C++ proqramlaşdırma dili

Birinci buraxılış

21.02.2011/Bakı,Azərbaycan

Mətndə verilən məlumatların sizin hansısa bir işinizə yarayacağına və ya burada verilmiş proqramlardan istifadə nəticəsində sizə dəyəcək hər hansı ziyana görə müəllif heç bir öhdəlik götürmür.

Free Software Promotion/Azerbaijan.

Mündəricat

$1 Giriş.

Bu mətndə Windows sistemlərində 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 çalışmalar mütləq yerinə yetirilməlidir. Bu sizə materialı daha da aydın başa düşməyə kömək etməklə yanaşı, sizdə gələcək inkişaf üçün əvəzedilməz 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 sistemləri üçün nəzərdə tutulub.

1.2 Kompilyatorun installyasiyası

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 Ilk 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.

```
// prog1.cpp : Defines the entry point for the console
//    application.

#include "stdafx.h"

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

Bu mətn kompiyator tərəfindən avtomatik yaradılıb.

Proqramın mətn faylında aşağıdakı kimi dəyişikliklər edirik.

`#include "stdafx.h"` sətrindən sonra `#include <iostream> ,`

`{`

mötərəzəsindən sonra isə

```
     std::cout<<"Salam dunya \n"; ,
```

`return 0;` sətrindən əvvəl isə

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

sətrlərini daxil edirik.

Aşağıdakı kimi:

```
// prog1.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 **Output** pəncərəsinə ötürüləcək.

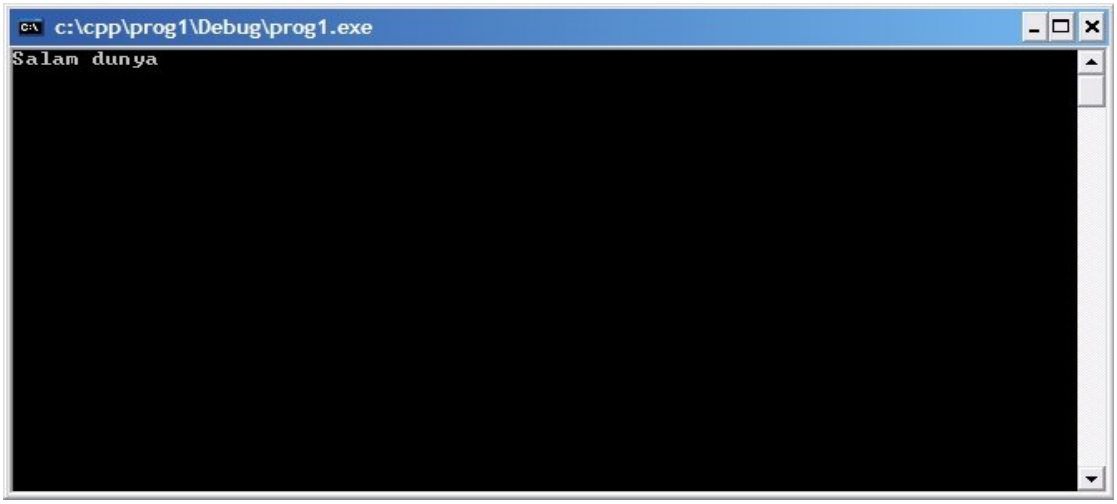
Əgər sonda **Build: 1 Succeeded ...** sətri çap olunursa deməli proqramımız uğurla kompilyasiya olunmuşdur.

İndi isə proqramımızı yerinə yetirək.

Bunun üçün **Debug -> Start Debugging** düyməsini basırıq.

Nəticədə kansole pəncərəsi açılacaq və Salam dunya mətni çap olunacaq.

Proqramı söndürmək üçün klaviaturadan hər-hansı simvol daxil edib enter düyməsinə basmağımız kifayətdir.



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

Əgər bu nəticəni almamısınızsa 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 dunya \n";` sətrində verilən Salam dunya ifadəsini ekranda çap edir.

Əgər Salam dunya əvəzinə istənilən digər ifadə yazsanız onda ekranda həmin ifadə çap olunar.

Proqramın yerdə qalan detalları barədə irəlidə müvafiq bölmələrlə tanış olduqdan sonra məlumat veriləcək.

Hələlik isə onu deyə bