində onlara yer ayrılır,proqramın icrası boyu onlara ayrılan yer olduğu kimi qalır və bu yer dəyişəndən geri alınıb hansısa başqa məqsəd üçün istifadə oluna bilməz, proqramın icrası boyu yaddaşda eyni bir ünvanı istinad edirlər, bu dəyişənlərin istinad etdiyi ünvanı dəyişdirmək olmaz.

Dinamik dəyişənlər :

Dinamik dəyişənləri proqramın icrasının istənilən anında yaratmaq olar,onlara ayrılmış yaddaş proqramın icrasının istənilən anında geri alıb həmin yeri istənilən digər məqsəd üçün istifadə etmək olar,dinamik dəyişənlərin yaddaşda istinad etdikləri ünvanı istənilən digər ünvanı dəyişdirmək olar, hətta digər proqramın və ya nüvənin yaddaş sahəsinə.Ancaq buna etmək istədiyimiz ilk cəhddə nüvə proqramımızı təməmilə söndürər(başqa proqramların məlumatlarına icazəsiz müraciət etmək olmaz).

Ünvan dəyişənlərinin özəlliyi odur ki, onlar özlərində məlumat olaraq bizim üçün əhəmiyyətli olan məlumatın ünvanını saxlayırlar. Əgər biz bu ünvanı dəyişsək onda onlar ayrı məlumata istinad edəcəklər. Bu C/C++ dillərinə xarakterik olan, proqramçıya yaddaşla istədiyi kimi manipulyasiya etmə imkanı yaradan güclü vasitələrdən biridir.

Proqramda hər hansı tiptən ünvan dəyişəni elan etmək üçün aşağıdakı sintaksisdən istifadə olunur:

**tip \*dəyişən;**

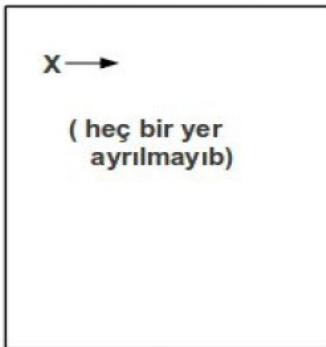
Tam tipli hər-hansı ünvan dəyişəni elan edək:

**int \*x;**

Göründüyü kimi bunun adi dəyişən elan etmək qaydasından ( int x; ) fərqi ancaq dəyişən adının əvvəlində \* - ulduz simvolunun olmasıdır.

Bu zaman yaddaşın vəziyyəti belədir(şəkil 5):

**Fiziki yaddaş**

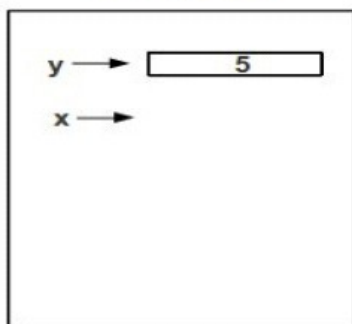


İndi mən x ünvan dəyişənini istənilən yaddaş ünvanına və ya istənilən dəyişənin yaddaş sahəsinə mənimsədə bilərəm və ya yaddaşda dinamik şəkildə əlavə yer ayıra və həmin yerə mənimsədə bilərəm.

Misal üçün gəlin adi int tipli y dəyişəni elan edək( int y; ) və onun yaddaş sahəsinə 5 qiyməti yazaq( y = 5; ).

Yaddaşın vəziyyəti:

Fiziki yaddaş



İndi mən x -i y -in yaddaş sahəsinə mənimsədə bilərəm.

Bunun üçün & ünvan operatorundan istifadə edəcəm.

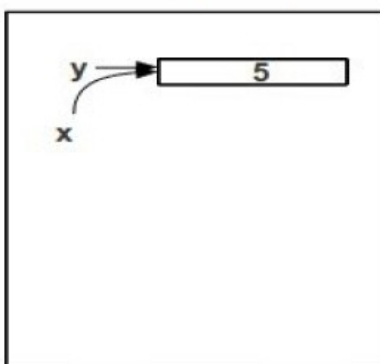
& operatoru istənilən dəyişənin və funksiyanın ünvanını almaq üçün istifadə olunur.

x -i y -in ünvanına mənimsətmək üçün sadəcə olaraq yazırıq:

```
x = &y ;
```

Yaddaşın vəziyyəti:

Fiziki yaddaş



Əgər bizə ünvan dəyişəninin istinad etdiyi ünvana müraciət etmək lazımdırsa onda sadəcə ünvan dəyişəninin adından istifadə edirik. Əgər biz həmin ünvanda yerləşən məlumata müraciət etmək istəyiriksə bu zaman dəyişənin əvvəlinə elanda olduğu kimi \* simvolu artırırıq. Gəlin bu dediklərimizi proqram nümunəsində test edək.

```
//prg_4_1.cpp
#include<iostream>

int main(int argc, char *argv[])
{
    int *x; // tam tipli unvan deyisheni
    int y;
    y=5;
    x=&y; // & unvan operatorudur
    // y -in qiymetini cap edek
    std::cout<<"y-in qiymeti = "<<y;
    // x-in qiymetini cap edek (aha y-in unvani)
    std::cout<<"\n x-in qiymeti (y-in unvani) "<<x;
    // x ünvanında yerləşən məlumatı çap edək
    std::cout<<"\n x-in istinad etdiyi məlumat "<<*x;
    // x-in istinad etdiyi məlumatı deyişək
    *x=67;
```

```
// y -in qiymetini cap edek
std::cout<<"y-in yeni qiymeti = "<<y;
return 0;
}
```

Programı icra edək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
y-in qiymeti = 5
x-in qiymeti (y-in unvani) 0x7ffffb4cbe824
x-in istinad etdiyi melumat 5
y-in yeni qiymeti = 67
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

Burada x-in qiyməti olaraq **0x7ffffb4cbe824** çap olunmuşdur. Ünvan ədədlə ifadə olunur və adətən 16-lıq say sisteminə göstərilir(hex). Mövzudan kənar olduğuna görə biz fiziki yaddaşın strukturu və fiziki, virtual ünvanlar barəsində söhbət açmayacağıq.

\* (çətinliyi artırılmış mövzudur, bu hissəni oxumadan 4.2 -yə keçə bilərsiniz)

Biz dedik ki, ünvan dəyişənlərinə hər hansı adi dəyişənin və ya funksiyanın ünvanını mənimsətmək üçün & operatorundan istifadə olunur. Gəlin int tipindən olan x dəyişəni elan edək və ona programın icraya başlama ünvanını (main funksiyanın ünvanını) mənimsədək və bu ünvanı çap edək.

```
#include<iostream>

int main(int argc, char *argv[])
{
int *x;
x = &main;
std::cout<<x;
return 0;
}
```

İcra edək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
0x4004c4
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

Başqa bir program tərtib edin və onun icraya başladığı ünvanı çap edin, eyni nəticə alacaqsınız. Sual: necə ola bilər ki, iki müxtəlif program yaddaşda eyni ünvandan icra olunmağa başlasın? - Cavab virtual yaddaş.

## 4.2 Dinamik yaradılma

Biz ünvan dəyişənlərinə qiymət mənimsədilməsi, həmin ünvanda yerləşən məlumata müraciət ilə tanış olduq və qeyd elədik ki, ünvan dəyişənləri programın icrası zamanı dinamik yaradıla və yaddaşdan silinə bilər. Ünvan dəyişənləri üçün yaddaşdayer ayırmaq üçün new funksiyanından istifadə eirlər.



Sintaksis aşağıdakı kimidir:

```
unvan_deyisheni = new tip;
```

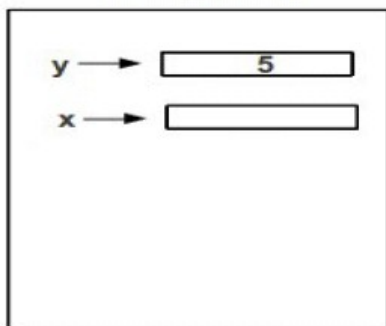
Aşağıdakı kimi:

```
int x;
```

```
x = new int;
```

Bu zaman yaddaşın vəziyyəti belə olar:

**Fiziki yaddaş**



### 4.3 Dinamik silinmə

Ünvan dəyişəni üçün ayrılan yaddaş sahəsini silmək üçün **delete** funksiyasından istifadə edirlər.

Sintaksis belədir:

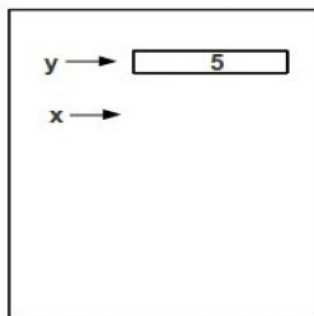
```
delete unvan_deyisheni;
```

Misal üçün :

```
delete x;
```

Yaddaşın vəziyyəti:

**Fiziki yaddaş**



Biz demək olar ki, ünvan dəyişənlərlə bağlı bir çox əsas anlayışlarla tanış olduq.

İrəlidəki paragraflarda biz bu biliklərimizi daha da möhkəmləndirəcəyik.

İndi isə proqram nümunələri ilə tanış olaq.

Bu proqramların hər biri üzərində ayrı-ayrılıqda işləmədən növbəti paragrafa keçmək məsləhət deyil.

Aşağıdakı proqramları icra edib yerinə yetirin və onların nə etdiyini izah edin.

```
//prg_4_2.cpp
```

```
#include <iostream>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
int x;
```

```

int *y;
// tam tipli unvan deyisheni
x=5;
std::cout<<"x-in qiymeti = <<x<<"\n";
y=&x;
// & unvan operatorudur
*y=155;
std::cout<<"y-in istinad etdiyi qiymet = <<*y<<"\n";
// y x-in yaddshina unvanlandigindan(y=&x)
// *y i deyismek birbasha x-i deyishdirir.
std::cout<<"x-in qiymeti = <<x<<"\n"
return 0;
}

```

```

//prg_4_3.cpp
#include <iostream>
int main(int argc, char *argv[])
{
int x;
int *y;
x=5;
y= new int; // dinamik yaradilma
*y=176;
std::cout<<"y-in istinad etdiyi qiymet = <<*y<<"\n";
delete y;
// y -e ayrilan yaddash sahesini y-den azad edir
// ve bu sahede olan butun melumat silinir
return 0;
}

```

#### 4.4 Ünvan dəyişənlərinin qiymətlərinin dəyişdirilməsi

Biz qeyd etdik ki, ünvan dəyişənləri özlərində qiymət olaraq fiziki yaddaşın hansısa ünvanın saxlayır. Bu ünvanda hər-hansı məlumat və ya proqram kodu yerləşə bilər. Həm də biz qeyd elədik ki, ünvan dəyişənlərinin istinad etdikləri bu ünvanı yaddaşın istənilən başqa yerinə yönləndirə bilərik. Gəlin proqram kodlarında bütün bu dedikləmizlə əyani tanış olaq.

```

// prg_4_4.cpp
#include<iostream>

int main()
{
int x,y,*z;
x=35;
y=68;
// x ve y-in qiymetlerini cap edek
std::cout<<" x-in ve y-in qiymetleri "<<x<<" , "<<y<<"\n";
// evvelce z -te x-in unvanin menimsedek
z=&x;
// bu unvanda olan melumati cap edek
std::cout<<z<<" unvaninda olan melumat "<<*z<<"\n";
// sonra z -te y-in unvanin menimsedek
z=&y;
// bu unvanda olan melumati cap edek
std::cout<<z<<" unvaninda olan melumat "<<*z<<"\n";
return 0;
}

```

Programı icra edək:

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
x-in və y-in qiymətləri 35 , 68
```

```
0x7fff036b1d54 ünvanında olan məlumat 35
```

```
0x7fff036b1d50 ünvanında olan məlumat 68
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

İzahı:

Programda *x* və *y* adlı iki adi dəyişən və *z* ünvan dəyişəni elan edirik. *x* və *y*-ə qiymətlər mənimsədirik. Daha sonra *z*-tə əvvəlcə *x*-in ünvanını mənimsədirik və bu ünvanda yerləşən məlumatı çap edirik, daha sonra *z*-tə *y*-in ünvanını mənimsədirik. Gördüyümüz kimi ünvan dəyişənlərinin özlərinin qiymətlərini dəyişməklə yaddaşın müxtəlif ünvanlarına istinad etmək olar.

### **Çalışmalar**

1. Ancaq ünvan dəyişənlərindən istifadə etməklə iki ədədin cəmini hesablayan program tərtib edin.
2. Ancaq ünvan dəyişənlərindən istifadə etməklə iki ədədin maksimumunu hesablayan program tərtib edin.

## \$5 Funksiyalar.

C++ dilinin proqramçılar arasında ən məşhur dil olmasında rol oynayan 2 ən güclü imkanından biri funksiyalardır. Funksiyalar bizə proqramın istənilən yerindən digər hissəsinə (funksiyaya) müraciət etməyə imkan verir. Proqramda funksiyadan istifadə etmək üçün biz əvvəlcə funksiyanı elan etməliyik. Daha sonra isə funksiyanın proqram kodunu tərtib etməliyik.

### 5.1 Funksiyanın elanı

C++ dilində funksiya aşağıdakı kimi elan olunur:

**Nəticənin\_tipi funksiyanın\_adi ( tip1 argument1, tip2 argument2, ...);**

Burada **nəticənin\_tipi** funksiyanın qaytaracağı nəticənin tipini göstərir. Əgər funksiya heç bir nəticə qaytarmırsa onda **nəticənin\_tipi** olaraq **void** yazırıq.

**funksiyanın\_adi** olaraq ingilis əlifbasının hərflərindən, rəqəmlərdən, \_ simvolundan istifadə edə bilərik. Funksiya adı mütləq ingilis əlifbası hərfi ilə başlamalıdır və operator adları ilə üst-üstə düşməməlidir. Funksiyanın adından sonra mötərizə daxilində funksiyanın qəbul edəcəyi arqumentlərin siyahısı verilir. Arqumentlər bir-birindən vergüllə ayrılır. Arqumentlərin əsas tipi önəmlidir. Funksiyanın elanında arqumentlərə verilən adlar heç bir əhəmiyyət daşımır və onlar buraxıla bilər.

Aşağıdakı kimi:

**Nəticənin\_tipi funksiyann\_ad ( tip1 , tip2 , ...);**

Nümunə:

**int cem (int x, int y);**

Burada biz **int** tipli nəticə qaytaran və **int** tipli iki arqument qəbul edən **cem** funksiyası elan elədik. Biz bunu aşağıdakı kimi də yazı bilərik, burada ki arqumentlərin adları göstərilir.

**int cem (int , int );**

### 5.2 Funksiyanın mətn kodunun tərtibi

Funksiyayı elan etməklə biz kompilyatora funksiya (ad, tip, arqumentlər) barəsində məlumat veririk. Funksiyanın mətn kodunu tərtib etməklə biz onun görəcəyi işi proqramlaşdırmış oluruq. Bunun üçün aşağıdakı qaydadan istifadə edirik:

```
Nəticənin_tipi funksiyann_ad ( tip1 arg1, tip2 arg2) {  
    proqram kodu  
    return nəticə;  
}
```

Burada ilk sətir funksiyanın elanı sətridir. Fərq yalnız odur ki, mötərizədən sonra ; deyil { simvolu gəlir. { simvolu funksiyanın proqram kodu blokunun başlanğıcını bildirir.

{ simvolundan sonra funksiyanın proqram kodu yerləşdirilir. Burada biz adi halda olduğu kimi istənilən proqram kodu yerləşdirə bilərik və hətta digər funksiyalara müraciət də edə bilərik. Bundan əlavə biz funksiyanın öz kodu daxilində onun özünə müraciət də edə bilərik. Buna proqramlaşdırmada **rekursiya** deyirlər.

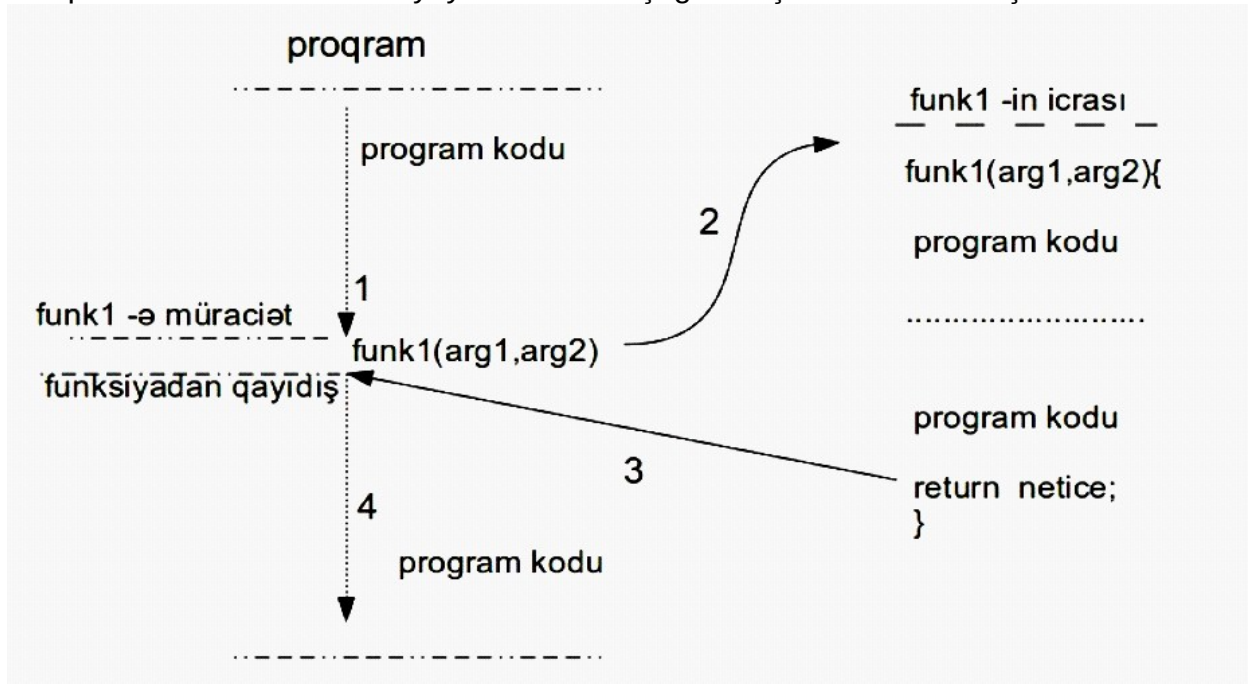
Gəlin yuxarıda elan etdiyimiz **cem** funksiyasının proqram kodunu tərtib edək:

```
int cem ( int x, int y)  
{  
    int z;  
    z = x + y;  
    return z;  
}
```

Burada funksiyanın daxilində **int** tipli **z** dəyişəni elan etdik. Daha sonra **z** dəyişəninə funksiyanın arqumentlərinin (**x** və **y**) cəmini mənimsətdik və alınmış qiyməti nəticə olaraq qaytardıq.

### 5.3 Return əmri, funksiyadan geri qayıtma

Proqramda hər- hansı funksiyaya müraciət aşağıdakı şəkildəki kimi baş verir:



Proqram kodu icra olunur və hansısa yerdə funksiyaya müraciət olunur. Bu zaman proqramın hal-hazırda icra olunan instruksiyasının ünvanı yadda saxlanılır və icarolunma çağırılan funksiyanın kodu yerləşən hissəyə ötürülür. Funksiya öz işini yekunlaşdırdıqdan sonra isə icraolunma yenidən proqramın funksiya çağırılan yerinə qaytarılır(həmin yerin ünvanı yaddaşa yerləşdirilmişdi).

Funksiyayı çağırmaq üçün biz onun adından istifadə edirik(aşağıdakı proqram nümunəsinə bax).Bəs icraolunma funksiyadan onu çağıran kod hissəsinə geri necə ötürülür? Funksiyayı çağırmaq və ondan geri qayıtmaq üçün prosessorun **call** və **ret** assembler instruksiyalarından istifadə olunur, lakin mən bu məsələdə çox dərinə getmək istəmirəm. Sadəcə olaraq onu bilməyimiz kifayətdir ki, funksiyanın daxilində istənilən yerdən geri qaytarmaq istəyiriksə (funksiyadan çıxmaq) **return** operatorundan istifadə edirik.

Funksiya daxilində **return** operatoru icra olunan yerdən sonra gələn hissələr yerinə yetirilmir.Biz dedik ki, funksiyanın tipi ya hər hansı tip , yada **void** ola bilər(funksiya heç bir nəticə qaytarmır).Əgər funksiya icra olunduqdan sonra hər hansı nəticə qaytarmalıdırsa bu da **return** operatoru vasitəsilə həyata keçirilir. Bu zaman funksiyanın qaytaracağı məlumatı **return** operatoruna argument kimi vermək lazımdır.

Aşağıdakı kimi:

```
return netice;
```

### 5.4 Funksiyalardan istifadə

Biz funksiyaların elanı , mətn kodunun tərtibi və funksiyadan qayıtmanın qaydalarını örgəndik.İndi isə gəlin funksiyalardan istifadə olunan proqram nümunələri ilə tanış olaq.

**Proqram nümunəsi:**

Funksiyadan istifadə etməklə iki ədədin cəmini hesablayan proqram:

```

//prg_5_1.cpp
#include <iostream>

/*cem funksiyasının elanı */
int cem (int x, int y);

int main (int argc, char *argv[])
{
int x,y,z;
std::cout<<"x -i daxil edin \n";
std::cin>>x;
std::cout<<"y -i daxil edin \n";
std::cin>>y;
/* cem funksiyasının çağırılması */
z = cem(x,y);
std::cout<<"x ile y -in cemi = "<<z<<"\n";
return 0;
}

//-----
/* cem funksiyasının kodu */
int cem ( int dey1, int dey2)
{
int dey3;
dey3 = dey1 + dey2;
return dey3;
}

```

Proqramı kompilyasiya edib yerinə yetirək:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
x -i daxil edin 7
y -i daxil edin 789
x ile y -in cemi = 796
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

### İzahı:

Baxdığımız proqramda əvvəlcə **cem** funksiyasının elanı sətiri yerləşir, daha sonra isə proqramın əsas funksiyası **main** funksiyası gəlir. Burada istifadəçi **x** və **y** dəyişənlərinin qiymətlərini daxil edir. Daha sonra bu dəyişənlər **cem** funksiyasına ötürülür, **cem** funksiyası bu dəyişənlərin cəmini hesablayır və nəticəni qaytarır.

## 5.5 Lokal və Qlobal dəyişənlər

Funksiyalardan istifadə edərkən bilməli olduğumuz vacib anlayışlardan biri də **lokal** və **qlobal** dəyişənlər anlayışıdır. Nədir **lokal** və **qlobal** dəyişənlər? **lokal** və **qlobal** dəyişən -nin nə olduğunu bilmək üçün biz **blok** anlayışını daxil etməliyik. C++ dilində { və } mötərəzələri arasında qalan hissə **blok** adlanır.

Əgər diqqət yetirsəniz, görürsünüz ki, funksiyanın mətn kodu bütövlükdə bir **blok** -dan ibarətdir. **blok** daxilində **blok** elan edə bilərik və bu zaman "içəridə" yerləşən **blok** -lar "üst" **blok** -lardakı dəyişənləri görür, "üst" **blok** -lar isə "içəri" **blok** -larda elan olunan dəyişənləri görmür.

Aşağıdakı kimi:

```

{
/* blok A */
int x;
/* x y-i gormur*/
{
/* blok B */ int y;
/* y ise x-i gorur, ona gore yaza bilerem*/
y = x;
{
/* blok C */
int z;
/* z -ti ne blok A ne de blok B gormur.*/
/* z ise x ve y-i gorur, ona gore yaza bilerem*/
z = x + y;
/* blok C -nin sonu*/
}
/* blok B -nin sonu */
}
/* blok A -nin sonu */
}

```

Biraz əvvəl daxil etdiyimiz proqrama nəzər salsaq görürük ki, **cem** funksiyası daxilində biz **int** tipli **dey3** dəyişəni elan etmişik. Aydın ki, bu dəyişən **main** funksiyası üçün lokaldır, yəni biz **main** funksiyasından və ümumiyyətlə proqramın **cem** funksiyasından başqa heç bir yerindən **dey3** -ə müraciət edə bilmərik.

## 5.6 Dəyişənlərin Ünvana və Qiymətə görə ötürülməsi

Yuxarıda baxdığımız proqramda biz **cem** funksiyasına argument olaraq **int** tipli iki dəyişən ötürdük. Funksiya bu dəyişənlərin cəmini hesablayıb nəticə olaraq qaytardı. Bu zaman biz dəyişənlərin funksiyaya qiymətə görə ötürülməsi qaydasından istifadə etdik. Bir çox hallarda isə bizə nəticə ilə yanaşı, funksiyanın ona ötürülən parametrlərin də qiymətlərini dəyişdirməsi tələb olunur.

Bu zaman isə biz dəyişənlərin funksiyaya **ünvana görə ötürülməsi** qaydasından istifadə etməliyik.

### Fərq nədədir?

#### Dəyişənlərin qiymətə görə ötürülməsi.

Dəyişəni qiymətə görə ötürəndə (indiyə kimi baxdığımız hal), dəyişənlərin nüsxələri (kopiya) yaradılır və funksiyaya bu nüsxələr ötürülür. Aydın məsələdir ki, bu zaman nüsxə üzərində aparılan heç bir əməliyyat dəyişənlərin orijinal qiymətlərinə təsir etmir. Bir tərəfdən bu yaxşıdır, çünki bu zaman biz dəyişənləri mühafizə etmiş oluruq. Amma pis cəhət odur ki, dəyişənlərin nüsxəsinin yaradılmasına həm əlavə vaxt, həm də yaddaşda əlavə yer ayrılır və iri həcmli dəyişənlər olanda bu qayda sərfəli olmur. Həm də əgər məsələnin tələbi ilə funksiya parametrlərinin qiymətlərinin dəyişdirilməsi lazım olsa qiymətə görə ötürmədə biz bunu edə bilmərik.

#### Dəyişənlərin ünvana görə ötürülməsi.

Ünvana görə ötürülmə zamanı isə funksiyaya ötürülən dəyişənlərin heç bir nüsxəsi yaradılmır, funksiyaya dəyişənlərin yaddaşdakı ünvanları ötürülür. Bu zaman funksiya daxilində dəyişən üzərində aparılan bütün əməliyyatlar funksiya bitdikdən sonra qüvvədə qalır. Dəyişənləri ünvana görə ötürmək üçün biz funksiyanı aşağıdakı kimi elan etməliyik:

**Nəticənin tipi funksiyanın adı ( tip1 \*arg1, tip2 \*arg2, ...);**

Dəyişənləri bu funksiya parametrlərinə kimi ötürəndə isə onların əvvəlinə ünvan - & operatoru əlavə etməliyik.

Nümunə:

Ünvana görə ötürülmə zamanı funksiyanın elanı:

```
int funk (int *, int *);
```

Funksiyaya müraciət:

```
int x,y;
```

```
funk(&x, &y);
```

Gəlin arqumentlərin funksiya qiymətə və ünvana görə ötürülməsinin fərqi baxmaq üçün program tərtib edək.

```
// prg_5_2.cpp
```

```
#include<iostream>
```

```
void funk1 (int);
```

```
void funk2 (int*);
```

```
int main(){
    int x=45;
    std::cout<<"x-in ilkin qiymeti "<<x<<"\n";
    funk1(x);
    std::cout<<"x-in funk1-den sonraki qiymeti "<<x<<"\n";
    funk2(&x);
    std::cout<<"x-in funk2-den sonraki qiymeti "<<x<<"\n";
    return 0;
}
```

```
// qiymete gore oturulme
```

```
void funk1 (int x){
```

```
    x=90;
```

```
    std::cout<<"x-in funk1 daxilinde qiymeti "<<x<<"\n";
```

```
}
```

```
// unvana gore oturulme
```

```
void funk2 (int *x){
```

```
    *x=200;
```

```
    std::cout<<"x-in funk2 daxilinde qiymeti "<<*x<<"\n";
```

```
}
```

Programı icra edək:

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
x-in ilkin qiymeti 45
```

```
x-in funk1 daxilinde qiymeti 90
```

```
x-in funk1-den sonraki qiymeti 45
```

```
x-in funk2 daxilinde qiymeti 200
```

```
x-in funk2-den sonraki qiymeti 200
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

İzahı:

funk1 -də arqument funksiya qiymətə görə, funk2-də isə ünvana görə ötürülür. Başqa sözlə funk1 -də arqumentin nüsxəsi yaradılır və funksiya nüsxə ötürülür, funk2 -də isə arqumentin ünvanı funksiya ötürülür. funk1 daxilində ona ötürülən dəyişən üzərində aparılan bütün dəyişikliklər funksiya qayıtdıqdan sonra itir. funk2 -də bütün dəyişikliklər bir başa dəyişənə aid yaddaş sahəsi üzərində aparıldığından funksiya qayıtdıqdan sonra ona ötürülən parametrlər üzərində aparılmış bütün dəyişikliklər qalır.



## \*5.7 Ünvan Dəyişənlərinin Funksiyaya Ünvana və Qiymətə görə ötürülməsi

(\*Çətinliyi artırılmış mövzudur) Biz adi dəyişənlərin funksiyaya qiymətə və ünvana görə ötürülməsi ilə tanış olduq. İndi isə gəlin ünvan dəyişənlərinin funksiyaya parametr kimi ötürülməsi və bu zaman qiymətə və ünvana görə ötürülmənin fərqi ilə tanış olaq.

```
// prg_5_3.cpp
```

```
#include<iostream>
```

```
void funk1(int *);  
void funk2(int **);
```

```
int x,y,*z;
```

```
int main(){  
x=35;  
y=68;  
std::cout<<"x-in ve y -in qiymetleri "<<x<<" , "  
    <<y<<"\n";  
std::cout<<"x-in ve y -in unvanlari "<<&x<<" , "  
    <<&y<<"\n";  
// z-te x-in unvanin menimsedek  
z=&x;  
std::cout<<"z-in ilkin qiymeti "<<z<<"\n"  
    <<"bu unvanda olan melumat "<<*z<<"\n";  
// unvan deyishenin qiymete gore oturulmesi  
funk1(z);  
std::cout<<"z-in funk1-den sonraki qiymeti "<<z<<"\n"  
    <<"bu unvanda olan melumat "<<*z<<"\n";  
// unvan deyishenin unvana gore oturulmesi  
funk2(&z);  
std::cout<<"z-in funk2-den sonraki qiymeti "<<z<<"\n"  
    <<"bu unvanda olan melumat "<<*z<<"\n";  
return 0;  
}
```

Programı icra edək:

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
x-in ve y -in qiymetleri 35 , 68
```

```
x-in ve y -in unvanlari 0x6013a0 , 0x6013a4
```

```
z-in ilkin qiymeti 0x6013a0
```

```
bu unvanda olan melumat 35
```

```
z-in funk1 daxilinde qiymeti 0x6013a4
```

```
bu unvanda olan melumat 68
```

```
z-in funk1-den sonraki qiymeti 0x6013a0
```

```
bu unvanda olan melumat 35
```

```
z-in funk2 daxilinde qiymeti 0x6013a4
```

```
bu unvanda olan melumat 68
```

```
z-in funk2-den sonraki qiymeti 0x6013a4
```

```
bu unvanda olan melumat 68
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

Izahı:

Proqramda global  $x, y$  dəyişənləri və  $z$  ünvan dəyişəni elan edirik. Əvvəlcə  $x$  və  $y$ -in qiymətlərini və ünvanlarını çap edirik. Daha sonra  $z$ -tə  $x$ -in ünvanın mənimsədirik. **funk1** -ə  $z$ -ti qiymətə görə parametr kimi ötürürük. **funk1**-in daxilində  $z$ -in qiymətin dəyişib  $y$ -in ünvanına mənimsədirik. **funk1**-dən sonra  $z$ -in qiyməti dəyişməmişdir, o əvvəlki kimi  $x$ -ə istinad edir. **funk2** -yə isə  $z$ -ti ünvana görə ötürürük, buna görə **funk2** daxilində  $z$  üzərində etdiyimiz dəyişiklik funksiya qayıtdıqdan olduğu kimi saxlanılır.

### Çalışmalar:

1. Funksiyalardan istifadə etməklə iki ədədin maksimumunu hesablayan proqram tərtib edin.
2. Elə funksiya qurun ki, istifadəçidən 10 tam ədəd daxil etməsini istəsin və onların cəmini qaytarsın. Bu funksiya istifadə etməklə proqram qurun və onu icra edin.
3. kvadrat adlı elə funksiya tərtib edin ki, ekranda \* simvollarından ibarət, tərəflərinin uzunluğu 10 olan, kvadrat çəksin (içini doldurmaqla). kvadrat funksiya istifadə etməklə proqram qurub icra edin.
4. Çalışma 3-dəki kvadrat funksiyanı elə dəyişin ki, tərəflərinin sayı və təşkil olunduğu simvol bu funksiya parametr kimi ötürülsün. Bu funksiya istifadə etməklə elə proqram qurun ki, istifadəçidən hər hansı simvol və ədəd daxil etməsini istəsin, daha sonra isə ekranda həmin parametrlərə uyğun kvadrat çəksin.
5. Çalışma 4-dəki funksiyanı elə dəyişin ki, istifadəçi kvadratın içinin rəngləndiyi simvolu da daxil edə bilsin. Bu funksiya istifadə etməklə proqram tərtib edib, icra edin.
6. Çalışma 5-in tələblərini yerinə yetirən romb funksiya qurun, hansı ki, ekranda romb çəksin. Bu funksiya istifadə edib proqram tərtib edin və icra edin.
7. Çalışma 5-in tələblərini yerinə yetirən ucbucaq funksiya qurun, hansı ki, ekranda ucbucaq çəksin. Bu funksiya istifadə edib proqram tərtib edin və icra edin.
8. Kvadrat, romb və ucbucaq funksiyalarından istifadə etməklə elə proqram tərtib edin ki, əvvəl istifadəçidən tərəfin uzunluğun, tərəfin və fiqurun daxil rəngləmək üçün simvolları daxil etməyi istəsin. Daha sonra istifadəçidən 1,2 və 3 rəqəmlərdən birini daxil etməyini istəsin. Əgər istifadəçi 1 daxil edərsə onda ekranda kvadrat, 2 daxil edərsə romb, 3 daxil edərsə ucbucaq çəksin.
- 9.(\*). Çalışma 8-i elə dəyişin ki, proqram istifadəçidən fiqurun tərəfinin uzunluğunu və rəng simvolları daxil etdikdən sonra istədiyi fiqurun çəkilməsi üçün 1,2,3 simvollarından birini daxil etməsini istəsin. Bu prosesi istifadəçi 0 rəqəmi daxil edənə kimi təkrar eləsin. Bu zaman proqramın istifadəçidən tərəfin uzunluğu və rəng simvollarını qəbul edən hissəsini də ayrı bir funksiya kimi tərtib edin.

## \$6 Cərgələr və ya Massivlər.

İndiyə kimi biz baxdığımız nümunə proqramlarda elə də çox sayda dəyişəndən istifadə etmədik. 1,2 və ya 3 dəyişən maksimum halda. Aydın ki, bütün bunlar nümunə proqramlardır və materialı izah etmək üçün verilir. Real tətbiqi proqramlar isə 10 000 -lərlə və ya 100 000 -lərlə müxtəlif məlumatlar üzərində əməliyyatlar aparmalı olur.

Sadə misala baxaq:

Tutaq ki, universitetdə bu il bütün fakültələr üzrə təhsil alan tələbələrin, bütün fənlər üzrə qış imtahanlarının yekun nəticələrinin orta qiymətini hesablamaq lazımdır. Əgər universitetdə 6000 tələbə, 5 fəndən imtahan veribsə bu 30 000 imtahan nəticəsi deməkdir. Başqa sözlə bu proqramı qurmaq üçün biz 30 000 dənə dəyişən elan etməliyik, hansı ki, praktiki olaraq mümkün deyil (əlbəttə mümkündür, amma heç kim bu işi görməz).

Bu zaman cərgələrdən istifadə olunur.

Cərgə elan edən zaman yaddaşda biz lazım olan sayda dəyişən (10, 5000, 100 000 ) yerləşdirmək üçün sahə ayırırıq və bu sahəyə bütövlükdə bir ad veririk. Dəyişənlər bu sahəyə ardıcıl düzülür, cərgə şəkilində. Daha sonra bu cərgədən istənilən elementə müraciət etmək üçün həmin sahəyə verdiyimiz addan və həmin dəyişənin indeksindən (cərgədəki sıra nömrəsindən) istifadə edirik.

Cərgədə dəyişənlərin nömrələnməsi 0 -dan başlayır, yəni cərgənin ilk elementinə müraciət etmək üçün cərgənin adı və 0 indeksindən istifadə olunur. Cərgə elan etmək üçün aşağıdakı çox sadə sintaksisdən istifadə edirik:

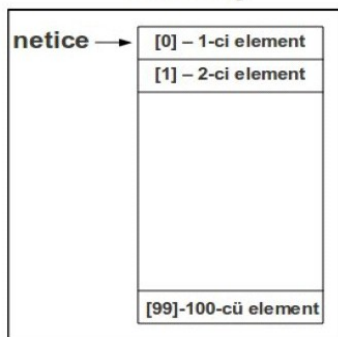
**tip cərgənin\_adi [elementlərin\_sayı];**

Misal üçün `int` tipli 100 elementdən ibarət `netice` adlı cərgə elan etmək üçün yazırıq:

`int netice[100];`

Yaddaşa nəzər salaq:

Yaddaş



Eyni sətirdə biz həm də adi dəyişən və ünvan dəyişəni elan edə bilərik:

`int x, netice[100], *dey;`

### 6.1 Cərgənin elementlərinə müraciət

Qeyd elədik ki, cərgənin elementlərinə müraciət etmək üçün cərgənin adı və müraciət etmək istədiyimiz elementin indeksindən istifadə edirik.

Aşağıdakı kimi:

`cergenin_adi[index];`

Misala baxaq :

```

int x, y[10], *dey;
// y cergesinin 1-ci elementine (indeks -0) 127 qiymeti menimsedek
y[0] = 127;
// y cergesinin 2-ci elementine (indeks -1) 600 qiymeti menimsedek
y[1] = 600;
// x -e y cergesinin 2-ci elementini menimsedek
x = y[1];

```

Göründüyü kimi cərgələrlə işləmək kifayət qədər maraqlı və asandır.

### Program nümunəsi:

Cərgədən istifadə etməklə , elə proqram tərtib edək ki, istifadəçidən 10 ədəd daxil etməsini xahiş etsin, daha sonra bu 10 ədədin cəmini hesablasın.

```

//prg_6_1.cpp
#include <iostream>

int main(int argc, char *argv[])
{
int cem,x[10],z,i;

std::cout<<"10 dene eded daxil edin \n";
// cergenin elementlerinin daxil edilmesi
for(i=0; i<10; ++i)
std::cin>>x[i];

// cergenin elementlerinin ceminin hesablanmasi
// evvelce cem -deyishenin 0-ra menimsedirik
cem = 0;
for(i=0; i<10; ++i)
cem = cem + x[i];

std::cout<<"daxil etdiyiniz ededlerin cemi = "<<cem<<" \n";
return 0;
}

```

Proqramı icra edək:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
10 dene eded daxil edin
1 23 4 56 7 78 9 7 234 6
daxil etdiyiniz ededlerin cemi = 425
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## 6.2 Cərgələrin elan olunma qaydaları

Cərgələr aşağıdakı kimi elan oluna bilər:

Birinci qayda:

```
int x[10];
```

İkinci qayda:

```
int x[]={1,2,3,4,5,6,7,8,9,10};
```

Bu zaman cərgənin elementlərinin sayı  $\square$  mütərəzələri arasında verilmir, onun təşkil olduğu elementlər  $\{ \}$  mütərəzələri arasında göstərilir, vergüllə ayrılaraq.

### 6.3 Cərgə ilə ünvan dəyişənləri arasında əlaqə

C++ dili ünvan dəyişənlərini çox sevir və hər şeyi onunla əlaqələndirməyə çalışır. Bu da səbəsiz deyil, ilk əvvəllər biraz çətin gəlsə də, ünvan dəyişənləri çox praktikdirlər. Cərgələr ilə ünvan dəyişənləri bir biri ilə çox əlaqəlidir və onların bu əlaqəsindən proqramlaşdırmada geniş istifadə olunur.

C++ dilində cərgənin adı onun ilk elementinə istinad edən ünvan tipli dəyişəndir.

Aşağıdakı kimi adı dəyişən, ünvan dəyişəni və cərgə elan edək, onların əlaqələrini izah etməyə çalışaq.

```
int x, *y, z[10];
```

z yalnız ilk elementə (z[0] -a) istinad edir və ayrı yerə ünvanlara bilməz. Bu deyilənə əsasən z cərgəsinin ilk elementinə z[0] -dən başqa aşağıdakı kimi də müraciət etmək olar.

```
*z;
```

Yəni adı x dəyişəninə z -in ilk elementini aşağıdakı iki yolla mənimsədə bilərik:

```
x = z[0]; və ya x = *z;
```

Ancaq ilk elementə əlbəttə ki;

Digər tərəfdən z ünvan tipli dəyişən olduğundan və özündə cərgənin ilk elementinin ünvanını saxladığından, mən y = z; yazsam onda y -də z cərgəsinin ilk elementinə istinad edəcək. Əgər mən y -i cərgənin ikinci elementi üzərinə sürüşdürmək istəyirəmsə onda yazıram:

```
y = y + 1; və ya y = z + 1;
```

Beləliklə y-i cərgənin ilk elementinə mənimsətməklə və onun üzərində artırma və azaltma əməlləri aparmaqla onu cərgənin elementləri boyu yuxarı - aşağı sürüşdürmək olar.

### 6.4 Cərgələrin funksiyaya parametr kimi ötürülməsi

Cərgələr funksiyaya parametr kimi ötürülə bilər. C++ dilində cərgələr funksiyaya ancaq bir yolla, ünvana görə ötürülə bilər. Tutaq ki, aşağıdakı kimi x cərgəsi elan etmişik.

```
int x[10];
```

Əgər hər hansı **funk** funksiyasına x cərgəsini parametr kimi ötürmək istəyiriksə, onda **funk** -da ünvan tipli parametr elan etməliyik.

```
int funk(int *);
```

ünvan dəyişənlərində olduğu kimi (cərgənin adının ünvan tipli dəyişən olması elə buradan da görünür, eyni elan ilə funksiyaya biz həm ünvan tipli dəyişən ötürə bilərik, həm də cərgə). Daha sonra x -i **funk** -a parametr kimi ötürmək istəsək, sadəcə **funk(x)**; yazırıq.

#### Proqram nümunələri:

Proqram 1. eks.c

Elə proqram tərtib edək ki, istifadəçinin daxil etdiyi rəqəmləri əks sıra ilə çap etsin:

```
//prg_6_2.cpp
```

```
#include <iostream>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
int i, x[100], say;
```

```
std::cout<<"100 -den kicik her hansı bir eded daxil edin \n";
```

```
std::cin>>say;
```

```
std::cout<<say<<" sayda eded daxil edin\n";
```

```
for (i=0; i<say; i++ )
```

```
std::cin>>x[i];
```

```
std::cout<<"sizin daxil etdiyiniz ededler eks siyahida \n";
```

```
for (i=(say-1); i>=0; i--)
```

```
std::cout<<x[i]<<" ";
```

```
std::cout<<"\n";
```

```
return 0;
```

```
}
```

Programı kompilyasiya edib icra edək:

```
C:\cpp\prog2\Debug>prog2.exe
100 -den kicik her hansı bir eded daxil edin > 0
6
6 sayda eded daxil edin
1 2 3 4 5 6
sizin daxil etdiyiniz edeler eks siyahıda
6 5 4 3 2 1
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## Program 2. max.c

Cərgələrdən istifadə edib ele program quraq ki, verilmiş sayda ədələrin içindən ən böyüyünü tapsın.

```
// prg_6_3.cpp
#include <iostream>

int main(int argc, char *argv[])
{
    int i, x[100], say, max;
    std::cout<<"100 -den kicik her hansı bir eded daxil edin \n";
    std::cin>>say;

    std::cout<<say<<" sayda eded daxil edin\n";
    for (i=0; i<say; i++ )
        std::cin>>x[i];

    max = 0;
    for (i=0; i<say; i++)
        if (x[i] > max) max = x[i];

    std::cout<<"sizin daxil etdiyiniz ededlerin icinde en boyuyu "
    <<max<<" -dir\n";
    return 0;
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
100 -den kicik her hansı bir eded daxil edin
7
7 sayda eded daxil edin
1 23 45 678 2 321 89
sizin daxil etdiyiniz ededlerin icinde en boyuyu 678 - dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### Program 3. max\_func.c

Program 2 -ni funksiyadan istifadə etməklə tərtib edək.

```
// prg_6_4.cpp
#include <iostream>

int max_eded(int *, int);

int main(int argc, char *argv[])
{
    int i, x[100], say, max;
    std::cout<<"100 -den kicik her hansı bir eded daxil edin \n";
    std::cin>>say;
    std::cout<<say<<" sayda eded daxil edin\n";
    for (i=0; i<say; i++ )
        std::cin >>x[i];
    max = max_eded(x,say);
    std::cout<<"sizin daxil etdiyiniz ededlerin icinde en boyuyu "
    <<max<<" -dir\n";
    return 0;
}

//-----

int max_eded(int *x, int say)
{
    int net,i;
    net = 0;
    for (i=0; i<say; i++)
        if (x[i] >net) net = x[i];
    return net;
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
100 -den kicik her hansı bir eded daxil edin
5
5 sayda eded daxil edin
23 456 7 89 0
sizin daxil etdiyiniz ededlerin icinde en boyuyu 456 - dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### Çalışmalar

1. max programını ele dəyişin ki, istifadəçinin daxil etdiyi ədələrin içində ən kiçiyini tapsın.
- 2.(\*) Elə program qurun ki, istifadəçinin daxil etdiyi ədədləri artan sıra ilə düzsün.
3. Elə program qurun ki, istifadəçinin daxil etdiyi ədədləri azalan sıra ilə düzsün.
4. Elə funksiya tərtib edin ki, verilmiş cərgənin elementləri arasında ən böyüyünü tapsın. Bu funksiyadan istifadə edərək program tərtib edin ki, istifadəçidən əvvəl say, daha sonra bu say qədər hər-hansı ədəd daxil etməsini istəsin və bu ədədlərin ən böyüyünü çap etsin.
5. 4 - cı programı elə dəyişin ki, istifadəçi say olaraq 0 daxil edənə kimi program təkrar olunsun.
6. Elə funksiya qurun ki, verilmiş siyahının elementlərini artan sıra ilə düzsün. Bu funksiyadan istifadə etməklə elə program qurun ki, istifadəçidən əvvəlcə say, daha sonra bu say qədər

ədəd daxil etməsini istəsin. Daha sonra proqram istifadəçinin daxil etdiyi ədədləri artan sıra ilə düzsün.

7. 6 - ci proqramı elə dəyişin ki, istifadəçi say olaraq 0 daxil edənə kimi proqram təkrar olunsun.

8.(\*). Verilmiş cərgənin elementlərini artan və azalan sırada düzən art və azl funksiyalarını tərtib edin. Daha sonra bu funksiyalardan istifadə edərək elə proqram qurun ki, istifadəçidən əvvəl say, daha sonra bu say qədər ədəd daxil etməsini istəsin.

Daha sonra istifadəçiyə seçim olaraq 1 və ya 2 rəqəmi daxil etməsini təklif etsin. Əgər istifadəçi 1 rəqəmini daxil edərsə onda ədədləri artan sırada, 2 daxil edərsə ədədləri azalan sırada çap eləsin.

9. 8 -ci proqramı elə dəyişin ki, istifadəçi proqramdan çıxmadan ondan bir neçə dəfə istifadə edə bilsin. 7-ci proqramdakı kimi.



## \$7 Sətirlər

Sətirlər cərgələrin xüsusi halıdır. Cərgə elementlərinin tipinin `char` olduğu hal. `char` tipi simvol tipi adlanır. Başqa sözlə sətirlər simvollar cərgəsidir. Sətirlərin tətbiq sahəsi çox genişdir, həm nüvədə, həm də istifadəçi proqramlaşdırmasında sətirlərdən çox geniş istifadə olunur. Elementlərinin hər biri 1 bayt yer tutduğuna görə müxtəlif məqsədlər üçün yaddaşda istənilən ölçülü yer ayırmaq lazım olanda həmin ölçülü sətir elan edib bu sətirdən bufer kimi istifadə edə bilərik. İstənilən məlumat baytlar ardıcılığı olduğundan, sətir kimi elan etdiyimiz yerə istənilən məlumat yazıla bilər.

Proqramda sətir elan etmənin 2 yolu var: `char` tipindən olan ünvan dəyişəni və ya cərgə elan etməklə.

Aşağıdakı kimi:

```
char setir1[100], *setir2;
```

Burada bizim üçün yeni heçnə yoxdu, bildiyimiz cərgə və ünvan dəyişəni elan etmişik. Sadəcə tip `char` olduğundan sətirlər üçün bir neçə standart funksiya təyin edilmişdir. Bu funksiyalardan istifadə edə bilmək üçün `string.h` faylını proqrama əlavə etməliyik.

### 7.1 Sətirlərə qiymətlərin mənimsədilməsi

Sətirlərə qiymətlər müxtəlif yollarla mənimsədilir: Adı biz bildiyimiz cərgənin hər bir elementinə qiymət mənimsətmə yolu ən az istifadə olunandır. Məsələn üçün, 20 simvoldan ibarət sətir elan edək, daha sonra bu sətirə "Azerbaycan" sözünü yerləşdirək. Qeyd eliyirəm bir daha ki, bu qayda ilə sətirə qiymət mənimsədilmir, birazdan bunun asan qaydasını göstərəcəyik. Sadəcə sətirlərlə cərgələrin əlaqəsini göstərmək üçün bu misalı veririk.

```
char set[20];  
set[0] = 'A';  
set[1] = 'z';  
set[2] = 'e';  
set[3] = 'r';  
set[4] = 'b';  
set[5] = 'a';  
set[6] = 'y';  
set[7] = 'c';  
set[8] = 'a';  
set[9] = 'n';  
set[10] = '\0';
```

Bu nümunədə sətirin sonuncu elementinə mənimsətdiyimiz `'\0'` simvolu xüsusi simvoldur və sətirlərin sonunu bildirmək üçün istifadə olunur. Aşağıda daxil edəcəyimiz sətir funksiyaları sətirin sonunu bu simvola görə təyin edir. Beləki sətir funksiyaları sətirdə `'\0'` simvoluna rast gəldikləri yerdən sətirin qalan hissəsini inkar edirlər.

Bu misalda sətirlərin adı `char` tipindən olan cərgə olmaları aydın görülür. Yuxarıdakı `set` sətirinə "Azerbaycan" sözünü biz asanlıqla elanda

```
char set[]="Azerbaycan";
```

kimi də mənimsədə bilərdik. Amma bu da sətirlərə qiymət mənimsətmək üçün əsas qayda deyil.

#### **strcpy() funksiyası**

Sətirlərə qiymət mənimsətmək üçün ən əsas istifadə olunan funksiya `strcpy()` funksiyasıdır. Bu yerdə sizin diqqətinizə xüsusi bir məsələni mütləq çatdırmalıyam. Şəbəkə hücumlarının geniş yayılmış müxtəlif növləri var. Bunlara DoS(Denial of Service - Xidmətdən imtina),Buffer

Overflow (buferi daşımaq) və digərlərini misal göstərmək olar.

C++ dilində yazılmış proqramların zəif tərəflərindən biri **strcpy()** funksiyasıdır. Beləki, pis məqsədli hər kimsə, **strcpy()** funksiyasına parametr olaraq böyük ölçülü sətir verə bilər. Nəticədə **strcpy()** həmin sətiri göstərilən ünvandan proqramın daxilinə köçürəndə proqramın bütün ehtiyat yaddaşın doldurur və nəticədə proqram dayanmağa məcbur olar. Çıxış yolu **strcpy()** əvəzinə **strncpy()** funksiyasından istifadə etməkdir.

**strncpy()** funksiyası verilmiş ünvandan ən çoxu verilmiş sayda simvol köçürməyə icazə verir. Sadə məsələlərdə **strcpy()** -dən istifadə edə bilərsiniz, amma ciddi projelərdə mütləq **strncpy()** -dən istifadə etmək lazımdır.

**strcpy()** funksiyasının sintaksisi aşağıdakı kimidir:

```
char * strcpy(char *menseb, char *menbe);
```

Burada **menbe** köçürüləcək sətirin ünvanı, **menseb** sətiri köçürmək istədiyimiz ünvandır.

Proqram nümunəsi:

```
// prg_7_1.cpp
#include <iostream>
#include <string.h>

int main()
{
//15 simvoldan ibaret sətir elan edirik
char set[15];

strcpy(set, "Ekvator");
std::cout<<set;

return 0;
}
```

**strlen() funksiyası.**

```
int strlen(char *s);
```

Bu funksiya **s** sətirində olan simvolların (baytların) sayını qaytarır. '\0' simvolunu hesaba almır.

**strncpy() funksiyası.**

```
char * strncpy(char *s1, char *s2, int n);
```

**s2** sətirinin ilk **n** elementini **s1**-ə köçürür (**s1**-in əvvəlindən başlayaraq).

Qeyd: bu funksiya sətirin sonuna '\0' simvolunu yazmır.

**strcmp() funksiyası.**

```
int strcmp(char *s1, char *s2);
```

Bu funksiya sətirlərin müqaisəsi üçün istifadə edirlər. Əgər **s1** sətiri **s2** sətiri ilə eynidirsə onda funksiya 0 qiymətini qaytarır. Əgər **s1**-in elementlərinin sayı **s2**-dən azdırsa onda <0 əks halda >0 qiymətini qaytarır.

**strcat() funksiyası.**

```
char * strcat (char *s1, char *s2);
```

Bu funksiya **s1**-in sonuna **s2**-ni əlavə edir.

## Proqram nümunələri:

### Proqram 1:

İstifadəçinin daxil etdiyi sətirin uzunluğunu ekranda çap edən proqram tərtib edək.

```
// prg_7_2.cpp
#include <iostream>
#include <string.h>

int main()
{

char set[1024];
int k;

std::cout<<"Zehmet olamsa her hansı setir daxil edin \n";
std::cin>>set;
k = strlen(set);
std::cout<<"Sizin daxil etdiyiniz setrin simvollarının sayı "
<<k<<" \n";
return 0;
}
```

Proqramı icra edək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Zehmet olamsa her hansı setir daxil edin
Olimpiada
Sizin daxil etdiyiniz setrin simvollarının sayı 9
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### Proqram 2.

Elə proqram qurun ki, istifadəçinin daxil etdiyi sətirin 10-cu simvolundan başlayaraq yerdə qalan hissəsini çap etsin. Əgər sətirin uzunluğu 10-dan kiçikdirsə onda proqram ekranda bütöv sətiri çap etsin.

```
// prg_7_3.cpp
#include <stdio.h>
#include <string.h>

int main(){
char set[1024];
int k;
std::cout<<"Zehmet olamsa her hansı setir daxil edin \n";
std::cin>>set;
k = strlen(set);

if (k<10)
std::cout<<set<<"\n";
else
{
char *p = set;
p+=10;
std::cout<<p<<"\n";
}
return 0;
}
```

Proqramı icra edək:

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>./prog2.exe
```

```
Zehmet olamsa her hansı setir daxil edin
```

```
SerguzeshiVeziriXaniLenkeran
```

```
iVeziriXaniLenkeran
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

### **Çalışmalar:**

1. Elə proqram qurun ki, istifadəçinin daxil etdiyi sətirin 5-ci simvolu ilə 15-ci simvolu arasında qalan hissəsini çap etsin. Əgər sətirin uzunluğu 20-dən kiçik olarsa onda ekranda bu barədə məlumat çap etsin.
2. Elə proqram tərtib edin ki, istifadəçidən 6 sətir qəbul etsin və bu sətirləri ardıcıl birləşdirərək tam sətir kimi çap etsin.
- 3.\* Elə proqram tərtib edin ki, istifadəçidən 5 sətir qəbul etsin və bu sətirləri daxil olma sırasının əksi ardıcılığında birləşdirərək tam sətir kimi çap etsin.
- 4.\* Elə proqram tərtib edin ki, istifadəçidən 5 sətir qəbul etsin və bu sətirləri uzunluqlarının artma ardıcılığı ilə alt-alta çap etsin.

## \$8 Struct tiplər.

İndiyə qədər biz dəyişən elan edərkən **int**, **double**, **char**, **long** kimi standart tiplərdən istifadə edirdik. Əlbətdə bu tiplər çox əlverişlidir, lakin çox vaxt məsələnin şərtinə uyğun olaraq proqramçının özü yaratdığı tiplərdən istifadə etmək lazım gəlir. Bu zaman strukt tiplərdən və ya siniflərdən istifadə olunur. Strukt tipi bizə bir neçə müxtəlif tiptən olan dəyişəni bir ad altında birləşdirməyə imkan verir.

Yeni struct tipi təyin etmək üçün sintaksis aşağıdakı kimidir.

```
struct yeni_tip {
    tip1 dey1;
    tip2 dey2;
    tip3 dey3;
    .
    .
    tipk deyn; };
```

Bu zaman biz **yeni\_tip** adlı tip yaratmış oluruq, hansı ki, özündə tip1, tip2 v.s. tiplərdən olan dəyişənləri saxlayır. Bu elandan sonra artıq biz standart tiplərdə olduğu kimi dəyişən elan edə bilərik.

Aşağıdakı kimi:

```
yeni_tip st1;
```

Bu elanda biz yeni yaratmış olduğumuz yeni\_tip strukt tipindən st1 adlı dəyişən elan etdik. Sadə nümunəyə baxaq. Özündə int və char tipli dəyişənlər saxlayan str adlı yeni strukt tipi yaradaq. Daha sonra bu tiptən dey adlı dəyişən elan edək.

```
struct str{
    int x;
    char c;
};
```

```
str dey;
```

Strukt tiplərinin tətbiqinə aid aşağıdakı kimi məsələyə baxaq. Tutaq ki hər-hansı zavodun 100 000 işçisi var. Bizdən tələb olunur ki, bu zavodun işçilərinin məlumatlar bazası proqramını yazaq. Hər bir işçi haqqında onun adı, soyadı, yaşı, maaşı, vəzifəsi barədə məlumatlar qeyd edilməlidir. Bunun üçün yeni **struct tipi** təyin edək.

Məsələmizə uyğun təyin etməli olduğumuz yeni tip belə olar. Gəlin bu yeni yaradacağımız tipə **ishci** adını verək.

```
struct ishci {
    int yash;
    char ad[12];
    char soyad[20];
    char vezife[20];
    double maash; };
```

Beləliklə biz yeni ishci tipi təyin etdik.

Bu elandan sonra biz proqramımızda bu tiptən adi tiplər kimi dəyişənlər və ünvan dəyişənləri elan edə bilərik. Məsələn: **ishci dey1, ishci1, \*muhendis, yeni\_ishciler[100];**

Yuxarıdakı elanda biz **ishci** tipindən olan **dey1** və **ishci1** dəyişənləri, **muhendis** ünvan dəyişəni və 100 elementli **yeni\_ishciler** cərgəsi yatardıq. 100 000 ishci barəsində məlumat saxlamaq üçün biz yeni yaratdığımız ishci tipindən olan 100 000 elementli cərgədən istifadə edə bilərik. **Ishci ishçiler[100000];**

İndi isə maraqlı məqam. Tutaq ki, **int** tipindən olan **x** dəyişənimiz var. **int x;** Əgər biz bu

dəyişənə 4 qiyməti mənimsətmək istəyiriksə  $x=4$ ; yazırıq. Bəs strukt tipindən olan dəyişənlərə və ya onların təşkil onlunduqları ayrı-ayrı elementlərə necə qiymət mənimsədək? **strukt** tipinin elementlərinə müraciət etmək üçün ( . ) və ya ( -> ) operatorlarından istifadə olunur.

Aşağıdakı kimi:

```
ishci reis;
reis.yash = 50;
strcpy(reis.ad,"Anar");
```

Yuxarıdakı kod hissəsində biz **ishci** tipindən olan **reis** dəyişəni elan elədik və onun **yash** həddinə **50** , ad həddinə isə "Anar" qiymətlərini mənimsətdik. Artıq strukt tipindən istifadə etməklə program tərtibinin vaxtı çatıb.

Program nümunəsi:

```
//prog_8_1.c
#include <iostream>
#include <string.h>

struct str{
int x;
char soz[30];
};

int main(int argc, char *argv[])
{

str str_dey;
str_dey.x=50;

strcpy(str_dey.soz, "Ali");

std::cout<<"str_dey -in heddleri \nx - "<<str_dey.x
<<"\nsoz - "<<str_dey.soz<<"\n";
return 0;
}
```

Programı yerinə yetirək

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
str_dey -in heddleri
x - 50
soz - Ali
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## 8.1 Strukt tipindən olan ünvan dəyişənləri

**strukt** tipindən aşağıdakı kimi ünvan dəyişəni təyin edək:

```
ishci *muhendis;
```

Əgər biz **strukt** tipdən adi yox ünvan dəyişəni təyin ediriksə onda **strukt** tipinin elementlərinə müraciət zamanı ( . ) əvəzinə (->) istifadə edirik.

Misal üçün:

```
muhendis->yash=45; kimi.
```

Paraqraf 4-dən bilirik ki, hər hansı bir tipdən olan ünvan dəyişəninə yaddaşda yer ayırmaq üçün

```
tip *dey; dey = new ischi;
```

sintaksisindən istifadə edirik.

Bu sintaksisə əsasən yuxarıda təyin etdiyimiz **muhendis** ünvan dəyişəninin elementlərinə qiymət mənimsədə bilmək üçün əvvəlcə ona yaddaşda aşağıdakı kimi yer ayırmalıyıq.

```
muhendis = new ishci;
```

Bu operatorndan sonra biz artıq muhendis dəyişəninin istənilən elementinə -> operatoru vasitəsilə müraciət edə bilərik.

```
muhendis->yash=33;
```

```
strcpy(muhendis->ad,"Rustem");
```

Proqram nümunəsi:

```
//prog_8_2.c
```

```
#include <iostream>
```

```
#include <string.h>
```

```
struct yeni_tip
```

```
{
```

```
char ad[20];
```

```
int x;
```

```
};
```

```
int main(int argc, char *argv[])
```

```
{
```

```
yeni_tip *dey;
```

```
dey = new yeni_tip;
```

```
dey->x=50;
```

```
strcpy(dey->ad, "Veli");
```

```
std::cout<<dey->ad<<" " <<dey->x<<"\n";
```

```
return 0;
```

```
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>./prog2.exe
```

```
Veli 50
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

### Çalışmalar:

1. Aşağıdakı işləri görən proqram tərtib edib icra edin.

int tipli x və 30 simvollarla sətir tipli söz dəyişənlərindən ibarət olan str adlı yeni struct tipi yaradın. Bu yeni yaratdığınız tiptən str\_dey adlı dəyişən elan edin. Bu dəyişənin x və söz üzvlərinə müvafiq olaraq 10 və "proqramlashdirma" sözlərini mənimsədin. str\_dey dəyişəninin üzvlərinin qiymətlərini ekranda çap edin.

2. Yuxarıdakı məsələdə str tipindən ünvan tipli str\_gst dəyişəni elan edin və məsələnin tələblərin yerinə yetirin.

3. 1-ci çalışmada daxil olunan str tipli 5 elementdən ibarət strler cərgəsi elan edin. Bu cərgənin hər bir elementinin üzvlərinə istifadəçi tərəfindən daxil olunan qiymətlər mənimsədin. Daha sonra bu qiymətləri ekranda çap edin.

4. Çalışma 3-ü funksiyalardan istifadə etməklə həll edin. Bu məqsədlə 2 funksiya tərtib edin, daxil\_et və cap\_et. Müvafiq olaraq daxil\_et funksiyası istifadəçidən məlumatları oxuyub, strler cərgəsinin elementlərinə mənimsədəcək, cap\_et isə strler cərgəsinin elementlərinin

qiymətlərini çap edəcək.

5. Funksiyalardan istifadə etməklə elə program qurun ki, çalışma 1-də daxil olunan str tipli 5 elementdən ibarət strler cərgəsi elan etsin. daxil\_et funksiyası vastəsilə istifadəçidən oxunan qiymətləri bu cərgəsinin elementlərinə mənimsətsin. Daha sonra max\_el funksiyası tərtib edin, hansı ki, strler cərgəsinin elementləri arasında x-i ən böyük olanın qiymətlerini ( x və soz ) çap etsin.

6. Çalışma 5-deki max\_el funksiyasını ele deyişin ki, strler cərgəsinin elementləri arasında soz həddlərinin ən uzunun qiymətlərini ( x və soz ) çap etsin.



## \$9 Siyahılar

Bu mövzuda biz C++ dilində yazılmış proqramlarda çox geniş istifadə olunan yeni tiplərlə - siyahılarla tanış olacağıq. Siyahıların C++ dilində tətbiqi olduqca genişdir və siyahısız C++ dilində yazılmış proqramları təsəvvür etmək mümkün deyil. Praktiki cəhətdən siyahılar cərgələrə oxşardır, onlar da cərəglər kimi özündə verilmiş tiptən olan elementlər ardıcılığını saxlayır. Lakin siyahıların cərgələrdən fundamental üstün cəhətləri oldur ki, cərgə elan edən zaman biz onun elementlərinin sayını əvvəlcədən elan etməliyiksə və sonra biz cərgəyə əlavə element yerləşdirə və ya onun elementlərinin sayının dəyişdirə bilməyiksə, siyahının elementləri ilə istədiyimiz kimi manipulyasiya edə bilərik. Yəni proqramın icrası boyu biz siyahıya istənilən sayda yeni element əlavə edə və ya mövcud elementləri siyahıdan silə bilərik.

Misal üçün:

Kompüterdə hal-hazırda icra olunan proqramlara nəzarət etmək üçün nüvə `task_struct` adlı siyahıdan istifadə edir. İstifadəçi yeni proqramlar yüklədikcə, nüvə `task` -lar siyahısına yeni element əlavə edir və bu elementdə yeni proqram barədə müvafiq məlumatları (yükləndiyi yerin ünvanı, adı, təşkil olunduğu hissələr, açdığı fayllar, v.s.) yerləşdirir. Daha sonra hər hansı proqram sona çatdıqda nüvə müvafiq elementi `task` -lar siyahısından silir. Əgər bu zaman siyahı əvəzinə cərgədən istifadə olunsaydı onda biz kompterdə müəyyən saydan artıq proqram yükləyə bilməzdik.

Cərgənin elementlərinin sayın böyük götürdükdə isə , lazım olunmayan elementlər boş-boşuna yaddaşda yer tutar.

### 9.1 Siyahının elan edilməsi

`int` tipindən biz adi dəyişən, ünvan dəyişəni və cərgə elan etmək qaydalarını bilirik.

```
int x, *y, z[10];
```

Yuxarıdakı misalda biz `int` tipindən `x` dəyişəni, `y` ünvan dəyişəni və 10 elementdən ibarət `z` cərgəsi elan etdik. İndi isə hər bir elementində `int` tipli dəyişən olan siyahı elan edək. Bunun üçün əvvəlcə siyahını təşkil edən obyektlərin tipini yaratmalıyıq. Yəni, qeyd elədik ki, siyahı da cərgə kimi elementlər(obyektlər) ardıcılığıdır və bu elementlərin(obyektlərin) hər birində müxtəlif məlumatlar yerləşdirmək olar. İndi biz həmin 1 obyekt yaratmağa çalışırıq, daha sonra bir neçə bu cür obyekt bir-biri ilə əlaqələndirib siyahı yaratma qaydasını örgənəcəyik. Baxdığımız sadə halda yaratmaq istədiyimiz obyekt özündə bir məlumat - `int` tipindən olan dəyişən saxlayır. Bu obyekt yaratmaq üçün biz `struct` tiptən istifadə edəcəyik.

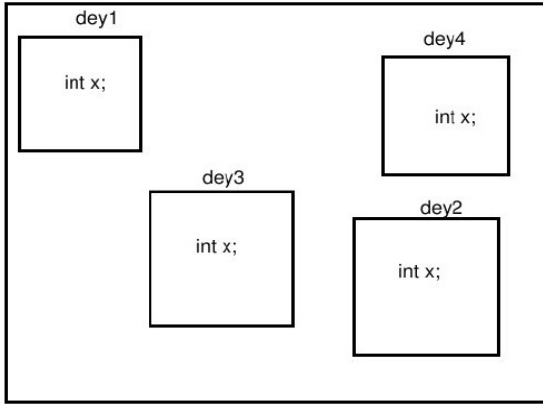
Aşağıdakı kimi:

```
struct syh_el {  
int x;  
}
```

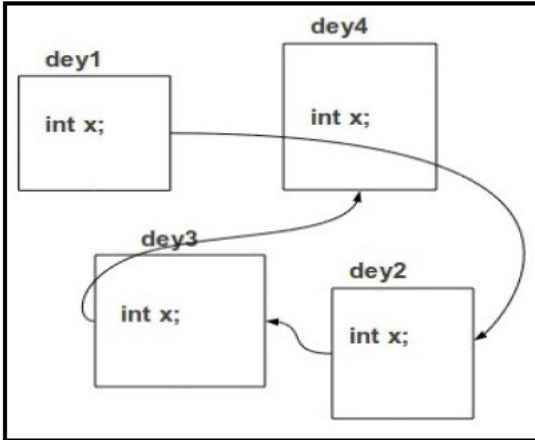
Yuxarıda biz özündə `int` tipindən olan dəyişən, `x` -i saxlayan yeni `struct` tipi yaratdıq. İndi bu tiptən olan bir neçə dəyişən elan edək.

```
syh_el dey1, dey2, dey3, dey4;
```

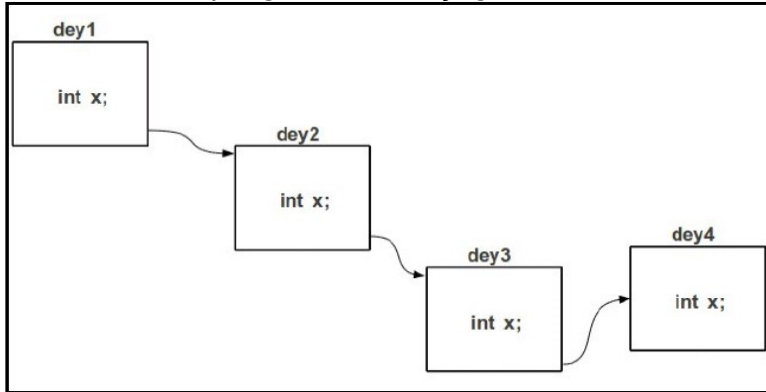
Hal-hazırda yaddaşda vəziyyət aşağıdakı kimidir:



Biz isə siyahı almaq üçün bu obyektləri bir-biri ilə əlaqələndirməliyik, aşağıdakı kimi:



Bir az daha səliqəli göstərsək, aşağıdakı kimi:



### Siyahı yaratma qaydası:

Əsas məsələ bir obyektə başqa obyekt ilə əlaqələndirməkdir. Əgər yaratmış olduğumuz obyektlərdən birini, digəri ilə əlaqələndirə bilsək, daha sonra yeni obyektə digər başqa biri ilə əlaqələndirə bilərik. Beləliklə də siyahıya istənilən sayda yeni obyekt əlavə edib siyahımızı istədiyimiz qədər uzada bilərik.

Bəs bir obyektə digəri ilə necə əlaqələndirək?

Bunun üçün proqramlaşdırmada aşağıdakı ilk baxışda elə də anlaşılmayan qaydadan istifadə edirlər. Siyahıda hər bir obyekt özündən sonra gələn obyektin yaddaşdakı ünvanını bilməlidir. Bunun üçün obyektin daxilində onun öz tipindən olan ünvan dəyişənini yerləşdirirlər və bu ünvan dəyişənini siyahının növbəti obyektinin ünvanını mənimsədirlər.

Aşağıdakı elana nəzər salaq:

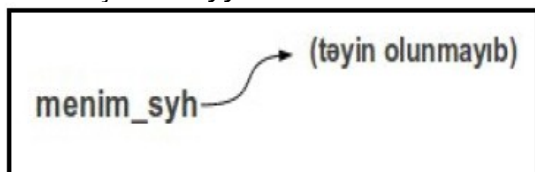
```
struct syh_el {  
int x;  
syh_el *novb_el;  
}
```

Bu elanın yuxarıdakı elandan yeganə fərqi o oldu ki, biz burada **struct syh\_el** tipinin daxilində bu tipdən olan **\*novb\_el** ünvan dəyişəni yerləşdirdik. Artıq bu bizə siyahı yaratmağa imkan verir. Əvvəlcə biz siyahımızı elan etməliyik.

Aşağıdakı kimi:

```
syh_el *menim_syh;
```

Yaddaşın vəziyyəti:

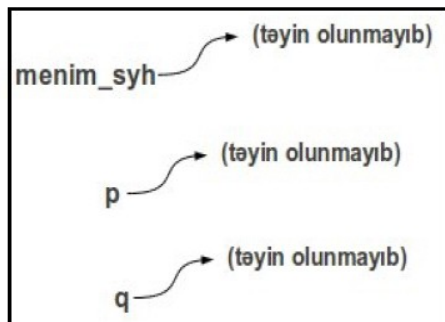


Siyahı yaratmaq üçün **syh\_el** tipindən olan daha iki ünvan dəyişəninə ehtiyacımız olacaq. Bunları `p` və `q` ilə işarə edək. Bunlardan biri - `p`, yeni obyektlərin yaradılması və inisializasiyası (ilkın qiymətin mənimsədilməsi), digəri - `q` isə iterasiya üçündür (siyahı boyu hərəkət etmək).

Gəlin bu dəyişənləri də elan edək:

```
syh_el *p,*q;
```

Yeri gəlmişkən yaddaşa da bir ötürü nəzər salaq:

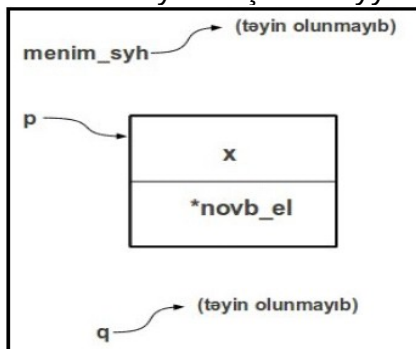


Hər şey hazırdır, siyahının ilk elementini yarada bilərik.

Bunun üçün yazırıq:

```
p = new syh_el;
```

Bu zaman yaddaşın vəziyyəti aşağıdakı kimi olar:

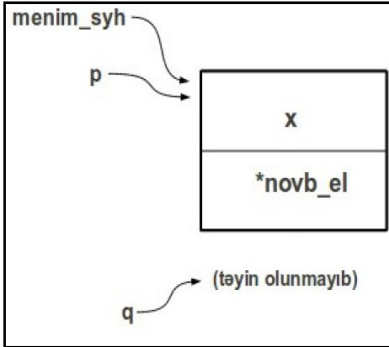


Növbəti addım `menim_syh` dəyişəninini bu yeni yaratdığımız obyektə mənimsətməkdir.

Bunun üçün sadəcə yazırıq:

```
menim_syh = p;
```

Yaddaşın vəziyyəti aşağıdakı kimidir:



Artıq siyahıya element əlavə etmək üçün **menim\_syh** dəyişənindən istifadə etməyəcəyik. O siyahının ilk elementinə istinad edir və siyahıya müraciət etmək üçün bu dəyişəndən istifadə edəcəyik. Hələlik isə siyahının yaradılmasını davam etdirək.

İkinci elementi yaratmaq üçün hazırlıq işləri görək.

Bunun üçün iterasiya dəyişənini siyahının ilk elementinə ( **p** -yə ) , ilk elementin ikinci elementlə əlaqələndirmə həddini( **novb\_el** ) isə **NULL** qiymətinə mənimsətməliyəm.

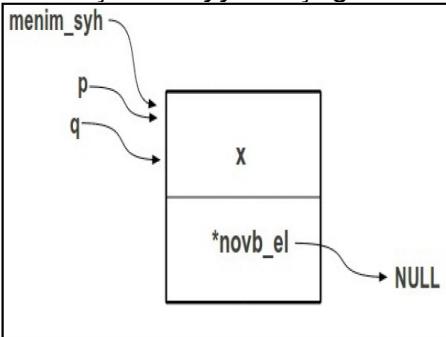
**NULL** adətən ünvan dəyişənlərinin yaddaşda heç bir yerə ümvanlanmadığını bildirmək üçün istifadə olunur.

Bu bizə siyahının sonun müəyyənləşdirmədə lazım olacaq.

```
q = p;
```

```
p->novb_el = NULL;
```

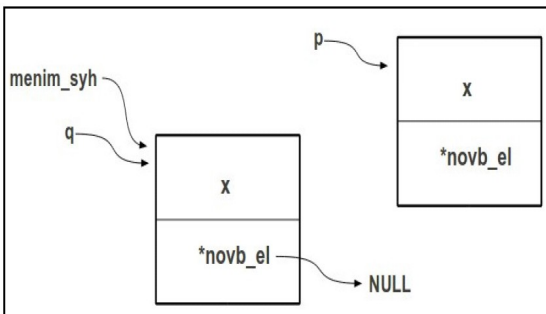
Yaddaşın vəziyyəti aşağıdakı kimidir:



Siyahının ikinci elementini yaradaq, eyni ilə birinci elementi yaratdığımız kimi:

```
p = new syh_el;
```

Yaddaşın vəziyyətinə baxaq:

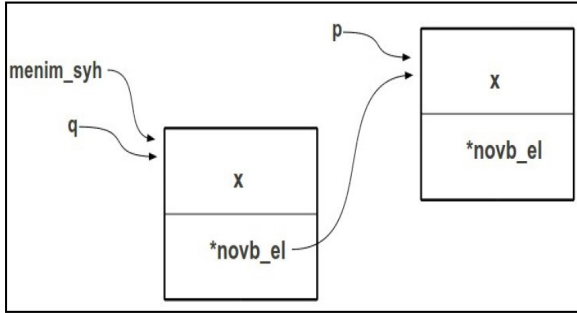


Hey diqqət! Əsas anlardan biri. İlk element ilə yeni yaratdığımız elementi birləşdiririk.

Bunun üçün sadəcə yazırıq:

```
q->novb_el = p;
```

Yaddaşa nəzər salaq:



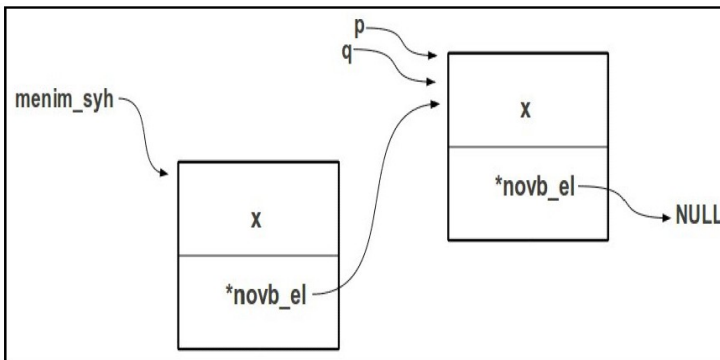
Artıq əlaqəmiz var.

Növbəti 3-cü elementi yaratmaq üçün hazırlıq işləri görək(bayaq bunu eləmişdik).

`q = p;`

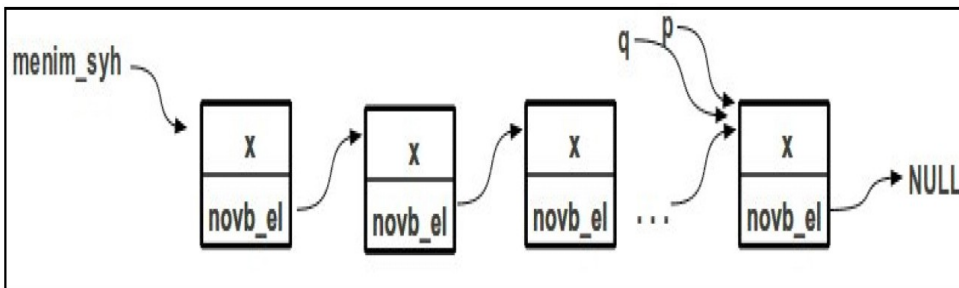
`p->novb_el = NULL;`

Yaddaşa baxırıq:



Artıq bu şəkildə davam edərək siyahıya istədiyimiz qədər yeni element əlavə edə bilərik.

Tutaq ki, siyahıya müəyyən qədər element əlavə etmişik və siyahımız aşağıdakı şəkildədir:

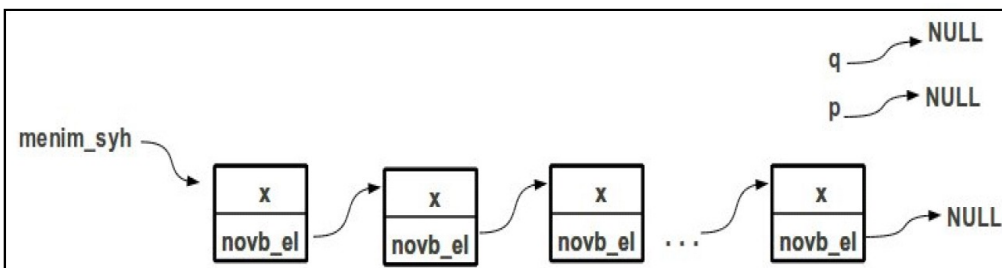


Ən sonda `p` və `q` dəyişənlərinin siyahıya istinadlarını ləğv etməliyik.

`p=NULL;`

`q=NULL;`

Bununla da siyahı yaratma prosesini tamamlamış olduq.



Çətin görünə bilər, amma siyahılar proqramlaşdırmada əvəzənilməzdir və çox praktikdirlər. Aşağıdakı nümunə proqramları yerinə yetirdikdən sonra siyahılarla işləmə bacarığımız bir qədər də artmış olar.

### Proqram nümunələri:

Gəlin yuxarıda daxil etdiyimiz, nümunəyə uyğun proqram hazırlayaq.

Yəni sadə bir proqram tərtib edək, bu proqramda özündə ancaq bir hədd , `int x`; saxlayan obyektərdən ibarət siyahı tərtib edək, bu siyahının obyektlərinin həddlərinə ( `x` ) qiymətlər mənimsədək, daha sonra siyahının elementlərini çap edək.

Əlbəttə hələlik biz hər şeyi əlimizlə edəcəyik, bir az sonra isə prosesi avtomatlaşdırmaq məqsədilə funksiyalardan istifadə edəcəyik.

### Proqram 1.

```
// prg_9_1.cpp
#include <iostream>
#include <string.h>

struct syh_el
{
    int x;
    syh_el *novb_el;
};

int main(int argc, char *argv[])
{
    // istifade edəcəyimiz deyishenleri ilan edirik
    syh_el *menim_syh, *p, *q;
    int dey;

    // ilk elementi yaradaq
    p = new syh_el;

    // lazimi menimsetmeleri aparaq
    benim_syh = p;
    q = p;
    p->novb_el = NULL;
    // her shey hazirdir ikinci elementi yaradiriq
    p = new syh_el;

    // siyahi ile ikinci elementin elaqesini qururuq
    q->novb_el = p;

    // lazim menimsetmeleri edirik
    q = p;
    p->novb_el = NULL;

    // artiq siyahida iki obyekt var, 3-nu yaradaq
    // 3-cu elementi yaradiriq
    p = new syh_el;

    // siyahinin sonu ile ucuncu elementin elaqesini qururuq
    q->novb_el = p;

    // lazimi menimsetmeleri edirik
```

```

q = p;
p->novb_el = NULL;

// siyahida hal-hazirda 3 element var, helelik besdir.
// p ve q -nu siyahidan ayiraq
p=NULL;
q=NULL;

/* vessalam indi benim_syh deyisheni yeni yaratdigimiz siyahinin ilk
elementine istinad edir ve onun vastesile siyahini butun
obyektlere muraciet ede bilerem*/

/* Siyahinin elementlerine qiymetler menimsedek, daha sonra bu
qiymetleri cap edeceyik*/

// Yene p ye ehtiyacimiz olacaq
p = benim_syh;

// indi p dayanib siyahinin evvelinde , ashagidaki koda diqqet edin
std::cout<<"Siyahinin heddlerinin qiymetlerini daxil edin.\n";
std::cout<<"Siyahinin birinci heddinin qiymetini daxil edin. \n";
std::cin>>dey;

/*Siyahinin ilk obyektinin x heddine istifadecinin daxil etdiyi
qiymeti menimsedirem */
p->x = dey;

/* Siyahinin ikinci obyektini uzerine surushmek ucun ashagidaki
qaydadan istifade olunur */
p = p->novb_el;

/* Artiq p siyahinin ikinci obyektine istinad edir ,
onun x heddine qiymet menimsedek*/
std::cout<<"Siyahinin ikinci heddinin qiymetini daxil edin. \n";
std::cin>>dey;
p->x = dey;
p = p->novb_el;

//Nehayet 3-cu obyekt
std::cout<<"Siyahinin ucuncu heddinin qiymetini daxil edin. \n";
std::cin>>dey;
p->x = dey;

/* p oz ishini bitirdi, siyahini ondan azad eliyirem.
Calishin siyahiniza lazim olmayan elave istinadlar saxlamayasiniz */
p = NULL;

//Siyahinin elementlerine qiymetler menimsedik, indi onlari cap edek
// p-ni siyahinin ilk ilk elementine menimsedek, bunu etmeyi bilirik
p = benim_syh;

// benim_syh -nin funksiyasi ancaq siyahinin bashlangic unvanini
//ozunde saxlamaqdir
std::cout<<"Siyahinin elementleri ashagidakilardir: \n\n";
dey = p->x;
std::cout<<"Birinci element " <<dey<<"\n";

//ikinci elementin uzerine susrushek

```

```

p = p->novb_el;
dey = p->x;
std::cout<<"İkinci element "<<dey<<"\n";

//Ucuncu elementin uzerine susrushek
p = p->novb_el;
dey = p->x;
std::cout<<"Ucuncu element "<<dey<<"\n";

std::cout<<"\nSiayhinin elementleri cap olundu \n";

return 0;
}

```

Programı yerinə yetirək:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Siayhinin heddlərinin qiymətlərini daxil edin.
Siayhini birinci heddinin qiymətini daxil edin.
65
Siayhini ikinci heddinin qiymətini daxil edin.
345
Siayhini ucuncu heddinin qiymətini daxil edin.
78
Siayhini elementləri aşağıdakılardır:

```

```

Birinci element 65
İkinci element 345
Ucuncu element 78

```

```

Siayhinin elementləri cap olundu
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

Bu nümunədə biz sadə siyahı yaratdıq 3 elementdən ibarət, siyahının həddlərinə qiymətlər mənimsətdik və bu qiymətləri çap etdik. Praktikada bu cür məsələlərin həllində funksiyadan istifadə etmək zəruridir. Gəlin yuxarıdakı programın funksiyaların tətbiqi ilə olan variantını nəzərdən keçirək. Funksiyalardan istifadə etməklə yuxarıda baxdığımız program aşağıdakı kimi olar: Bizə mahiyyət etibarilə 2 funksiya lazımdır: biri siyahını yaratmaq və ona elementləri yerləşdirmək, digəri isə siyahının elementlərini çap etmək. Bu funksiyaları uyğun olaraq **siyahı\_yarat** və **siyahini\_cap\_et** kimi adlandıraraq.

Birinci funksiya 2 parametrlə qəbul edir : **syh\_el** \* tipindən olan dəyişən - yaratmaq istədiyimiz siyahı və **int** tipli dəyişən - siyahıya daxil etmək istədiyimiz elementlərin sayı. İkinci funksiya isə bir parametrlə qəbul edir: çap etməli olduğumuz siyahıya istinad, **struct syh\_el** \* tipli.

Funksiyaların elanları aşağıdakı kimi olar:

```

syh_el * siyahı_yarat(syh_el *syh, int elem_say); void siyahini_cap_et( syh_el *);

```

Programı daxil edək:



## Program 2.

```
// prg_9_2.cpp
#include <iostream>
#include <string.h>

struct syh_el
{
    int x;
    syh_el *novb_el;
};

syh_el *siyahi_yarat(struct syh_el *syh, int elem_say);
void siyahini_cap_et(struct syh_el *);

int main(int argc, char *argv[])
{
    // istifade edeceyimiz deyishenleri ilan edirik
    struct syh_el *menim_syh;

    // siyahinin bosh oldugunu bildirmek ucun
    benim_syh = NULL;
    int say;

    std::cout<<"Siyahinin elementlerinin sayini daxil edin \n";
    std::cin>>say;

    benim_syh = siyahi_yarat(menim_syh,say);

    siyahini_cap_et(menim_syh);

    return 0;
}

//-----

syh_el *siyahi_yarat( syh_el *syh, int elem_say)
{
    syh_el *p, *q;
    p=syh;
    q=syh;
    int i,dey;

    for (i=1; i<=elem_say; ++i)
    {

        std::cout<<"siyahinin "<<i
        <<" -ci elementini daxil edin \n";
        std::cin>>dey;

        p = new syh_el;
        p->x = dey;
        p->novb_el = NULL;

        if (syh==NULL)
        {
```

```

//siyahi boshdur, ilk element
syh=p;
q = p;
p = NULL;
}
else
{
//siyahida element var
q->novb_el = p;
q = p;
}
}

return syh;
}

//-----

void siyahini_cap_et( syh_el *syh)
{
syh_el *p;
int dey;
p = syh;

if (syh == NULL )
{
std::cout<<"Siyahi boshdur \n";
return;
}
std::cout<<"Siyahinin elementleri \n";
while(p!=NULL)
{
dey = p->x;
std::cout<<dey<<" ";
p = p->novb_el; // novbeti elemente kec
}
std::cout<<"\n";
}

```

Programı yerinə yetirək:

```

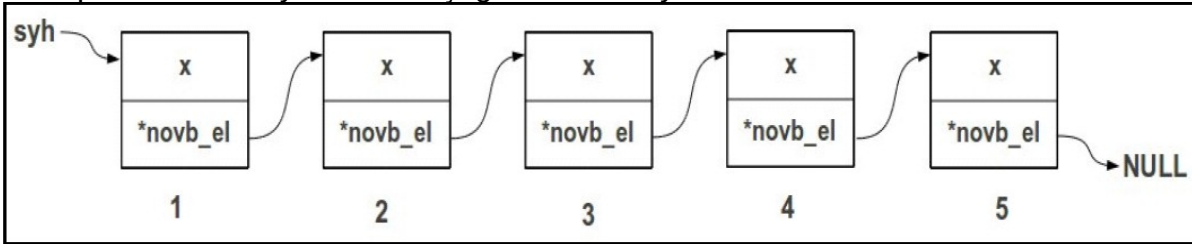
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Siyahinin elementlerinin sayini daxil edin
5
siyahinin 1-ci elementini daxil edin
23
siyahinin 2 -ci elementini daxil edin
45
siyahinin 3 -ci elementini daxil edin
567
siyahinin 4 -ci elementini daxil edin
1
siyahinin 5-ci elementini daxil edin
789
Siyahinin elementleri 23 45 567 1 789
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

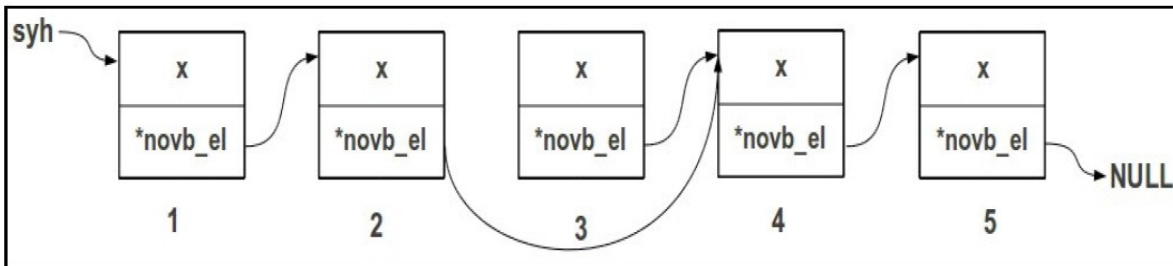
## 9.2 Siyahılardan elementlərin silinməsi.

Biz qeyd etdik ki, siyahılara proqramın icrası boyu istədiyimiz zaman istədiyimiz qədər yeni obyekt əlavə edə və siyahıda olan obyektləri silə bilərik. Gəlin indi də siyahıdan elementlərin silinməsi qaydasını göstərək.

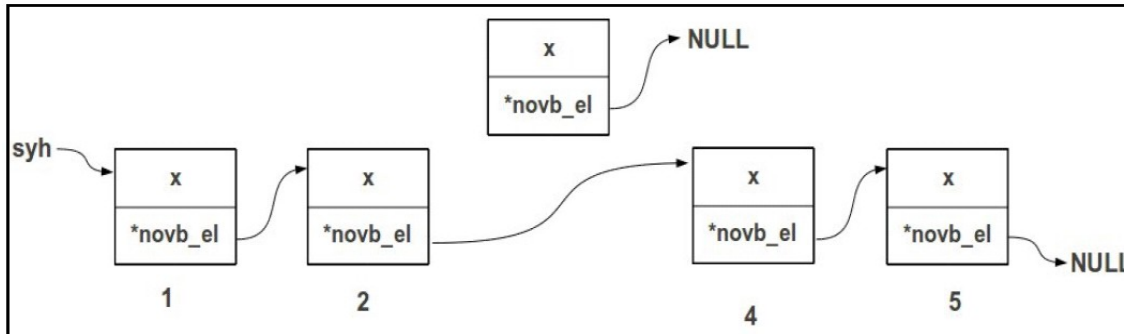
Tutaq ki, bizim 5 obyektli olan aşağıdakı kimi siyahımız var:



Əlbəttə siyahının aşağısında göstərdiyimiz nömrələrin siyahı ilə heç bir əlaqəsi yoxdur, sadəcə konkret obyektlərə müraciəti asanlaşdırmaq üçün göstərmişik. Biz bu siyahıdan üçüncü obyektə silmək istəyirik. Əvvəlcə qrafik şəkildə görəcəyimiz işi təsvir edək, daha sonra müvafiq proqram kodunu daxil edərək. Bunun üçün ilk olaraq 2-ci obyektin 3-cü obyektə olan istinadın 4-cü obyektə mənimsədirik:

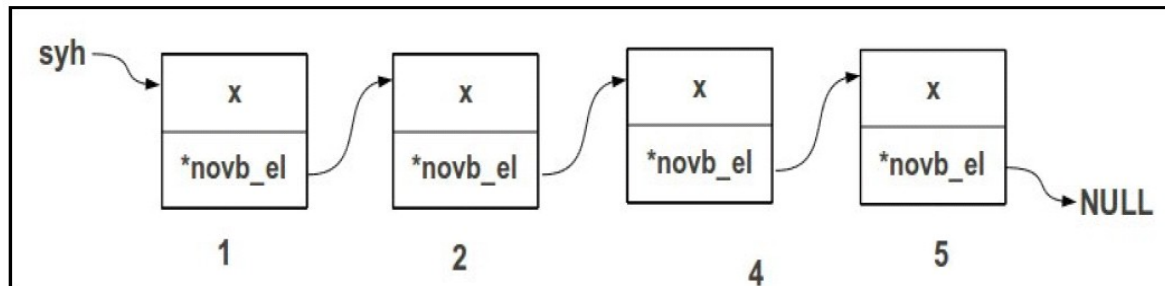


Daha sonra 3-cü obyektin 4-cü obyektə olan istinadın ləğv edirik.



Hal-hazırda biz qarşımıza qoyduğumuz məqsədə nail olmuşuq, artıq siyahıdan 3-cü obyekt kənar edilib və 2-ci obyekt birbaşa 4-cü obyektə birləşir.

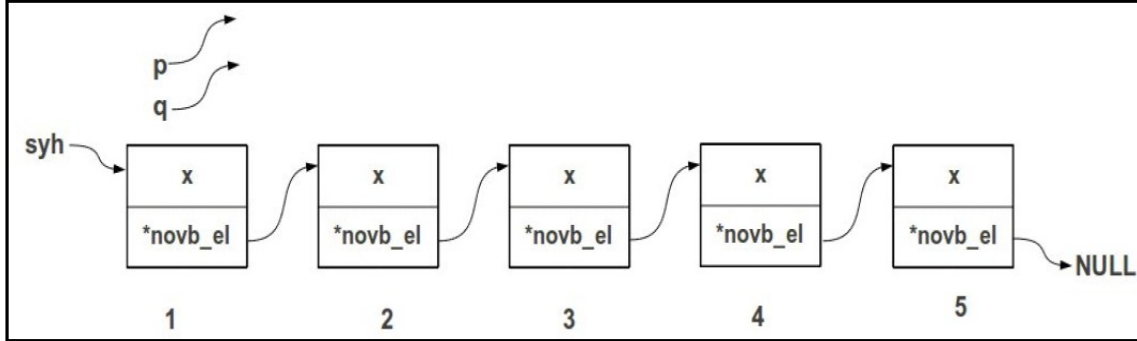
Amma 3-cü obyekt hələ-də yaddaşdadır və əgər o bizə lazım deyilsə biz onu yaddaşdan silməliyik.



Görəcəyimiz iş ilə tanış olduq, indi isə gəlin bu işi yerinə yetirən müvafiq program kodunu daxil edək. Sadəlik üçün siyahı tipi olaraq yuxarıda elan etdiyimiz **syh\_el** tipindən istifadə edək. Tutaq ki, **syh\_el \*** tipindən olan **syh** dəyişəni elan olunub (siyahının başlanğıcı) və siyahıya 5 element əlavə olunub.

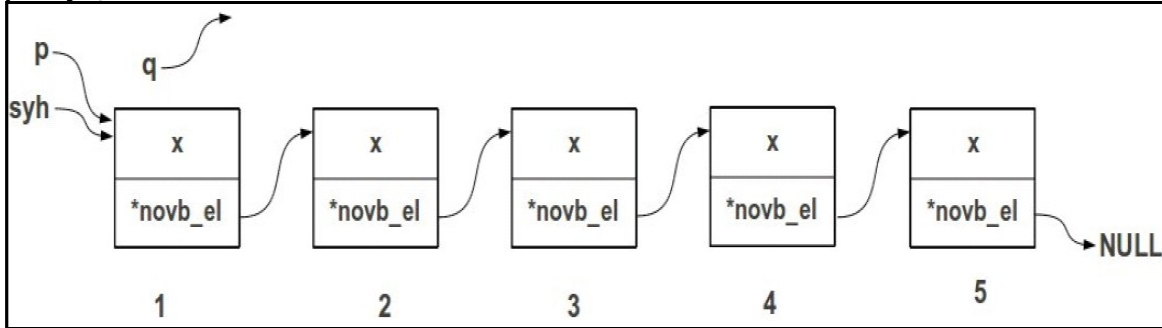
**syh\_el \*** tipindən olan **p** və **q** dəyişənlərini elan edək.

```
struct syh_el *p, *q;
```



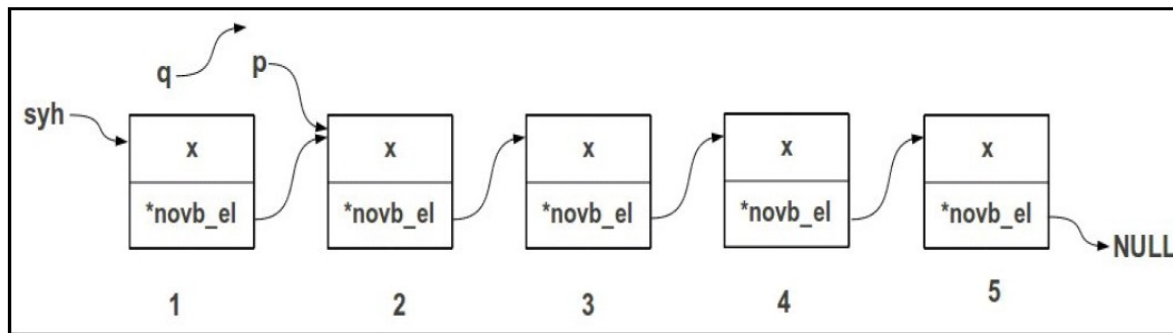
**p** -ni siyahının başlanğıcına mənimsədik.

```
p = syh;
```



**p** -ni siyahının ikinci obyektinə sürüşdürək.

```
p = p->novb_el;
```

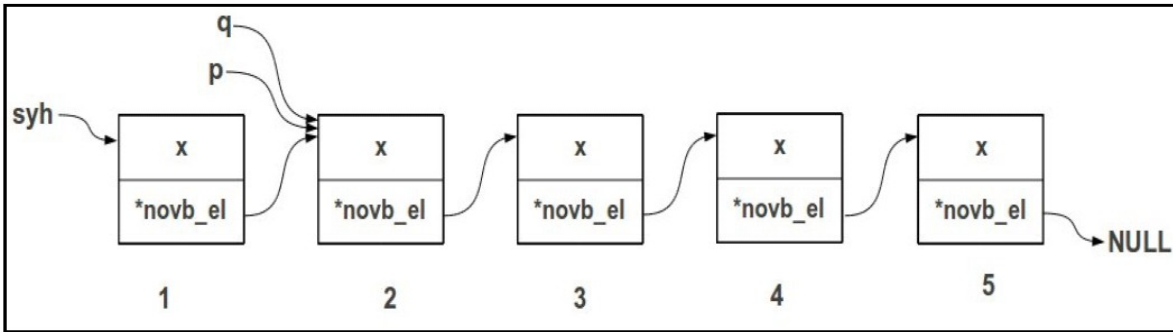


**q** -nü də ikinci obyektə mənimsətməliyik.

Bu bizə ikinci obyektin **novb\_el** həddinə müraciət etməyə imkan verəcək. Hal-hazırda **p** ikinci obyektə istinad etdiyindən daha siyahının əvvəldən başlamağa ehtiyac yoxdur.

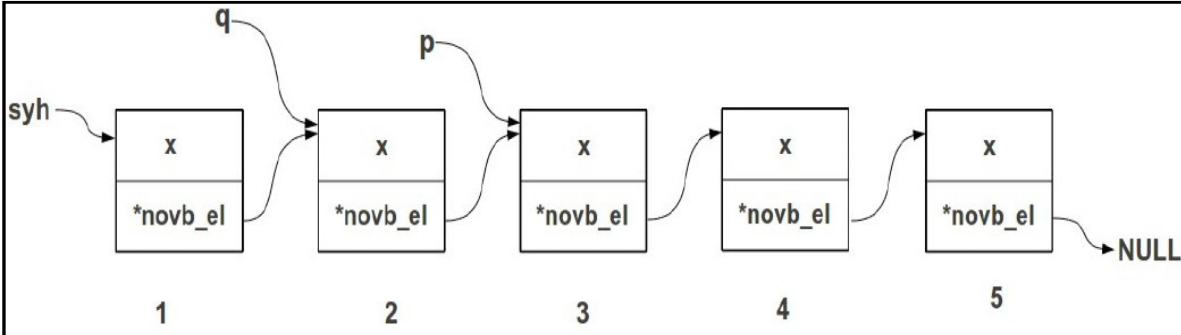
Sadəcə yazırıq:

```
q = p;
```



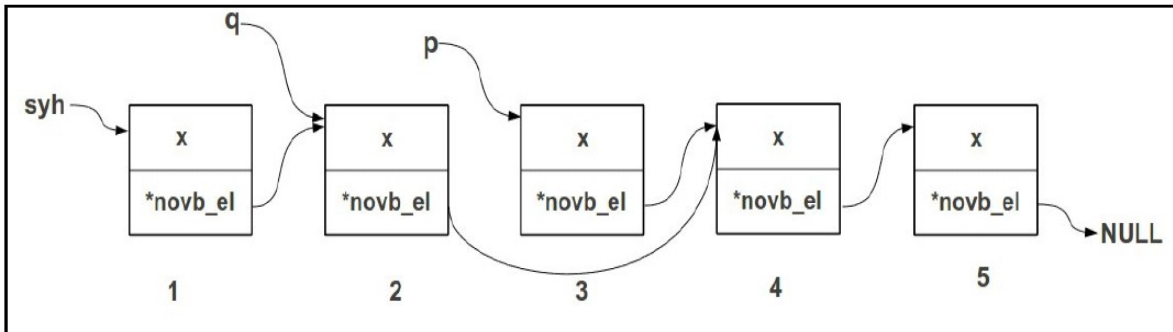
p -ni siyahının 3-cü obyektinin üzərinə sürüşdürək.

`p = p->novb_el;`



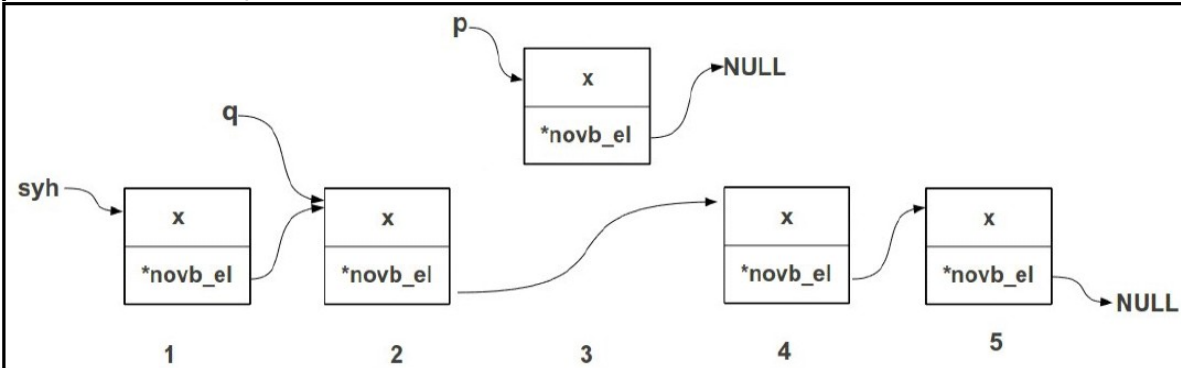
Siyahının ikinci obyektinin `novb_el` həddini 4 -cü obyektə mənimsədək:

`q->novb_el = p->novb_el;`



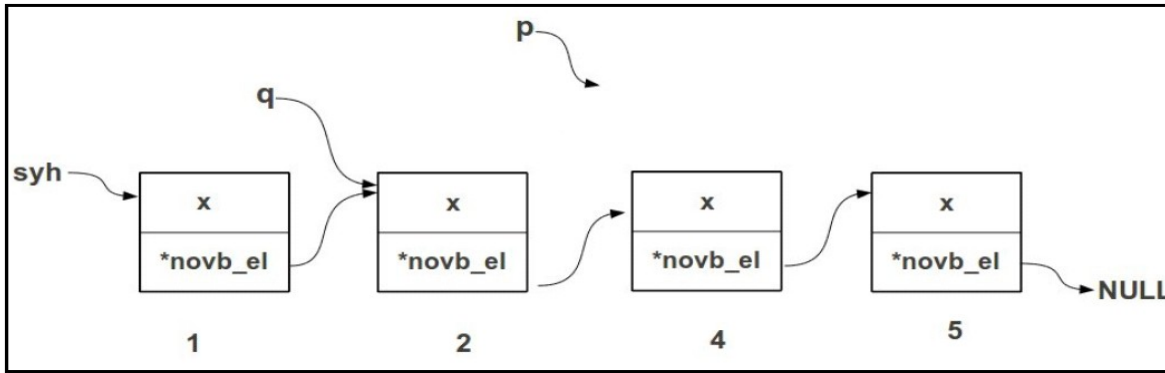
3-cü obyektin siyahı ilə əlaqəsini ləğv edirik :

`p->novb_el = NULL;`



Əgər bizə 3-cü obyekt artıq lazım deyilsə onu yaddaşdan silirik:

`delete p;`



İndi isə gəlin bütün bu dediklərimizi icra edən konkret program nümunəsi daxil edək.

### Program 3.

```
// prg_9_3.cpp
```

```
#include <iostream>
```

```
#include <string.h>
```

```
struct syh_el{
int x;
struct syh_el *novb_el;
};
```

```
syh_el *siyahi_yarat(struct syh_el *syh, int elem_say);
```

```
syh_el *siyahi_sil(struct syh_el *syh, int elem);
```

```
void siyahini_cap_et(struct syh_el *);
```

```
int main(int argc, char *argv[])
{
```

```
// istifade edeceyimiz deyishenleri elan edirik
syh_el *menim_syh;
```

```
// siyahinin bosh oldugunu bildirmek ucun
menim_syh = NULL;
int say,elem;
```

```
std::cout<<"Siyahinin elementlerinin sayini daxil edin \n";
std::cin>>say;
```

```
menim_syh = siyahi_yarat(menim_syh,say);
siyahini_cap_et(menim_syh);
```

```
std::cout<<"Siyahidan silmek istediyyiniz elementin indeksini daxil edin\n";
std::cin>>elem;
```

```
menim_syh = siyahi_sil(menim_syh, elem);
siyahini_cap_et(menim_syh);
```

```
return 0;
}
```

```
//-----
```

```
syh_el *siyahi_yarat( syh_el *syh, int elem_say)
{
syh_el *p, *q;
```

```

p=syh;
q=syh;
int i,dey;

for (i=1; i<=elem_say; ++i)
{
std::cout<<"siyahinin "<<i<<" -ci elementini daxil edin \n";
std::cin>>dey;
p = new syh_el;
p->x = dey;
p->novb_el = NULL;

if (syh==NULL){
//siyahi boshdur, ilk element
syh=p;
q = p;
p = NULL; }
else {
//siyahida element var
q->novb_el = p;
q = p;
}
}

return syh;
}

```

//-----

```

void siyahini_cap_et( syh_el *syh)
{
syh_el *p;
int dey;
p = syh;

if (syh == NULL ) {
std::cout<<"Siyahi boshdur \n";
return;
}

std::cout<<"Siyahinin elementleri \n";

while(p!=NULL){
dey = p->x;
std::cout<<dey<<" ";
p = p->novb_el; // novbeti elemente kec
}

std::cout<<"\n";
}

```

//-----

```

syh_el *siyahi_sil(syh_el *syh, int elem)
{
syh_el *p, *q;
p=syh;
int i,dey;

```

```

if (syh==NULL)
return NULL; // siyahi boshdur

if (elem==1){
// silmek istediyyimiz element ilk elementdir
syh = p->novb_el;
p->novb_el = NULL;
delete p;
return syh;
}

for (i=1; i<elem-1; ++i)
if (p==NULL) break;
else
p = p->novb_el;

if (p==NULL){
std::cout<<"Siyahida "<<elem<<" sayda element movcud deyil\n";
return syh;
}

q=p;
p = p->novb_el;

if (p->novb_el==NULL)
{
// silmek istediyyimiz element sonuncu elementdir
q->novb_el=NULL;
delete p;
p=NULL;
q=NULL;

return syh;
}

// silmek istediyyimiz element araliq elementdir
q->novb_el = p->novb_el;
p->novb_el = NULL;
delete p;
q=NULL;

return syh;
}

```



Nəticə:

```
C:\cpp\prog2\Debug>prog2.exe
Siyahinin elementlerinin sayini daxil edin
7
siyahinin 1 -ci elementini daxil edin
23
siyahinin 2 -ci elementini daxil edin
45
siyahinin 3 -ci elementini daxil edin
123
siyahinin 4 -ci elementini daxil edin
567
siyahinin 5 -ci elementini daxil edin
78
siyahinin 6 -ci elementini daxil edin
345
siyahinin 7 -ci elementini daxil edin
99
Siyahinin elementleri
23 45 123 567 78 345 99
Siyahidan silmek istediyyiniz elementin indeksini daxil edin
3
Siyahinin elementleri
23 45 567 78 345 99
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### Çalışmalar

1. Yuxarıda verilmiş proqram 2 nümunəsini elə dəyişin ki, siyahının obyektləri özündə int tipli  $x$  həddindən əlavə char tipli  $ad[20]$  həddi də saxlasın. Müvafiq olaraq siyahı yaratma və `cap_et` funksiyalarında lazımi dəyişikliyi edin. Proqramı kompilyasiya və icra edin, siyahı yaradin , onun elementlərini çap edin.
2. Yuxarıdakı proqrama axtarış funksiyası əlavə edin. Proqram istifadəçidən int tipli ədəd istəyir və siyahıda  $x$  -i bu ədədə bərabər olan obyektin  $ad[20]$  -həddini çap edir.
3. İki siyahını birləşdirən funksiya tərtib edin. Funksiya parametr olaraq iki siyahı qəbul edir, daha sonra bu siyahılardan birincini ikinci ilə birləşdirir.
- 4.Nümunə Proqram 3 -də daxil edilmiş silmə funksiyasının elə dəyişdirin ki, siyahının verilmiş indeksli obyektindən başlayaraq verilmiş sayda obyekt siyahıdan silsin.

## \$10 Siniflər.

Paraqraf \$8 -də biz `stuct` tiplər ilə tanış olduq. Qeyd elədik ki, `stuct` tiplər proqramçı tərəfindən yaradılan yeni tiplərdir və bu tiplərdən biz digər standart tiplərdən olduğu dəyişənlər elan edə bilərik. C++ dilinin adı çəkiləndə yada düşən ilk anlayış bizim yeni daxil edəcəyimiz `sinif` anlayışıdır. Burada qeyri-adi, yeni heç bir şey yoxdur. Sadəcə olaraq siniflərə `strukt` tiplərin biraz fərqli və inkişafetmiş növü kimi baxmaq olar. Belə ki, `strukt` tipinin elementlərini tərtib edərkən biz standart tiplərdən və əvvəl yaratdığımız `strukt` tiplərdən istifadə edirdik. Bu qayda siniflər üçün də keçərlidir. Lakin siniflərin `strukt` tiplərdən fundamental üstünlüyü ondadır ki, biz siniflərin tərkibinə nəinki hər hansısa tiptən olan dəyişən hətta funksiyaları da daxil edə bilərik. ( C dilində `sinif` təyin olunmur, amma `strukt` daxilində funksiya elan etməyə imkan var) Siniflərdən istifadə etməklə tərtib olunmuş sadə nümunə proqram fikirlərimizə daha da aydınlıq gətirər.

Proqram nümunəsi:

```
// prg_10_1.cpp
#include <iostream>
// sade_sinif tipini elan edirik

class sade_sinif{
public:
int x;
int y;
int cem(void);
};

// sade_sinif -in cem funksiyasının metn kodu
int sade_sinif::cem(void)
{
return x+y;
}

//-----
int main(){

//yeni tertib etdiyimiz sinif tipinden deyil adli deyishen elan edirik
// sinif tipinden olan deyishen numunelerine "obyekt" deyilir.
sade_sinif dey1;

//dey1 obyektinin heddlerine qiymetler menimsedirik
dey1.x = 5;
dey1.y = 10;

//dey1 obyektinin cem funksiyasına muraciet edirik
std::cout<<"dey1 obyektinin x ve y heddlerinin cemi = "
<<dey1.cem()<<"\n";

return 0;
}
```

### Proqramın izahı:

Gəlin yuxarıda daxil etdiyimiz proqramı analaiz edək, daha sonra siniflərlə bağlı digər proqram nümunələrinə baxarıq. Yeni `sinif` tipi yaradarkən `class` sözündən istifadə edirik. Daha sonra isə yeni yaratmaq istədiyimiz `sinif` tipinin adını yazırıq.

Aşağıdakı kimi:

```
class sinif_adi
```

Sinifi adi dəyişənləri adlandırdığımız kimi adlandıra bilərik. Daha sonra isə { mötərəzəsini yerləşdiririk. Nümunə proqramda bu **class sade\_sinif** sətirinə uyğun gəlir. { mötərəzəsindən sonra sinifi təşkil edən elementlər: dəyişənlər və funksiyalar yerləşdirilir. sinifin sonun bildirmək üçün } mötərəzəsindən və sintaksis tələbi olan ; -dən istifadə olunur.

Baxdığımız sinifdə biz **int** tipli **x** və **y** dəyişənləri, **int** tipli nəticə qaytaran **cem** funksiyası elan etmişik.

```
class sade_sinif{
```

```
public:
```

```
int x;
```

```
int y;
```

```
int cem(void);
```

```
};
```

**public** ifadəsinin mənasını irəlidə daxil edəcəyik.

Beləliklə biz artıq **sade\_sinif** adlı yeni **class** tipi elan etmişik. Yeni sinif tipi yaradarkən yerinə yetirməli olduğumuz ikinci iş sinifi funksiya həddlərinin mətn kodunu tərtib etməkdir. Sinifin funksiya həddlərinin mətn kodu adi funksiyaların mətn kodu kimi tərtib olunur, sadəcə funksiyanın hansı sinifə məxsus olduğunu bildirmək üçün funksiyanın adının əvvəlinə **sinif\_adi::** ifadəsi yazırıq.

Nümunə proqramında bu **int sade\_sinif::cem(void)**{ sətirinə uyğun gəlir. Daha sonra isə adi qaydada olduğu kimi funksiyanın mətn kodunu tərtib edirik. Bizim daxil etdiyimiz funksiyanın mətn kodu bir sətirdən ibarətdir **return x+y;** .

**x** və **y** dəyişənləri **cem** funksiyasına nə parametrlər kimi ötürülüb, nə də onun daxilində dəyişən kimi elan olunublar. Bəs **cem** funksiyası **x** və **y** dəyişənlərini necə tanıdı və onların qiymətlərini haradan götürdü?

Burada siniflərin aşağıdakı xassəsindən istifadə olunur:

Sinifin funksiya həddləri onun dəyişən həddlərinə birbaşa müraciət edə bilər . **cem** funksiyası daxilində istifadə olunan **x** və **y** dəyişənləri məhs sinifə məxsusdur. Sinifin elementlərini daxil etdikdən və funksiya həddlərinin mətn kodunu tərtib etdikdən sonra biz artıq yeni yaratdığımız sinif tipindən dəyişənlər elan edə bilərik, adi tiplərdən olduğu kimi.

Nümunə proqramın main funksiyasında biz **sade\_sinif** tipindən **dey1** adlı dəyişən elan edirik, aşağıdakı kimi:

```
sade_sinif dey1;
```

**Struct** tipində olduğu kimi **class** tipində də həddlərə müraciət etmək üçün nöqtə (.)

operatorundan istifadə olunur. Misal üçün kvadrat sinifinin tərəf həddinə müraciət etmək

üçün **kvadrat.tərəf** yazırıq. Proqramda biz **dey1** obyektinin **x** və **y** həddlərinə müvafiq olaraq 5 və 10 qiymətləri mənimsətmişik, aşağıdakı kimi:

```
dey1.x = 5;
```

```
dey1.y = 10;
```

Daha sonra isə **dey1** obyektinin **cem** funksiya həddinə müraciət olunur hansı ki, öz növbəsində **dey1** obyektinin **x** və **y** həddlərinin cəmini nəticə olaraq qaytarır. Bu baxdığımız sadə nümunədə biz yeni sinif tipi elan etməyi, onun funksiya həddlərinin mətn kodunu tərtib etməyi və eləcə də sinifin dəyişən və funksiya həddlərinə müraciət etməni örgəndik. Bu siniflər barəsində bilməli olduğumuz zəruri anlayışlar idi. İndi isə siniflərlə bağlı digər anlayışları və istifadə qaydalarını daxil edək.

## 10.1 açıq və gizli həddlər

Sınıf tipinin **strukt** tipindən digər fərqi də odur ki, **sınıf** tipinin həddləri açıq və gizli ola bilər. Sınıfın açıq həddlərinə proqramın istənilən funksiyasından müraciət etmək olar, gizli həddlərə isə yalnız və yalnız sınıfın öz funksiyaları müraciət edə bilər. Sınıfın hər-hansı həddini açıq elan etmək üçün **public**: ifadəsindən istifadə edirlər.

Bu zaman **sınıf** daxilində **public**: ifadəsindən sonra yerləşdirilmiş həddlər açıq olurlar. Sınıfın həddlərini gizli elan etmək üçün **private**: ifadəsindən istifadə olunur. Sınıfın elanı daxilində açıq və gizli həddlərin hansı ardıcılıqla yerləşdirilməsi ilə bağlı heç bir məhdudiyyət yoxdur. Gəlin daxil etdiyimiz bu yeni anlayışlara aid proqram nümunəsi ilə tanış olaq.

Proqram nümunəsi

```
// prg_10_2.cpp
#include <iostream>
// duzbucaqli tipini elan edirik

class duzbucaqli
{
public:
void terefler(int,int);
int sahe(void);

private:
int en;
int uzunluq;
};

// duzbucaqli -nin terefler funksiyasının metn kodu
//klasin en ve uzunluq gizli heddlerini qiymetlendiririk
void duzbucaqli::terefler(int x,int y)
{
en = x;
uzunluq = y;
}

// duzbucaqli -nin sahe funksiyasının metn kodu
int duzbucaqli::sahe(void)
{
return en*uzunluq;
}

//-----
int main()
{

duzbucaqli duzbl;

//duzbl -in en ve uzunluguna qiymetler menimsedek
duzbl.terefler(5,8);

//duzbl -in sahəsini cap edek
std::cout<<"duzbucaqlinin sahəsi = "<<duzbl.sahe()<<"\n";

return 0;
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
duzbucaqlinin sahəsi = 40
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## 10.2 Konstruktor və Destruktor

Yuxarıda baxdığımız sınıf2.cpp proqramında biz duzbucaqli sinfindən **duzb1** obyektini yaradırıq. Daha sonra bu obyektin **en** və **uzunluq** -nu mənimsətmək üçün **terefler** funksiya həddini çağırırıq.

Obyekt yönümlü proqramlaşdırmada bir çox hallarda obyektin həddlərinin qiymətlərini obyekt yaradılarkən mənimsətmək tələb olunur. Bunun üçün **kanstruktur** adlandırılan funksiyadan istifadə olunur, hansı ki, obyekt yaradılarkən avtomatik çağırılır. Obyekt yaradılarkən deyəndə verilmiş **sınıf** tipindən hər- hansı dəyişənin elan olunması nəzərdə tutulur. Misal üçün yuxarıdakı nümunə proqramda **duzbucaqli duzb1**; sətirində **duzb1** obyektini yaradılır və əgər biz duzbucaqli sinifi üçün **kanstruktur** təyin etsəydik onda o avtomatik çağırılardı.

## 10.3 Kanstrukturun təyin olunması

Hər hansı sinifin kanstrukturunu təyin etmək üçün bu sinif daxilində həmin sinifin adı ilə üst-üstə düşən funksiya təyin etmək lazımdır. Bu funksiyaya sinifin qurucusu - kanstruktur deyilir. Qurucu funksiyasının məqsədi sinifin həddlərinin ilkin qiymətlərini mənimsətməkdir. Gəlin yuxarıda daxil etdiyimiz sınıf2.cpp proqramının sinifdən istifadə etməklə variantını daxil edək.

Proqram nümunəsi

```
// prg_10_3.cpp
#include <iostream>

// duzbucaqli tipini elan edirik
class duzbucaqli
{
public:
    duzbucaqli();
    void terefler(int,int);
    int sahe(void);

private:
    int en;
    int uzunluq;
};

//qurucu - kanstruktur
duzbucaqli::duzbucaqli()
{
    en = 10;
    uzunluq = 15;
}

// duzbucaqli -nin terefler funksiyasının metn kodu
//sinifin en ve uzunluq gizli heddlerini qiymetlendiririk
void duzbucaqli::terefler(int x,int y)
{
    en = x;
```

```

uzunluq = y;
}

// duzbucaqli -nin sahe funksiyasinin metn kodu
int duzbucaqli::sahe(void)
{
return en*uzunluq;
}

//-----

int main(){

duzbucaqli duzbl;

//qurucu avtomatik cagrilir ve heddlere qiymetler menimsedir
//bunu sahe funksiyasini cagirmaqla yoxlaya bilerik
std::cout<<"duzbucaqlinin sahesi = "<<duzbl.sahe()<<"\n";

//duzbl -in tereflerine ayri qiymetler menimsedek
duzbl.terefler(5,8);

std::cout<<"duzbucaqlinin sahesi = "<<duzbl.sahe()<<"\n";

return 0;
}

```

Nəticə:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
duzbucaqlinin sahesi = 150
duzbucaqlinin sahesi = 40
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## 10.4 Destruktorun təyin olunması

Destruktor funksiyası konstrukturun əksinə olaraq obyekt yaddaşdan silindiği zaman çağırılır. Bu funksiya da sinifin adı ilə eyni adlandırılır, yalnız adın əvvəlinə ~ işarəsi artırılır. Destruktorlar əsasən dinamik yaradılan obyektlər (ünvan tipli sinif dəyişənləri) yaddaşdan silinərkən avtomatik çağırılır, lakin əgər biz obyekti özümüz istədiyimiz vaxt məhv etmək istəsək onda onun destrukturu çağırırıq.

## 10.5 Obyektlərin dinamik yaradılması və silinməsi

Növbə çatdı ünvan dəyişənləri barəsində örgəndiyimiz biliklərin siniflərə tətbiqinə. \$4 -də qeyd eləmişdik ki, statik dəyişənlər elan olunan zaman onlara yaddaşda yer ayrılır və bu yeri sonradan həmin dəyişəndən azad edib başqa məqsədlər üçün istifadə etmək mümkün deyil. Bu səbəbdən də iri həcmli dəyişənlərdən istifadə olunan zaman adi dəyişənlərdən istifadə etmək proqramın məhsuldarlığın aşağı salır, dinamik dəyişənlər isə əksinə. Yuxarıda baxdığımız nümunə proqramda biz duzbucaqli sinfindən adi dəyişən elan etmişdik. Gəlin indi həmin proqramı unvan tipli dəyişəndən (dinamik) istifadə edilən halına baxaq. Proqrama baxmazdan öncə onu da qeyd edək ki, (bax \$8.1) dinamik obyektin funksiya və dəyişən heddlərinə müraciət etmək üçün . yox -> operatorundan istifadə olunur. Proqram nümunəsi

```

// prg_10_4.cpp
#include <iostream>

// duzbucaqli tipini elan edirik
class duzbucaqli
{
public:
duzbucaqli();
void terefler(int,int);
int sahe(void);
private:
int en;
int uzunluq;
};
//kanstruktur
duzbucaqli::duzbucaqli()
{
en = 10;
uzunluq = 15;
}
// duzbucaqli -nin terefler funksiyasinin metn kodu
//klasin en ve uzunluq gizli heddlerini qiymetlendiririk
void duzbucaqli::terefler(int x,int y)
{
en = x;
uzunluq = y;
}
// duzbucaqli -nin sahe funksiyasinin metn kodu
int duzbucaqli::sahe(void)
{
return en*uzunluq;
}
//~~~~~
int main()
{

duzbucaqli *duzbl;

//heleki yaddashda duzbl ucun hec bir yer ayrilmiyib
//duzbl -i yaradaq
duzbl = new duzbucaqli;
//kanstruktur avtomatik cagrilir
std::cout<<"duzbucaqlinin sahesi = "<<duzbl->sahe()<<"\n";
//duzbl -in tereflerine ayri qiymetler menimsedek
duzbl->terefler(5,8);
std::cout<<"duzbucaqlinin sahesi = "<<duzbl->sahe()<<"\n";
return 0;
}

```

Nəticə:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
duzbucaqlinin sahesi = 150
duzbucaqlinin sahesi = 40
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## \$11 Makroslar və başlıq fayllar.

Biz indiyə kimi proqramlarda

```
#include<iostream>, #include<string.h>
```

kimi sətirlərdən istifadə etdik və qeyd etdik ki proqramın mətninə bu sətirlərin əlavə olunması bizə `std::cout`, `std::cin`, `new`, `delete`, `strcpy` ... kimi funksiyalardan istifadə etməyə imkan verir.

Hər-hansı bir funksiyadan proqramda istifadə edə bilmək üçün proqrama bu funksiyanın elanı (adı və parametrlərinin qeyd edildiyi sətir) və mətni (funksiyanın kod hissəsi) verilməlidir. Biz öz funksiyalarımızı tərtib edərkən həm elanı, həm də mətni eyni faylda yerləşdirirdik.

Kompilyator imkan verir ki, biz ayrı-ayrı fayllarda elan olunmuş funksiya və dəyişənlərə öz proqramımızdan müraciət edə bilək.

Bunun üçün `#include` direktivindən istifadə edirlər.

```
#include<fayl.h> və ya #include"fayl.h" kimi.
```

Bir qayda olaraq proqrama `#include` vastəsilə əlavə olunan faylların sonu `.h` ilə bitir. Sadə proqram nümunəsinə baxmağımız kifayətdir. İndiyə kimi baxdığımız nümunələrdə bütün proqram kodunu bir fayla yerləşdirirdik.

İndi isə bizə iki və daha çox fayl lazım olacaq:

Dəyişənlərin, funksiyaların elan olduğu başlıq fayllar və bu dəyişən və funksiyalara müraciət edən proqram kodu faylları.

### Proqram 1.

menim\_faylim.h faylının mətni

```
// prg_11_1.cpp
```

```
/*başlıq fayli menim_faylim.h */
```

```
#ifndef MENIM_FAYLIM_H  
#define MENIM_FAYLIM_H
```

```
int yeni_deyishen;
```

```
#endif
```

proqram kodu prog2.cpp faylının mətni

```
#include <iostream>
```

```
#include "menim_faylim.h"
```

```
int main()  
{
```

```
yeni_deyishen = 5;
```

```
std::cout<<" yeni deyishen "  
<<yeni_deyishen<<"\n";  
}
```

Proqramı yerinə yetirək.

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
yeni deyishen 5
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```



menim\_faylim.h faylındakı

```
#ifndef  
#define  
#endif
```

makrosları **menim\_faylim.h** faylının bizim proqramam sonsuz əlavə olunmasının qarşısını alır. **menim\_faylim.h** başlıq faylında biz **int** tipli **yeni\_deyishen** dəyişəni elan edirik. Daha sonra **prog2.cpp** faylında **yeni\_deyishen** dəyişəninə müraciət edirik.

Əgər diqqət yetirdinizsə biz **iostream** faylını < və > vastəsilə, **menim\_faylim.h** faylını isə " və " simvolları vastəsilə proqrama əlavə etdik. Bu kompilyatora **menim\_faylim.h** başlıq faylının standart deyil, bizim tərəfimizdən yaradıldığını bildirir və kompilyator bu faylı bizim proqram yerləşən qovluqda axtarır.

## MAKROSLAR

C++ dilində istifadə olunan digər əhəmiyyətli vasitələrdən biri də makroslardır. Makroslar 2 cür olur: **şərt** makrosları və **təyin** makrosları.

### Təyin makrosalrı

Təyin makrosalrı **#define** direktivindən istifadə olunaraq yaradılır. Təyin makrosları hər hansı bir ifadənin başqa ifadə ilə əvəz edilməsinə xidmət edir. Misal üçün əgər biz proqramın hər hansı yerində **#define MAX\_QIYMET 1024** sətirini yerləşdiririksə onda kompilyator proqramda **MAX\_QIYMET** ifadəsinə rast gəldiyi bütün yerlərdə onu **1024** ilə əvəz edəcək.

Sadə proqrama baxaq:

```
#define MAX 8
```

```
int main()  
{  
int i,x[MAX];  
  
for (i=0, i<MAX; ++i)  
x[i]=i;  
  
return 0;  
}
```

Bu proqram 8 elementli tam tipli x cərgəsi elan edir və onun elementlərinə 0-dan 7-yə kimi qiymətlər mənimsədir.

### Şərt makrosları

Şərt makrosları **#ifdef**, **#ifndef**, **#endif** direktivlərdən istifadə olunaraq yaradılır. Şərt makrosları bizə imkan verir ki, müəyyən şərtədən asılı olaraq proqramın hər-hansı hissəsinin kompilyator tərəfindən nəzərə alınmamasını təmin edək.

Sintaksis belədir:

```
# if şərt  
proqram kodu  
#endif
```

Bu zaman əgər şərt 1 qiyməti alarsa onda kompilyator proqram kodu hissəsinə nəzərə alacaq, əks halda isə bu hissə kompilyator tərəfindən inkar ediləcək, başqa sözlə şərh kimi qəbul olunacaq.

## Əlavələr

### Əlavə A – bəzi standart funksiyalar

`std::cout` funksiyası.

`std::cout` funksiyası yaddaşın müxtəlif məlumatları ekrana çap etmək üçün istifadə olunur. Məsələn üçün əgər ekranda "Salam dünya" sətirini çap etmək istəyirsinizsə onda aşağıdakı kimi yazırıq:

```
std::cout<<"Salam dünya";
```

Əgər `std::cout` vasitəsilə ekrana müxtəlif məlumatlar göndərmək istəyirsinizsə onda bir neçə müxtəlif məlumatı "<<" vasitəsilə birləşdirə bilərik. Məsələn üçün tutaq ki, x,y,z dəyişənlərinin qiymətlərini çap etmək istəyirəm. Onda kod aşağıdakı kimi olar:

```
std::cout<<x<<y<<z;
```

`std::cin` funksiyası.

`std::cin` funksiyası `std::cout` funksiyasının gördüyü işin əksini görür. Əgər `std::cout` vasitəsilə biz dəyişənlərin qiymətlərini ekrana çap edirdiksə, `std::cin` vasitəsilə biz istifadəçinin klaviaturadan daxil etdiyi qiymətləri dəyişənlərə mənimsədirik. Məsələn üçün əgər mən hər-hansı x dəyişəninə istifadəçinin daxil etdiyi qiymət mənimsətmək istəyirəmsə onda aşağıdakı kimi yazıram:

```
std::cin>>x;
```

`std::cin` də `std::cout` kimi bir neçə dəyişənlə eyni anda işləməyə imkan verir. Məsələn üçün əgər mən x,y,z dəyişənlərinə istifadəçi tərəfindən daxil olunan qiymət mənimsətmək istəyirəmsə onda yazı bilərəm:

```
std::cin>>x>>y>>z;
```

### Əlavə B. ASCII Kodlar Cədvəli.

Bu cədvəldən istifadə etmək üçün , sadəcə lazım olan simvolu tap və solda yerləşən rəqəmlə yuxarıda yerləşən rəqəmin cəmi bu simvolun kodunu göstərir. Cədvəl B-1.10-luq say sistemində ASCII Kodlar Cədvəli.

Char	Dec						
\0	0	(sp)	32	@	64	`	96
(soh)	1	!	33	A	65	a	97
(stx)	2	"	34	B	66	b	98
(etx)	3	#	35	C	67	c	99
(eot)	4	\$	36	D	68	d	100
(enq)	5	%	37	E	69	e	101
(ack)	6	&	38	F	70	f	102
(bel)	7	'	39	G	71	g	103
(bs)	8	(	40	H	72	h	104

\t	9	)	41	l	73	i	105
\n	10	*	42	J	74	j	106
(vt)	11	+	43	K	75	k	107
(np)	12	,	44	L	76	l	108
(cr)	13	-	45	M	77	m	109
(so)	14	.	46	N	78	n	110
(si)	15	/	47	O	79	o	111
(dle)	16	0	48	P	80	p	112
(dc1)	17	1	49	Q	81	q	113
(dc2)	18	2	50	R	82	r	114
(dc3)	19	3	51	S	83	s	115
(dc4)	20	4	52	T	84	t	116
(nak)	21	5	53	U	85	u	117
(syn)	22	6	54	V	86	v	118
(etb)	23	7	55	W	87	w	119
(can)	24	8	56	X	88	x	120
(em)	25	9	57	Y	89	y	121
(sub)	26	:	58	Z	90	z	122
(esc)	27	;	59	[	91	{	123
(fs)	28	<	60	\	92		124

(gs)	29	=	61	]	93	}	125
(rs)	30	>	62	^	94	~	126
(us)	31	?	63	_	95	(del )	127

## Bəzi çalışmaların həlləri

### \$1. Giriş

#### 5 - ci çalışma.

Ekranda Mən C++ dilini örgənirəm sətrini çap edən proqram tərtib edin.

```
// hell_1_5.cpp
```

```
#include <iostream>
```

```
int main()
{
std::cout<<"Mən C++ silini orgenirem \n";
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>prog2.exe
```

```
Mən C++ silini orgenirem
```

```
C:\cpp\prog2\Debug>
```

```
C:\cpp\prog2\Debug>
```

### \$2. Dəyişənlər

#### 1 - ci çalışma.

Elə proqram yazın ki, istifadəçidən 5 ədəd daxil etməsini istəsin, daha sonra proqram bu ədədlərin cəmini ekranda çap etsin.

Nəticə:

```
// hell_2_1.cpp
```

```
#include < iostream >
```

```
int main(){
```

```
int x1,x2,x3,x4,x5,y;
```

```
/*en primitiv variant*/
```

```
std::cout<<"Zehmet olmasa 1-ci ededi daxil edin \n";
```

```
std::cin>>x1;
```

```
std::cout<<"Zehmet olmasa 2-ci ededi daxil edin \n";
```

```
std::cin>>x2;
```

```
std::cout<<"Zehmet olmasa 3-cu ededi daxil edin \n";
```

```
std::cin>>x3;
```

```
std::cout<<"Zehmet olmasa 4-cu ededi daxil edin \n";
```

```
std::cin>>x4;
```

```
std::cout<<"Zehmet olmasa 5-ci ededi daxil edin \n";
```

```
std::cin>>x5;
```

```
y = x1 + x2 + x3 + x4 + x5;
```

```
std::cout<<"Daxil etdiyiniz ededlerin cemi = "<<y<<"\n";
```

```
}
```

Nəticə:

```
C:\cpp\prog2\Debug>prog2.exe
Zəhmət olmasa 1-ci ədədi daxil edin
45
Zəhmət olmasa 2-ci ədədi daxil edin
34
Zəhmət olmasa 3-cu ədədi daxil edin
123
Zəhmət olmasa 4-cu ədədi daxil edin
7
Zəhmət olmasa 5-ci ədədi daxil edin
8
Daxil etdiyiniz ədədlərin cəmi = 217
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## 2 - ci çalışma.

Elə proqram tərtib edin ki, istifadəçidən üçbucağın tərəflərinin uzunluğunu daxil etməsini istəsin və üçbucağın perimetrini çap etsin.

```
//hell_2_2.cpp
#include <iostream>

int main()
{
    int ter1, ter2, ter3, perim;

    std::cout<<"Zehmet olmasa ucbucaqin 1-ci terefinin uzunlugunu daxil edin \n";
    std::cin>>ter1;
    std::cout<<"Zehmet olmasa ucbucaqin 2-ci terefinin uzunlugunu daxil edin \n";
    std::cin>>ter2;
    std::cout<<"Zehmet olmasa ucbucaqin 3-cu terefinin uzunlugunu daxil edin \n";
    std::cin>>ter3;

    perim = ter1 + ter2 + ter3;
    std::cout<<"Ucbucaqin perimetri = "<<perim<<"\n";

}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Zəhmət olmasa ucbucaqin 1-ci terefinin uzunlugunu daxil edin
23
Zəhmət olmasa ucbucaqin 2-ci terefinin uzunlugunu daxil edin
34
Zəhmət olmasa ucbucaqin 3-cu terefinin uzunlugunu daxil edin
45
Ucbucaqin perimetri = 102
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### \$3. Operatorlar

#### 1 - ci çalışma.

Elə program yazın ki, istifadəçidən 2 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.

```
//hell_3_1.cpp
#include <iostream>

int main()
{

int dey1, dey2, max;

std::cout<<"Birinci ededi daxil edin \n";
std::cin>>dey1;
std::cout<<"ikinci ededi daxil edin \n";
std::cin>>dey2;

if (dey1 > dey2)
max = dey1;
else
max = dey2;

std::cout<<"Bu ededlerden en boyuyu "<<max<<" -dir\n";

}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Birinci ededi daxil edin
234
ikinci ededi daxil edin
457
Bu ededlerden en boyuyu 457 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

#### 2 - ci çalışma.

Elə program yazın ki, istifadəçidən 3 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.

```
//hell_3_2.cpp
#include <iostream>

int main()
{

int dey1, dey2, dey3, max;

std::cout<<"Birinci ededi daxil edin \n";
std::cin>>dey1;
std::cout<<"ikinci ededi daxil edin \n";
std::cin>>dey2;
std::cout<<"ucuncu ededi daxil edin \n";
std::cin>>dey3;

if ((dey1 > dey2) && (dey1 > dey3))
max = dey1;
```

```

else
if ((dey2 > dey1) && (dey2 > dey3))
max = dey2;
else
max = dey3;

std::cout<<"Bu ededlerden en boyuyu "<<max<<" -dir \n";

}

```

Nəticə:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Birinci ededi daxil edin
23
Ikinci ededi daxil edin
456
Ucuncu ededi daxil edin
981
Bu ededlerden en boyuyu 981 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

#### 4 - cü çalışma.

Elə program qurun ki, istifadəçinin daxil etdiyi ədəd sayda ekranda 'a' simvolu çap etsin.

```

//hell_3_4.cpp
#include <iostream>

```

```

int main()
{

int i,dey;

std::cout<<"Her hansı daxil edin \n";
std::cin>>dey;

for (i=0; i<dey; ++i)
std::cout<<'a'<<" ";

std::cout<<"\n";

}

```

Nəticə:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Her hansı daxil edin
6
a a a a a a
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```



## 5 - ci çalışma.

Elə program qurun ki, istifadəçidən hər-hansı ədəd qəbul etsin. Əgər bu ədəd 100-dən böyük olarsa ondaekranda 100 dəfə 'c' simvolu çap etsin, 50 ilə 100 arasında olarsa ekranda həmin ədəd sayda 'b' simvolu çap etsin,50 -dən kiçik olarsa həmin ədəd sayda 'a' simvolu çap etsin.

```
//hell_3_5.cpp
#include <iostream>

int main()
{

int i,dey;

std::cout<<"Her hansı daxil edin \n";
std::cin>>dey;

if (dey > 100 )
for (i=0; i<100; ++i)
std::cout<<'c'<<" ";
else
if (dey > 50)
for (i=0; i<dey; ++i)
std::cout<<'b'<<" ";
else
for (i=0; i<dey; ++i)
std::cout<<'a'<<" ";

std::cout<<"\n";
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Her hansı daxil edin
12
a a a a a a a a a a a a
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## 6 - cı çalışma.

**for** dövr operatorundan istifadə etməklə ekranda sonsuz olaraq "unix" kəlməsini çap edən program yazın.

```
//hell_3_6.cpp
#include <iostream>

int main()
{

int i;

for (i=0; i<1; i=i)
std::cout<<"unix\n";

}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
unix
unix
unix
unix
unix
unix
unix ^C
C:\cpp\prog2\Debug>
```

### 7 - ci çalışma.

**while** dövr operatorundan istifadə etməklə ekranda sonsuz olaraq "linux" kəlməsini çap edən program yazın.

```
//hell_3_7.cpp
#include <iostream>

int main()
{
while(1)
std::cout<<"linux\n";
}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
linux
linux
linux
linux
linux
linux
linux ^C
C:\cpp\prog2\Debug>
```

## \$4. Ünvan dəyişənləri

### 1 - ci çalışma.

Ancaq ünvan dəyişənlərindən istifadə etməklə iki ədədin cəmini hesablayan program tərtib edin.

```
//hell_4_1.cpp
#include <iostream>

int main(int argc, char *argv[])
{

int *dey1, *dey2, *cem;

// unvan deyishenleri ucun yaddashda yer ayiririq
dey1 = new int;
dey2 = new int;
cem = new int;

std::cout<<"Birinci ededi daxil edin \n";
std::cin>>*dey1;
std::cout<<"Ikinci ededi daxil edin \n";
std::cin>>*dey2;

*cem = *dey1 + *dey2;
std::cout<<"Bu iki ededin cemi <<*cem<<" -dir\n";

}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Birinci ededi daxil edin
45
Ikinci ededi daxil edin
234
Bu iki ededin cemi 279 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### 2- ci çalışma.

Ancaq ünvan dəyişənlərindən istifadə etməklə iki ədədin maksimumunu hesablayan program tərtib edin.

```
//hell_4_2.cpp
#include <iostream>

int main(int argc, char *argv[])
{
int *dey1, *dey2, *max;

// unvan deyishenleri ucun yaddashda yer ayiririq
dey1 = new int;
dey2 = new int;
max = new int;

std::cout<<"Birinci ededi daxil edin \n";
```

```
std::cin>>*dey1;
std::cout<<"ikinci ededi daxil edin \n";
std::cin>>*dey2;

if (*dey1 > *dey2)
*max = *dey1;
else
*max = *dey2;

std::cout<<"Bu iki ededin en boyuyu " <<*max<<" -dir\n";

}
```

Nəticə:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Birinci ededi daxil edin
45
ikinci ededi daxil edin
234
Bu iki ededin cemi 279 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

## \$5. Funksiyalar

### 1 - ci çalışma.

Funksiyalardan istifadə etməklə iki ədədin maksimumunu hesablayan proqram tərtib edin.

```
//hell_5_1.cpp
#include <iostream>

// en_boyuk funksiyasının elani
int en_boyuk (int x, int y);

int main(int argc, char *argv[])
{
    int dey1, dey2, max;

    std::cout<<"Birinci ededi daxil edin \n";
    std::cin>>dey1;
    std::cout<<"Ikinci ededi daxil edin \n";
    std::cin>>dey2;

    max = en_boyuk(dey1, dey2);

    std::cout<<"Bu iki ededin en boyuyu "<<max<<" -dir \n";

}

// en_boyuk funksiyasının proqram kodu
int en_boyuk ( int dey1, int dey2)
{
    if (dey1 > dey2)
        return dey1;
    else
        return dey2;
}
```

Nəticə:

```
C:\cpp\prog2\Debug>./prog2.exe
Birinci ededi daxil edin
5
Ikinci ededi daxil edin
67
Bu iki ededin en boyuyu 67 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### 2 - ci çalışma.

Elə funksiya qurun ki, istifadəçidən 10 tam ədəd daxil etməsini istəsin və onların cəmini qaytarsın. Bu funksiyaadan istifadə etməklə proqram qurun və onu icra edin.

```
//hell_5_2.cpp
#include <iostream>

// cem funksiyasının elani
int cem (void);
```

```

int main(int argc, char *argv[])
{

int dey1, dey2 ;

dey2 = cem();
std::cout<<"Daxil etdiyiniz ededlerin cemi "<<dey2<<" -dir \n";

}

// cem funksiyasının program kodu
int cem ( void)
{
int i, x, y;

// y-den umumi cemi yadda saxlamaq ucun istifade edəcəyik
y=0;

for (i=1; i<=10; ++i)
{
std::cout<<i<<" -ci ededi daxil edin\n";
std::cin>>x;
y = y + x;
}

return y;
}

```

Nəticə:

```

C:\cpp\prog2\Debug>./prog2.exe
1 -ci ededi daxil edin
4
2 -ci ededi daxil edin
56
3 -ci ededi daxil edin
67
4 -ci ededi daxil edin
23
5 -ci ededi daxil edin
45
6 -ci ededi daxil edin
67
7 -ci ededi daxil edin
2
8 -ci ededi daxil edin
6
9 -ci ededi daxil edin
78
10 -ci ededi daxil edin
123
Daxil etdiyiniz ededlerin cemi 471 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

### 3 - cü çalışma.

**kvadrat** adlı elə funksiya tərtib edin ki, ekranda "\*" simvollarından ibarət, tərəflərinin uzunluğu 10 olan, kvadrat çəksin (içini doldurmaqla). **kvadrat** funksiyaından istifadə etməklə proqram qurub icra edin.

```
//hell_5_3.cpp
#include <iostream>

// kvadrat funksiyaının elani
void kvadrat (void);

int main(int argc, char *argv[])
{
kvadrat();
}

// kvadrat funksiyaının proqram kodu
void kvadrat ( void)
{
int i, j;

for(i=0; i <10; ++i){
for(j=0; j <10; ++j)
std::cout<<"* " ;
std::cout<<"\n";
}

}
```

Nəticə:

```
C:\cpp\prog2\Debug>./prog2.exe
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### 4 - cü çalışma.

Çalışma 3-dəki kvadrat funksiyaını elə dəyişin ki, tərəflərinin sayı və təşkil olunduğu simvol bu funksiyaaya parametr kimi ötürülsün. Bu funksiyaadan istifadə etməklə elə proqram qurun ki, istifadəçidən hər hansı simvol və ədəd daxil etməsini istəsin, daha sonra isə ekranda həmin parametrlərə uyğun kvadrat çəksin.

```

// hell_5_4.cpp
#include <iostream>

// kvadrat funksiyasinin elani
void kvadrat (int , char);

int main(int argc, char *argv[])
{
int x;
char c;

std::cout<<"kvadratin terefinin uzunlugunu ve \n"
        <<" reng simvolunu daxil edin \n";
std::cin>>x>>c;

kvadrat(x,c);

}

/* kvadrat funksiyasinin program kodu */
void kvadrat (int teref, char reng)
{
int i, j;

for(i=0; i<teref; ++i){
for(j=0; j<teref; ++j)
std::cout<<reng<<" ";
std::cout<<"\n";
}
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
kvadratin terefinin uzunlugunu ve
reng simvolunu daxil edin
7 @
@ @ @ @ @ @ @
@ @ @ @ @ @ @
@ @ @ @ @ @ @
@ @ @ @ @ @ @
@ @ @ @ @ @ @
@ @ @ @ @ @ @
@ @ @ @ @ @ @
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```



## 5 - ci çalışma.

Çalışma 4-dəki funksiyaları elə dəyişin ki, istifadəçi kvadratın içinin rəngləndiyi simvolu da daxil edə bilsin. Bu funksiyadan istifadə etməklə program tərtib edib, icra edin.

```
// hell_5_5.cpp
#include <iostream>

// kvadrat funksiyasının elanı
void kvadrat (int , char, char);

int main(int argc, char *argv[])
{
    int x;
    char c,d;

    std::cout<<"kvadratın terefinin uzunluğunu , terefinin ve\n"
        <<" daxilinin reng simvollarını daxil edin \n";

    std::cin>>x>>c>>d;

    kvadrat(x,c,d);
}

// kvadrat funksiyasının program kodu
void kvadrat (int teref, char t_reng, char d_reng)
{
    /* burada bir balaca izaha ehtiyac var.
    cekdiyimiz fiqurun kenarlari t_reng simvolu, daxili ise
    d_reng simvolu ile renglenmelidir. */

    int i, j;

    /* tutaq ki, terefin uz-gu 5, terefin reng simvolu #,
    daxilin reng simvolu ise @ -dir.
    Onda ashagidaki kod ekranda

    # # # # #

    cap edecek */

    for(j=0; j<teref; ++j)
    std::cout<<t_reng<<" ";
    std::cout<<"\n";

    /* ashagidaki kod 5 - 2 = 3 defe tekrar olunacaq.
    ve her defe tekrar olunanda ekranda evvelce
    #, daha sonra 5 - 2 = 3 sayda @ , daha sonra ise #
    simvolunu cap edecek. ashagidaki kimi:
    # @ @ @ #
    3 defe tekrar olunanda ise ekranda bu shekil alinin
    # @ @ @ #
    # @ @ @ #
    # @ @ @ #
    */

    for(i=1; i<teref-1; ++i)
```

```

{
std::cout<<t_reng<<" ";

for(j=1; j<teref-1; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}

/*sonda yuxarida baxdigimiz kod,
figuru tamalamaq ucun tekrar olunur
# # # # #
*/

for(j=0; j<teref; ++j)
std::cout<<t_reng<<" ";
std::cout<<"\n";

/*
neticede ekranda terefinin uzunlugu 5, kenarlari #,
daxili ise @ simvolu ile renglenmish kvadrat alinir.
ashagidaki kimi.

# # # # #
# @ @ @ #
# @ @ @ #
# @ @ @ #
# # # # #

*/
}

```

Netice:

```

C:\cpp\prog2\Debug>./5_5
kvadratin terefinin uzunlugunu , terefinin ve
daxilinin reng simvollarini daxil edin
7 # ~
# # # # # # #
# ~ ~ ~ ~ ~ #
# ~ ~ ~ ~ ~ #
# ~ ~ ~ ~ ~ #
# ~ ~ ~ ~ ~ #
# ~ ~ ~ ~ ~ #
# # # # # # #
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## 6 - cı çalışma.

Çalışma 5-in tələblərini yerinə yetirən romb funksiyası qurun, hansı ki, ekranda romb çəksin. Bu funksiyadan istifadə edib proqram tərtib edin və icra edin.

```
// hell_5_6.cpp
#include <iostream>

// romb funksiyasının elanı
void romb (int , char, char);

int main(int argc, char *argv[])
{
    int x;
    char c,d;

    std::cout<<"rombun terefinin uzunlugunu , terefinin ve\n"
        <<"daxilinin reng simvollarini daxil edin \n";

    std::cin>>x>>c>>d;

    romb(x,c,d);
}

// romb funksiyasının proqram kodu
void romb (int teref, char t_reng, char d_reng)
{
    /*
    eger istifadeci 6 , =, + parametrlərini daxil etse
    proqram ekranda ashagidaki kimi romb cekmelidir.
        =
        = + =
        = + + + =
        = + + + + + =
        = + + + + + + + =
    = + + + + + + + + + =
        = + + + + + + + =
        = + + + + + =
        = + + + =
        = + =
        =
    */
    int i, j, k;
    // ust hisse

    for(j=0; j<teref; ++j)
        std::cout<<" ";
    std::cout<<t_reng<<" ";
    std::cout<<"\n";
    for(i=1; i<teref; ++i){
        for( j=0; j<teref-i; ++j)
            std::cout<<" ";
        std::cout<<t_reng<<" ";
        for (j=teref - i + 1; j<teref + i; ++j)
            std::cout<<d_reng<<" ";
        std::cout<<t_reng<<" ";
        std::cout<<"\n";
    }
}
```

```

}
// alt hisse
for(i=2; i<teref; ++i){
for( j=0; j<i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=i ; j<2*teref - i - 1; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}

```

Netice:

C:\cpp\prog2\Debug>./prog2.exe  
rombun terefinin uzunlugunu , terefinin ve  
daxilinin reng simvollarini daxil edin

```

6 + -
      +
    + - +
  + - - - +
+ - - - - - +
+ - - - - - - - +
+ - - - - - - - +
  + - - - - - +
    + - - - +
      + - +
        +

```

C:\cpp\prog2\Debug>

## 7 - ci çalışma.

7. Çalışma 5-in tələblərini yerinə yetirən ucbucaq funksiyası qurun, hansı ki, ekranda ucbucaq çəksin. Bu funksiyadan istifadə edib proqram tərtib edin və icra edin.

```

// hell_5_7.cpp
#include <iostream>

```

```

// ucbucaq funksiyasının elanı
void ucbucaq (int , char, char);

```

```

int main(int argc, char *argv[])
{
int x;
char c,d;

```

```

std::cout<<"ucbucaqın terefinin uzunlugunu , terefinin ve\n"
<<"daxilinin reng simvollarini daxil edin \n";

```

```

std::cin>>x>>c>>d;

```

```

ucbucaq(x,c,d);

```

```

}

// ucucuq funksiyasinin program kodu
void ucucuq (int teref, char t_reng, char d_reng)
{
/*
eger istifadeci 5 , &, = parametrlərini daxil etse
program ekranda ašagıdaki kimi ucucuq cekmelidir.
      =
      = + =
      = + + =
      = + + + =
      = + + + + =
      = = = = = = = = =
*/
int i, j, k;

/* ust hisse */
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
for(i=1; i<teref - 1; ++i){
for( j=0; j<teref-i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=teref - i + 1; j<teref + i; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}

// alt hisse
std::cout<<" ";
for(i=1; i<2*teref; ++i)
std::cout<<t_reng<<" ";
std::cout<<"\n";
std::cout<<"\n";
}

```

Netice:

```

C:\cpp\prog2\Debug>./prog2.exe
ucucuqın terefinin uzunlugunu , terefinin ve
daxilinin reng simvollarini daxil edin
7 u o

```

```

      u
      u o u
      u o o o u
      u o o o o o u
      u o o o o o o o u
      u o o o o o o o o o u
u u u u u u u u u u u u u
C:\cpp\prog2\Debug>

```

## 8 - ci çalışma.

Kvadrat, romb və ucbucaq funksiyalarından istifadə etməklə elə proqram tərtib edin ki, əvvəl istifadəçidən tərəfin uzunluğun, tərəfin və fiqurun daxil rəngləmək üçün simvolları daxil etməyi istəsin. Daha sonra istifadəçidən 1,2 və 3 rəqəmlərindən birini daxil etməyini istəsin. Əgər istifadəçi 1 daxil edərsə onda ekranda kvadrat, 2 daxil edərsə romb, 3 daxil edərsə ucbucaq çəksin.

```
//hell_5_8.cpp
#include <iostream>

void kvadrat (int , char, char);
void romb (int , char, char);
void ucbucaq (int , char, char);

int main(int argc, char *argv[])
{
    int x,k;
    char c,d;
    std::cout<<"Fiqurun terefinin uzunlugunu , terefinin daxilinin \n"
        <<"ve reng simvollarini daxil edin \n";
    std::cin>>x>>c>>d;

    std::cout<<"Ektranda cekmek istediyyiniz fiqurun nomresini daxil edin\n";
    std::cout<<"Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3\n"
        <<" reqemini daxil edin\n";

    std::cin>>k;

    switch(k){
    case 1:
        ucbucaq(x,c,d);
        break;
    case 2:
        romb(x,c,d);
        break;
    case 3:
        kvadrat(x,c,d);
    }
}

/* kvadrat funksiyasinin proqram kodu */
void kvadrat (int teref, char t_reng, char d_reng) {

    /* burada bir balaca izaha ehtiyac var.
    cekdiyimiz fiqurun kenarlari t_reng simvolu,
    daxili ise d_reng simvolu ile renglenmelidir.
    */
    int i, j;
    /*tutaq ki, terefin uz-gu 5, terefin reng simvolu #,
    daxilin reng simvolu ise @ -dir.
    Onda ashagidaki kod ekranda
    # # # # #
    cap edecek
    */
    for(j=0; j<teref; ++j)
        std::cout<<t_reng<<" ";
    std::cout<<"\n";
    /*ashagidaki kod 5 - 2 = 3 defe tekrar olunacaq.
```

```

ve her defe tekrar olunanda ekranda evvelce
#, daha sonra 5 - 2 = 3 sayda @ , daha sonra ise #
simvolunu cap edecek. ashagidaki kimi:
# @ @ @ #
3 defe tekrar olunanda ise ekranda bu shekil alinin
# @ @ @ #
# @ @ @ #
# @ @ @ #
*/
for(i=1; i<teref-1; ++i){
std::cout<<t_reng<<" ";
for(j=1; j<teref-1; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
/*sonda yuxarida baxdigimiz kod,
fiquru tamalamaq ucun tekrar olunur
# # # # #
*/
for(j=0; j<teref; ++j)
std::cout<<t_reng<<" ";
std::cout<<"\n";
/*
neticede ekranda terefinin uzunlugu 5, kenarlari #,
daxili ise @ simvolu ile renglenmish kvadrat alinir.
ashagidaki kimi.
# # # # #
# @ @ @ #
# @ @ @ #
# @ @ @ #
# # # # #
*/
}

/*=====*/
/* romb funksiyasinin proqram kodu */
void romb (int teref, char t_reng, char d_reng)
{
/*
eger istifadeci 6 , =, + parametrlarini daxil etse
proqram ekranda ashagidaki kimi romb cekmelidir.
=
= + =
= + + + =
= + + + + + =
= + + + + + + + =
= + + + + + + + + + =
= + + + + + + + + + =
= + + + + + =
= + + + =
= + =
=
*/
int i, j, k;
/* ust hisse */
for(j=0; j<teref; ++j)
std::cout<<" ";

```

```

std::cout<<t_reng<<" ";
std::cout<<"\n";
for(i=1; i<teref; ++i){
for( j=0; j<teref-i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=teref - i + 1; j<teref + i; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
/* alt hisse */
for(i=2; i<teref; ++i){
for( j=0; j<i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=i ; j<2*teref - i - 1; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}

/*=====*/
/* ucbucaq funksiyasinin program kodu */
void ucbucaq (int teref, char t_reng, char d_reng)
{
/*
eger istifadeci 5 , +, = parametrlərini daxil etse
program ekranda ashagidaki kimi romb cekmelidir.
=
= + =
= + + + =
= + + + + + =
= = = = = = = =
*/
int i, j, k;
/* ust hisse */
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
for(i=1; i<teref - 1; ++i){
for( j=0; j<teref-i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=teref - i + 1; j<teref + i; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
/* alt hisse */
std::cout<<" ";

```



```

for(i=1; i<2*teref; ++i)
std::cout<<t_reng<<" ";
std::cout<<"\n";
std::cout<<"\n";
}

```

Nəticə:

```

C:\cpp\prog2\Debug>./prog2.exe
Figurun terefinin uzunlugunu , terefinin
ve daxilinin reng simvollarini daxil edin
9 – k
Ekranda cekmek istediyyiniz fiqurun nomresini daxil edin
Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3
reqemini daxil edin
3
- - - - -
- k k k k k k k -
- k k k k k k k -
- k k k k k k k -
- k k k k k k k -
- k k k k k k k -
- k k k k k k k -
- k k k k k k k -
- - - - -
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## 9 - cu çalışma.

Çalışma 8-i elə dəyişin ki, proqram istidəçidən fiqurun tərəfinin uzunluğunu və rəng simvoları daxil etdikdən sonra istədiyi fiqurun çəkilməsi üçün 1,2,3 simvollarından birini daxil etməsini istəsin. Bu prosesi istifadəçi 0 rəqəmi daxil edənə kimi təkrar eləsin. Bu zaman proqramın istifadəçidən tərəfin uzunluğu və rəng simvollarını qəbul edən hissəsini də ayrı bir funksiya kimi tərtib edin.

```

//hell_5_9.cpp
#include <iostream>

```

```

void kvadrat (int , char, char);
void romb (int , char, char);
void ucbucaq (int , char, char);
void hazirliq_ishleri( int *);
void yerine_yetir(int);

```

```

/* main - proqramin esas funksiyasi */
int main(int argc, char *argv[])
{

```

```

int k;
hazirliq_ishleri(&k);
/* gorduyunuz kimi men daxil etdiyim variantda
ilkkin_hazirliq funksiyasi 2 defe cagirilib.
mence bu optimal variant deyil.
eger daha lokanik variantini tertib ede bilersinizse,
oz hellinizi progbits.az saytina yerleshdirmeyinizi xahish edirem.
*/
while (k!=0){
yerine_yetir(k);

```

```

    hazirliq_ishleri(&k);
}
}
/*=====*/

/* kvadrat funksiyasinin proqram kodu */
void kvadrat (int teref, char t_reng, char d_reng)
{

/* burada bir balaca izaha ehtiyac var.
cekdiyimiz fiqurun kenarlari t_reng simvolu, daxili ise
d_reng simvolu ile renglenmelidir. */

int i, j;
/*tutaq ki, terefin uz-gu 5, terefin reng simvolu #, daxilin reng
simvolu ise @ -dir. Onda ashagidaki kod ekranda
# # # # #
cap edecek
*/
for(j=0; j<teref; ++j)
std::cout<<t_reng<<" ";
std::cout<<"\n";
/*ashagidaki kod 5 - 2 = 3 defe tekrar olunacaq.
ve her defe tekrar olunanda ekranda evvelce
#, daha sonra 5 - 2 = 3 sayda @ , daha sonra ise #
simvolunu cap edecek. ashagidaki kimi:
# @ @ @ #
3 defe tekrar olunanda ise ekranda bu shekil alinin
# @ @ @ #
# @ @ @ #
# @ @ @ #
*/
for(i=1; i<teref-1; ++i){
std::cout<<t_reng<<" ";
for(j=1; j<teref-1; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
/*sonda yuxarida baxdigimiz kod, fiquru tamalamaq ucun tekrar olunur
# # # # #
*/
for(j=0; j<teref; ++j)
std::cout<<t_reng<<" ";
std::cout<<"\n";
/*
neticede ekranda terefinin uzunlugu 5, kenarlari #,
daxili ise @ simvolu ile renglenmish kvadrat alinir.
ashagidaki kimi.
# # # # #
# @ @ @ #
# @ @ @ #
# @ @ @ #
# # # # #
*/
}

/* romb funksiyasinin proqram kodu */

```

```
void romb (int teref, char t_reng, char d_reng)
{
```

```
/* eger istifadeci 6 , =, + parametrlərini daxil etse proqram ekranda
ashagidaki kimi romb cekmelidir.
```

```

      =
    = + =
  = + + + =
= + + + + + =
= + + + + + + + =
= + + + + + + + + + =
= + + + + + + + =
  = + + + + + =
    = + + + =
      = + =
        =

```

```
*/
```

```
int i, j, k;
```

```
/* ust hisse */
```

```
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
for(i=1; i<teref; ++i){
for( j=0; j<teref-i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=teref - i + 1; j<teref + i; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
```

```
/* alt hisse */
```

```
for(i=2; i<teref; ++i){
for( j=0; j<i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=i ; j<2*teref - i - 1; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
```

```
/* ucbucaq funksiyasının proqram kodu */
```

```
void ucbucaq (int teref, char t_reng, char d_reng)
{
```

```
/* eger istifadeci 5 , &, = parametrlərini daxil etse proqram ekranda
ashagidaki kimi romb cekmelidir.
```

```

      =
    = + =

```

```

    = + + + =
    = + + + + + =
    = = = = = = = =
*/
int i, j, k;
/* ust hisse */
for(j=0; j<teref; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
for(i=1; i<teref - 1; ++i){
for( j=0; j<teref-i; ++j)
std::cout<<" ";
std::cout<<t_reng<<" ";
for (j=teref - i + 1; j<teref + i; ++j)
std::cout<<d_reng<<" ";
std::cout<<t_reng<<" ";
std::cout<<"\n";
}
/* alt hisse */
std::cout<<" ";
for(i=1; i<2*teref; ++i)
std::cout<<t_reng<<" ";
std::cout<<"\n";
std::cout<<"\n";
}

/*=====*/
void hazirliq_ishleri( int *k)
{
std::cout<<"Ekranada cekmek istediyyiniz fiqurun nomresini daxil edin\n";
std::cout<<"Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3 reqemini daxil edin\n";
std::cout<<"Cixish ucun 0 reqemini daxil edin\n";
std::cin>>*k;
}

/*=====*/
void yerine_yetir(int k)
{
int x;
char c,d;
std::cout<<"Fiqurun terefinin uzunlugunu , terefinin ve daxilinin reng
simvollarini daxil edin \n";
std::cin>>x>>c>>d;
switch(k){
case 1:
ucbucaq(x,c,d);
break;
case 2:
romb(x,c,d);
break;
case 3:
kvadrat(x,c,d);
}
}
}

```

Nəticə:

C:\cpp\prog2\Debug>./prog2.exe

Ekranda cekmek istediyyiniz fiqurun nomresini daxil edin

Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3 reqemini daxil edin

Cixish ucun 0 reqemini daxil edin

1

Fiqurun terefinin uzunlugunu , terefinin ve daxilinin reng simvollarini daxil edin

5 y u

```
      y
     y u y
    y u u u y
   y u u u u u y
  y y y y y y y y y
```

Ekranda cekmek istediyyiniz fiqurun nomresini daxil edin

Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3 reqemini daxil edin

Cixish ucun 0 reqemini daxil edin

3

Fiqurun terefinin uzunlugunu , terefinin ve daxilinin reng simvollarini daxil edin

7 \* -

```
* * * * * * *
* - - - - - *
* - - - - - *
* - - - - - *
* - - - - - *
* - - - - - *
* - - - - - *
* * * * * * *
```

Ekranda cekmek istediyyiniz fiqurun nomresini daxil edin

Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3 reqemini daxil edin

Cixish ucun 0 reqemini daxil edin

0

C:\cpp\prog2\Debug> C:\cpp\prog2\Debug>

## \$6. Cərgələr

### 1 - ci çalışma.

max programını ele dəyişin ki, istifadəçinin daxil etdiyi ədələrin içində ən kiçiyini tapsın.

```
//hell_6_1.cpp
#include <iostream>

int main(int argc, char *argv[])
{
    /*en coxu 100 elemente hesablanib */
    int i, x[100], say, min;

    std::cout<<"100 -den kicik her hansı bir eded daxil edin > 0 \n";
    std::cin>>say;
    std::cout<<say<<" sayda eded daxil edin\n";

    for (i=0; i<say; i++ )
        std::cin>>x[i];

    min = x[0];

    for (i=0; i<say; i++)
        if (x[i] < min) min = x[i];

    std::cout<<"sizin daxil etdiyiniz ededlerin icinde en kiciyi "<<min<<"-dir\n";
}
```

Nəticə:

```
C:\cpp\prog2\Debug>./prog2.exe
100 -den kicik her hansı bir eded daxil edin > 0
7
7 sayda eded daxil edin
1 2 5 34 56 890 5
sizin daxil etdiyiniz ededlerin icinde en kiciyi 1 -dir
C:\cpp\prog2\Debug>
```

### 4 - cü çalışma.

Elə program qurun ki, istifadəçinin daxil etdiyi ədədləri artan sıra ilə düzsün.

```
//hell_6_4.cpp
#include <iostream>

int main(int argc, char *argv[])
{
    /*en coxu 100 elemente hesablanib */
    int i, j, x[100], say, min, movqe, kecid;
    std::cout<<"100 -den kicik her hansı bir eded daxil edin > 0 \n";
    std::cin>>say;
    std::cout<<say<<" sayda eded daxil edin\n";

    for (i=0; i<say; i++ )
        std::cin>>x[i];
    for (i=0; i<say; ++i){
        min = x[i];
        movqe = i;
        for (j=i; j<say; ++j)
```

```

if (x[j] < min){
min = x[j];
movqe = j;
}

kecid = x[i];
x[i] = min;
x[movqe] = kecid;
}

for (i=0; i<say; i++)
std::cout<<x[i]<<" ";

std::cout<<"\n";
}

```

Netice:

```

C:\cpp\prog2\Debug>./prog2.exe
100 -den kicik her hansı bir eded daxil edin > 0
7
7 sayda eded daxil edin
12 3 56 678 0 90 345
0 3 12 56 90 345 678
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## 6 - cı çalışma.

Elə funksiya tərtib edin ki, verilmiş cərgənin elementləri arasında ən böyüyünü tapsın. Bu funksiya istifadə edərək proqram tərtib edin ki, istifadəçidən əvvəl say, daha sonra bu say qədər hər-hansı ədəd daxil etməsini istəsin və bu ədədlərin ən böyüyünü çap etsin.

```

//hell_6_6.cpp
#include <iostream>

int max(int *, int);

int main(int argc, char *argv[])
{
    /*en coxu 100 elemente hesablanib */
    int i, x[100], say, max_eded;
    std::cout<<"100 -den kicik her hansı bir eded daxil edin > 0 \n";
    std::cin>>say;
    std::cout<<say<<" sayda eded daxil edin\n";

    for (i=0; i<say; i++ )
    std::cin>>x[i];

    max_eded = max(x,say);
    std::cout<<"sizin daxil etdiyiniz ededlerin icinde en boyuyu " <<max_eded<<"
-dir\n";
}
/*=====*/
int max(int *x, int say)
{
    int i , max;
    max = x[0];

```

```
for (i=0; i<say; i++)
if (x[i] < max) max = x[i];

return max;
}
```

Netice:

```
C:\cpp\prog2\Debug>./prog2.exe
100 -den kicik her hansı bir eded daxil edin > 0
6
6 sayda eded daxil edin
1 34 567 0 34 23
sizin daxil etdiyiniz ededlerin icinde en kiciyi 0 -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### **7 - ci çalışma.**

6 - cı proqramı elə dəyişin ki, istifadəçi say olaraq 0 daxil edənə kimi proqram təkrar olunsun.  
Bax çalışma 5\_9 - a.



## \$7. Sətirlər

### 1 - ci çalışma.

Elə program qurun ki, istifadəçinin daxil etdiyi sətirin 5-ci simvolu ilə 15-ci simvolu arasında qalan hissəsini çap etsin. Əgər sətirin uzunluğu 20-dən kiçik olarsa onda ekranda bu barədə məlumat çap etsin.

```
// hell_7_1.cpp
#include <iostream>
#include <string.h>

int main(int argc, char *argv[])
{

/*1024 elementden ibaret setir elan edirik*/
char setir[1024];

std::cout<<"Her hansı setir daxil edin\n";
std::cin>>setir;

if (strlen(setir) < 20)
std::cout<<"Sizin daxil etdiyiniz setrin uzunluđu 20- dan kicikdir \n";
else
{
char *p, *q, bufer[20];
memset(bufer, 0 ,20);

/* p-ni surushdururuk 5-ci simvolun uzerine*/
p = setir + 5;
/* 5-ci simvoldan bashlayaraq novbeti 9 simvolu bufere kocururuk*/
strncpy(bufer,p,9);

std::cout<<"5-ci simvol ile 15 -ci simvol arasinda olan hisse "
<<bufer<<" -dir\n";
}
}
```

Netice:

```
C:\cpp\prog2\Debug>./prog2.exe
Her hansı setir daxil edin
Bir zerrenin ishigina milyonlar sherik.
5-ci simvol ile 15 -ci simvol arasinda olan hisse "errenin i" -dir
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### 2 - ci çalışma.

Elə program tərtib edin ki, istifadəçidən 6 sətir qəbul etsin və bu sətirləri ardıcıl birləşdirərək tam sətir kimi çap etsin.

```
// hell_7_2.cpp
#include <iostream>
#include <string.h>

int main(int argc, char *argv[])
{

char setir[1024] , bufer[20];
```

```

int i,k;
memset(setir,0,1024);
memset(bufer,0,10);

std::cout<<"Uzunluđu 20-dan kicik olan 6 setir daxil edin\\n";

for(i=0; i<6; ++i){
std::cin>>bufer;
/*bufer -i setir -e elave edirik */
strcat(setir, bufer);
/*buferi yeniliiyirik*/
memset(bufer,0,20);
}
std::cout<<"\\n"<<setir<<"\\n";
}

```

Netice:

```

C:\cpp\prog2\Debug>./prog2.exe
Uzunluđu 20-dan kicik olan 6 setir daxil edin
aaa
bbb
cccc
dddddd
fffffffffff
ggggggg
aaabbbccccddddddfffffffffgggggg
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

### 3 - cü çalıřma.

Elə program tərrib edin ki, istifadəçidən 5 sətir qəbul etsin və bu sətirləri daxil olma sırasının əksi ardıcılığında birləřdirərək tam sətir kimi çap etsin.

```

// hell_7_3.cpp
#include <iostream>
#include <string.h>

/* bu programda biz elementleri setir olan cergelerden
istifade edeceyik */

int main(int argc, char *argv[])
{
char *setirler[5], bufer[20], *setir, butov_setir[1024];

memset(butov_setir,0,1024);

int i,k;
std::cout<<"Uzunluđu 20-dan kicik olan 5 setir daxil edin\\n";

for(i=0; i<5; ++i){
std::cin>>bufer;
k = strlen(bufer);
setirler[i] = new char[k];
strncpy(setirler[i],bufer,k);
}

for (i=4; i>=0; --i)
strcat(butov_setir,setirler[i]);

```

```
std::cout<<"\n"<<butov_setir<<"\n";  
}
```

Netice:

C:\cpp\prog2\Debug>

C:\cpp\prog2\Debug>./prog2.exe

Uzunlugu 20-dan kicik olan 5 setir daxil edin

qqq

www

eee

rrrrr

ttt

ttttrrrrreeeeewwwqqq

C:\cpp\prog2\Debug>

C:\cpp\prog2\Debug>

## \$8. Strukt tiplər

### 1 - ci çalışma.

Aşağıdakı işləri görən proqram tərtib edib icra edin.

int tipli x və 30 simvolla sətir tipli soz dəyişənlərindən ibarət olan str adlı yeni struct tipi yaradın. Bu yeni yaratdığınız tiptən str\_dey adlı dəyişən elan edin. Bu dəyişənin x və soz üzvlərinə müvafiq olaraq 10 və "proqramlashdirma" sözlərini mənimsədin. str\_dey dəyişəninin üzvlərinin qiymətlərini ekranda çap edin.

```
// hell_8_1.cpp
#include <iostream>
#include <string.h>

struct str
{
    int x;
    char soz[30];
};

int main(int argc, char *argv[])
{

    str str_dey;

    str_dey.x=10;
    strcpy(str_dey.soz, "proqramlashdirma");

    std::cout<<"str_dey -in heddleri \nx - "
    <<str_dey.x<<"\nsoz - " <<str_dey.soz<<"\n";
    return 0;
}
```

Netice:

```
C:\cpp\prog2\Debug>./prog2.exe
str_dey -in heddleri
x - 10
soz - proqramlashdirma
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

### 2 - ci çalışma.

Yuxarıdakı məsələdə str tipindən ünvan tipli str\_gst dəyişəni elan edin və məsələnin tələblərin yerinə yetirin.

```
// hell_8_2.cpp
#include <iostream>
#include <string.h>

struct str
{
    int x;
    char soz[30];
};

int main(int argc, char *argv[])
{
    str *str_dey;
    str_dey =new str;
```

```

str_dey->x=50;

memset(str_dey->soz,0,30);
strcpy(str_dey->soz,"proqramlashdirma");

std::cout<<"str_dey -in heddleri \nx - "
<<str_dey->x<<"\nsoz - " <<str_dey->soz<<"\n";
return 0;
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
str_dey -in heddleri
x - 50
soz - proqramlashdirma
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

### 3 - cü çalışma.

1 -ci çalışmada daxil olunan str tipli 5 elementdən ibarət strler cərgəsi elan edin. Bu cərgənin hər bir elementinin üzvlərinə istifadəçi tərəfindən daxil olunan qiymətlər mənimsədin. Daha sonra bu qiymətləri ekranda çap edin.

```

// hell_8_3.cpp
#include <iostream>
#include <string.h>

```

```

struct str
{
int x;
char soz[30];
};

int main(int argc, char *argv[])
{
str strler[5];
int i;
for(i=1; i<=5; ++i){
std::cout<<"strlerin " <<i
<<" -ci elementinin x ve soz heddlerini daxil edin\n";
std::cin>>strler[i].x>>strler[i].soz;
}

std::cout<<"strler -in elementlerinin x ve soz heddleri\n";

for(i=1; i<=5; ++i)
std::cout<<"x - " <<strler[i].x
<<"\nsoz - " <<strler[i].soz<<"\n";
return 0;
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
strlerin 1 -ci elementinin x ve soz heddlerini daxil edin
4 ferid

```

```

strlerin 2 -ci elementinin x ve soz heddlərini daxil edin
6 ramin
strlerin 3 -ci elementinin x ve soz heddlərini daxil edin
7 elif
strlerin 4 -ci elementinin x ve soz heddlərini daxil edin
56 mithat
strlerin 5 -ci elementinin x ve soz heddlərini daxil edin
23 taleh
strler -in elementlərinin x ve soz heddləri
x - 4, soz - ferid
x - 6, soz - ramin
x - 7, soz - elif
x - 56, soz - mithat
x - 5, soz - taleh
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

#### 4 - cü çalışma.

Çalışma 3-ü funksiyalardan istifadə etməklə həll edin. Bu məqsədlə 2 funksiya tərtib edin, daxil\_et və cap\_et . Müvafiq olaraq daxil\_et funksiyası istifadəçidən məlumatları oxuyub, strlər cərgəsinin elementlərinə mənimsədəcək, cap\_et isə strlər cərgəsinin elementlərinin qiymətlərini çap edəcək.

```

// hell_8_4.cpp
#include <iostream>
#include <string.h>

struct str
{
int x;
char soz[30];
};

void daxil_et( str strler[], int k);
void cap_et( str strler[], int k);

int main(int argc, char *argv[])
{
str strler[6];
int k=5;
daxil_et(strler,k);
cap_et(strler,k);

return 0;
}

/*=====*/
void daxil_et( str strler[], int k)
{
int i;
for(i=1; i<=k; ++i){
std::cout<<"strlerin "<<i
<<" -ci elementinin x ve soz heddlərini daxil edin\n";
std::cin>>strler[i].x>>strler[i].soz;
}
}

/*=====*/

```

```

void cap_et(struct str strler[], int k)
{
int i;
std::cout<<"strler -in elementlerinin x ve soz heddleri\n";
for(i=1; i<=k; ++i)
std::cout<<"x - "<<strler[i].x<<"\nsoz - "<<strler[i].soz<<"\n";
}

```

Netice:

```

C:\cpp\prog2\Debug>./prog2.exe
strlerin 1 -ci elementinin x ve soz heddlerini daxil edin
1 alma
strlerin 2 -ci elementinin x ve soz heddlerini daxil edin
2 heyva
strlerin 3 -ci elementinin x ve soz heddlerini daxil edin
3 nar
strlerin 4 -ci elementinin x ve soz heddlerini daxil edin
4 gilasi
strlerin 5 -ci elementinin x ve soz heddlerini daxil edin
5 erik
strler -in elementlerinin x ve soz heddleri
x - 1, soz - alma
x - 2, soz - heyva
x - 3, soz - nar
x - 4, soz - gilasi
x - 5, soz - erik
C:\cpp\prog2\Debug>

```

## 5 - ci çalışma

Funksiyalardan istifadə etməklə elə proqram qurun ki, çalışma 1-də daxil olunan str tipli 5 elementdən ibarət strler cərgəsi elan etsin. daxil\_et funksiyası vastəsilə istifadəçidən oxunan qiymətləri bu cərgəsinin elementlərinə mənimsətsin. Daha sonra max\_el funksiyası tərtib edin, hansı ki, strler cərgəsinin elementləri arasında x-i ən böyük olanın qiymətlerini ( x və soz ) çap etsin.

// hell\_8\_5.cpp

```

#include <iostream>
#include <string.h>

```

```

struct str
{
int x;
char soz[30];
};

```

```

void daxil_et( str strler[], int k);
str max_el( str strler[], int k);

```

```

int main(int argc, char *argv[])
{
str strler[6], elem;
int k=6;

```

```

daxil_et(strler,k);
elem = max_el(strler,k);

```

```

std::cout<<"max x, soz: "<<elem.x<<" "<<elem.soz<<"\n";

```

```

return 0;
}

/*=====*/
void daxil_et( str strler[], int k)
{
int i;
for(i=1; i<=k; ++i){
std::cout<<"strlerin "<<i
<<" -ci elementinin x ve soz heddlerini daxil edin\n";
std::cin>>strler[i].x>>strler[i].soz;
}
}

/*=====*/
struct str max_el(struct str strler[], int k)
{
int i,j=0,dey=strler[0].x;
for(i=1; i<k; ++i){
if (strler[i].x > dey ){
j = i;
dey = strler[i].x;
}
}
return strler[j];
}

```

Netice:

```

C:\cpp\prog2\Debug>./prog2.exe
strlerin 1 -ci elementinin x ve soz heddlerini daxil edin
45 baki
strlerin 2 -ci elementinin x ve soz heddlerini daxil edin
34 london
strlerin 3 -ci elementinin x ve soz heddlerini daxil edin
456 texas
strlerin 4 -ci elementinin x ve soz heddlerini daxil edin
78 istanbul
strlerin 5 -ci elementinin x ve soz heddlerini daxil edin
12 ankara
max x, soz: 456, texas
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```



## \$9. Siyahılar

### 1 - ci çalışma.

1. Program 2 nümunəsini elə dəyişin ki, siyahının obyektleri özündə int tipli x həddindən əlavə char tipli ad[20] həddi də saxlasın. Müvafiq olaraq siyahı yaratma və cap\_et funksiyalarında lazımı dəyişikliyi edin. Programı kompilyasiya və icra edin, siyahı yaradin , onun elementlərini çap edin.

```
// hell_9_1.cpp
#include <iostream>
#include <string.h>

struct syh_el
{
    int x;
    char ad[20];
    syh_el *novb_el;};

syh_el *siyahı_yarat(syh_el *syh, int elem_say);
void siyahını_cap_et( syh_el *);

int main(int argc, char *argv[])
{
    /* istifade edəcəyimiz deyishenleri elan edirik */
    syh_el *menim_syh;
    /* siyahinin bosh oldugunu bildirmek ucun */
    benim_syh = NULL;
    int say;

    std::cout<<"Siyahinin elementlerinin sayini daxil edin \n";
    std::cin>>say;

    benim_syh=siyahı_yarat(menim_syh,say);
    siyahını_cap_et(menim_syh);

    return 0;
}

//-----
syh_el *siyahı_yarat( syh_el *syh, int elem_say)
{
    syh_el *p, *q;

    p=syh;
    q=syh;
    int i,dey;
    char bufer[30];

    for (i=1; i<=elem_say; ++i){
        std::cout<<"siyahinin "<<i
        <<" -ci elementinin x ve ad heddlerini daxil edin \n";
        memset(bufer,0,30);
        std::cin>>dey>>bufer;
        p = new syh_el;
        p->x = dey;
        strncpy(p->ad,bufer,20);
```

```

*(p->ad + 20) = '\0';
p->novb_el = NULL;

if (syh==NULL){
syh=p;
q = p;
p = NULL; }
else {
q->novb_el = p;
q = p;
}
}
return syh;
}
//-----
void siyahini_cap_et(syh_el *syh)
{
syh_el *p;
int dey, fix = 0;
char bufer[30];
p = syh;
if (syh == NULL ) {
std::cout<<"Siyahi boshdur \n";
return;
}
std::cout<<"Siyahinin elementleri \n";
while(p!=NULL)
{
/*elementlerin capinin ekanda gozel gorunmesi ucun*/
if (fix++ != 0) std::cout<<" -> ";
dey = p->x;
memset(bufer,0,30);
strncpy(bufer, p->ad, 20);
*(bufer + 20) = '\0';
std::cout<<"(0<<"dey<<" "<<bufer<<")";
p = p->novb_el; /* novbeti elemente kec */
}
std::cout<<"\n";
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Siyahinin elementlerinin sayini daxil edin
4
siyahinin 1-ci elementinin x ve ad heddlerini daxil edin
12 kitab
siyahinin 2 -ci elementinin x ve ad heddlerini daxil edin
34 defter
siyahinin 3 -ci elementinin x ve ad heddlerini daxil edin
456 idman
siyahinin 4 -ci elementinin x ve ad heddlerini daxil edin
124 hefte
Siyahinin elementleri
(12 kitab) -> (34 defter) -> (456 idman) -> (124 hefte)
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## 2 - ci çalışma

Yuxarıdakı proqrama axtarış funksiyası əlavə edin. Proqram istifadəçidən int tipli ədəd istəyir və siyahıda x -i bu ədədə bərabər olan obyektin ad[20] -həddini çap edir.

```
// hell_9_2.cpp
#include <iostream>
#include <string.h>

struct syh_el
{
    int x;
    char ad[20];
    syh_el *novb_el;};

syh_el *siyahi_yarat(syh_el *syh, int elem_say);
syh_el * axtarish(syh_el *, int);

int main(int argc, char *argv[])
{
    /* istifade edeceyimiz deyishenleri ilan edirik */
    syh_el *menim_syh, *syh_dey = NULL;
    int x;
    char bufer[30];
    memset(bufer,0,30);
    /* siyahinin bosh oldugunu bildirmek ucun */
    benim_syh = NULL;
    int say;

    std::cout<<"Siyahinin elementlerinin sayini daxil edin \n";
    std::cin>>say;

    benim_syh=siyahi_yarat(menim_syh,say);

    std::cout<<"Siyahidan ad heddini tapmaq istediyyiniz "
        <<"elementin x -ni daxil edin \n";
    std::cin>>x;

    syh_dey = axtarish(menim_syh,x);

    strcpy(bufer,syh_dey->ad);
    if (syh_dey!=NULL)
    std::cout<<"x heddi "
        <<x<<" -ye beraber olan elementin ad heddi "
        <<bufer<<" -dir\n";

    return 0;
}

//~~~~~
syh_el *siyahi_yarat( syh_el *syh, int elem_say)
{
    syh_el *p, *q;
    p=syh;
    q=syh;
    int i,dey;
    char bufer[30];
    for (i=1; i<=elem_say; ++i)
    {
```

```

std::cout<<"siyahinin "<<i
<<" -ci elementinin x ve ad heddlerini daxil edin \n";
memset(bufer,0,30);
std::cin>>dey>>bufer;
p = new syh_el;
p->x = dey;
strncpy(p->ad,bufer,20);
*(p->ad + 20) = '\0';
p->novb_el = NULL;
if (syh==NULL){
syh=p;
q = p;
p = NULL; }
else {
q->novb_el = p;
q = p;
}
}
return syh;
}
//-----
syh_el * axtarish(syh_el *syh, int tap)
{
syh_el *p;
p = syh;
if (syh == NULL ) {
return NULL;
}
while(p!=NULL){
if (p->x==tap) return p;
p = p->novb_el; /* novbeti elemente kec */
}
return NULL;
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Siyahinin elementlerinin sayini daxil edin
6
siyahinin 1-ci elementinin x ve ad heddlerini daxil edin
123 kitab
siyahinin 2 -ci elementinin x ve ad heddlerini daxil edin
34 musiqi
siyahinin 3 -ci elementinin x ve ad heddlerini daxil edin
56 ehtimal
siyahinin 4 -ci elementinin x ve ad heddlerini daxil edin
79 shebeke
siyahinin 5 -ci elementinin x ve ad heddlerini daxil edin
875 prosessor
siyahinin 6 -ci elementinin x ve ad heddlerini daxil edin
257 fizika
Siyahidan tapmaq istediyyiniz elementin x -ni daxil edin
79
x heddi 79 -ye beraber elementin ad heddi shebeke -dir
C:\cpp\prog2\Debug>

```

### 3 - cü çalışma

İki siyahını birləşdirən funksiya tərtib edin. Funksiya parametr olaraq iki siyahı qəbul edir, daha sonra bu siyahılardan birincini ikinci ilə birləşdirir.

```
// hell_9_3.cpp
#include <iostream>
#include <string.h>

struct syh_el
{
    int x;
    char ad[20];
    syh_el *novb_el;};

syh_el *siyahı_yarat(syh_el *syh, int elem_say);
void birləshdir(syh_el *, syh_el *);
void siyahını_cap_et(syh_el *);

int main(int argc, char *argv[])
{
    /* istifade edəcəyimiz deyishenleri ilan edirik */
    syh_el *syh1, *syh2 = NULL;
    int x;
    char bufer[30];
    memset(bufer,0,30);
    /* siyahıların bosh olduğunu bildirmek ucun */
    syh1 = syh2 = NULL;
    int say;

    std::cout<<"Birinci siyahının elementlerinin sayini"
              <<" daxil edin \n";
    std::cin>>say;
    syh1=siyahı_yarat(syh1,say);

    std::cout<<"ikinci siyahının elementlerinin sayini "
              <<"daxil edin \n";
    std::cin>>say;
    syh2=siyahı_yarat(syh2,say);

    /*syh1 -i syh2 ilebirləshdiririk*/
    birləshdir(syh1,syh2);
    siyahını_cap_et(syh1);

    return 0;
}

//~~~~~
syh_el *siyahı_yarat( syh_el *syh, int elem_say)
{
    syh_el *p, *q;
    p=syh;
    q=syh;
    int i,dey;
    char bufer[30];

    for (i=1; i<=elem_say; ++i){
        std::cout<<"siyahının "<<i
        <<" -ci elementinin x ve ad heddlerini daxil edin \n";
```

```

memset(bufer,0,30);
std::cin>>dey>>bufer;
p = new syh_el;
p->x = dey;
strncpy(p->ad,bufer,20);
*(p->ad + 20) = '\0';
p->novb_el = NULL;

if (syh==NULL){
syh=p;
q = p;
p = NULL; }
else {
q->novb_el = p;
q = p;
}
}
return syh;
}

//-----
void birleshtir(syh_el *syh1, syh_el *syh2)
{
if (syh1 == NULL ) {
syh1 = syh2;
return;
}
if (syh2 == NULL)
return;
syh_el *p;
p = syh1;
while(p->novb_el!=NULL)
p = p->novb_el;
p->novb_el = syh2;
}

//-----
void siyahini_cap_et(syh_el *syh)
{
syh_el *p;
int dey, fix = 0;
char bufer[30];
p = syh;
if (syh == NULL ) {
std::cout<<"Siyahi boshdur \n";
return;
}
std::cout<<"Siyahinin elementleri \n";

while(p!=NULL)
{
/*elementlerin capinin ekanda gozel gorunmesi ucun*/
if (fix++ != 0) std::cout<<"->";
dey = p->x;
memset(bufer,0,30);
strncpy(bufer, p->ad, 20);
*(bufer + 20) = '\0';
std::cout<<"("<<dey<<" "<<bufer<<")";

```

```

p = p->novb_el; /* novbeti elemente kec */
}
std::cout<<"\n";
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Birinci siyahinin elementlerinin sayini daxil edin
3
siyahinin 1-ci elementinin x ve ad heddlərini daxil edin
12 vaxt
siyahinin 2 -ci elementinin x ve ad heddlərini daxil edin
34 saniye
siyahinin 3 -ci elementinin x ve ad heddlərini daxil edin
45 elifba
Ikinci siyahinin elementlerinin sayini daxil edin
2
siyahinin 1-ci elementinin x ve ad heddlərini daxil edin
56 telim
siyahinin 2 -ci elementinin x ve ad heddlərini daxil edin
78 terbiye
Siyahinin elementleri
(12 vaxt)->(34 saniye)->(45 elifba)->(56 telim)->(78 terbiye)
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

#### 4 - cü çalışma

Nümunə Program 3 -də daxil edilmiş silmə funksiyasının elə dəyişdirin ki, siyahının verilmiş indeksli obyektindən başlayaraq verilmiş sayda obyektı siyahıdan silsin.

```

// hell_9_4.cpp
#include <iostream>
#include <string.h>
struct syh_el
{
int x;
char ad[20];
syh_el *novb_el;
};
syh_el *siyahi_yarat(syh_el *syh, int elem_say);
syh_el *siyahi_sil(syh_el *syh, int elem, int say);
void siyahini_cap_et(syh_el *);

```

```

int main(int argc, char *argv[])
{
/* istifade edəcəyimiz deyishenleri ilan edirik */
syh_el *menim_syh;
/* siyahinin bosh oldugunu bildirmek ucun */
menim_syh = NULL;
int say,elem;

std::cout<<"Siyahinin elementlerinin sayini daxil edin \n";
std::cin>>say;

```

```

menim_syh=siyahi_yarat(menim_syh,say);
siyahini_cap_et(menim_syh);

std::cout<<"Siyahidan silmek istediyyiniz elementlerin \n"
    <<"bashlangic indeksini ve sayini daxil edin\n";
std::cin>>elem>>say;

menim_syh = siyahi_sil(menim_syh, elem,say);
siyahini_cap_et(menim_syh);

return 0;
}

//-----
syh_el *siyahi_yarat( syh_el *syh, int elem_say)
{
syh_el *p, *q;
p=syh;
q=syh;
int i,dey;
char bufer[30];

for (i=1; i<=elem_say; ++i)
{
std::cout<<"siyahinin "<<i
<<" -ci elementinin x ve ad heddlerini daxil edin \n";
memset(bufer,0,30);

std::cin>>dey>>bufer;
p = new syh_el;
p->x = dey;
strncpy(p->ad,bufer,20);
*(p->ad + 20) = '\0';
p->novb_el = NULL;
if (syh==NULL){
syh=p;
q = p;
p = NULL; }
else {
q->novb_el = p;
q = p;
}
}
return syh;
}

//=====
void siyahini_cap_et (syh_el *syh)
{
syh_el *p;
int dey, fix = 0;
char bufer[30];
p = syh;
if (syh == NULL ) {
std::cout<<"Siyahi boshdur \n";
return;
}
std::cout<<"Siyahinin elementleri \n";

```



```

while(p!=NULL)
{
/*elementlerin capinin ekanda gozel gorunmesi ucun*/
if (fix++ != 0) std::cout<<" -> ";
dey = p->x;
memset(bufer,0,30);
strncpy(bufer, p->ad, 20);
*(bufer + 20) = '\0';
std::cout<<"("<<dey<<" "<<bufer<<")";
p = p->novb_el; /* novbeti elemente kec */
}
std::cout<<"\n";
}

//=====
syh_el *siyahi_sil(syh_el *syh, int elem, int say)
{
syh_el *p, *q;
p=syh;
int i,dey;

if (syh==NULL)
return NULL; /* siyahi boshdur */
if (elem==1)
{
/* silmek istediyyimiz element ilk elementdir */
syh = p->novb_el;
p->novb_el = NULL;
delete p;
return syh;
}

for (i=1; i<elem-1; ++i)
if (p==NULL) break;
else
p = p->novb_el;
if (p==NULL){
std::cout<<"Siyahida "
<<elem<<"sayda element movcud deyil\n";
return syh;
}
q=p;
/*
indi q simek istediyyimiz yerin bashlangicina istinad edir
p -ni bu elementden bashlayaraq say qeder sona
surushdurmeliyyik
*/
for (i=0; i<say; ++i)
if (p==NULL) break;
else
p = p->novb_el;
if (p==NULL){
std::cout<<"Siyahida "
<<elem<<"sayda element movcud deyil\n";
return syh;
}

if (p->novb_el==NULL)

```

```

{
/* siyahini q-dən başlayaraq sona kimi silmeliyik */
q->novb_el=NULL;
delete p;
p=q=NULL;
return syh;
}

q->novb_el = p->novb_el;
p->novb_el = NULL;
delete p;
q=NULL;
return syh;
}

```

Netice:

```

C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Siyahinin elementlerinin sayini daxil edin
9
siyahinin 1-ci elementinin x heddini daxil edin
1
siyahinin 2-ci elementinin x heddini daxil edin
2
siyahinin 3-ci elementinin x heddini daxil edin
3
siyahinin 4-ci elementinin x heddini daxil edin
4
siyahinin 5-ci elementinin x heddini daxil edin
5
siyahinin 6-ci elementinin x heddini daxil edin
6
siyahinin 7-ci elementinin x heddini daxil edin
7
siyahinin 8-ci elementinin x heddini daxil edin
8
siyahinin 9-ci elementinin x heddini daxil edin
9
Siyahinin elementleri
(1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9)
Siyahidan silmek istediyyiniz elementlerin
bashlangic indeksini ve sayini daxil edin
3 4
Siyahinin elementleri
(1) -> (2) -> (7) -> (8) -> (9)
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>

```

## Qeydlər

Oxucular üçün:

Bu buraxılışda əsasən sintaktik səhvlərin aradan qaldırılması və bəzi yerlərdə proqram kodunun və izahın əlavə olunmasına üstünlük verdik.

Çox istədik ki, Obyekt yönümlü proqramlaşdırmanın Varislik və Polimorfizm kimi mövzuların əlavə edək, amma bu buraxılışda buna imkanımız olmadı.

Qurucular üçün:

Kitabın gələcək buraxılışlarının hazırlanmasında , kitabda olan mövzuların təkmilləşdirilməsində, yeni mövzuların və proqram nümunələrinin əlavə olunmasında könüllü köməklik göstərmək istəyən proqramçılar müəlliflə aşağıdakı ünvandan əlaqə saxlaya bilər.

[ahmed.sadikhov@gmail.com](mailto:ahmed.sadikhov@gmail.com)

Görülməli işlər:

1. Continue və break operatorlarına aid proqram nümunəsi, izahın yenilənməsi.
2. Müqaisə əməllərinin 3.1 -dən əvvələ keçirilməsi + proqram kodu.
3. Rekursiv funksiyaların izahı və proqram kodu nümunəsi
4. Bütün nəticələrdə proqram adları prog2.exe kimi göstərilib, müvafiq adlarla dəyişmək.
5. Ünvan dəyişənləri üzərində hesab əməlləri + proqram kodu.
6. Null qiymətinin izahı + proqram kodu (\$4). Hazırki buraxılışda NULL barəsində ancaq \$9 -da qısa izah verilir.
7. Varislik mövzusunun əlavə olunması
8. Polimorfizm mövzusunun əlavə olunması
9. Axınlar (streams) mövzusunun əlavə olunması
10. İkiistiqamətli siyahılar, ağaclar, növbələr, steklər mövzularının əlavə olunması
11. Const, statik, volatile, external dəyişən və funksiyalara aid mövzunun əlavə olunması + proqram kodu
12. Fayllarla işləmə funksiyalarına aid mövzunun əlavə olunması
13. Proqram nümunələrinin artırılması, yeni çalışmaların əlavə olunması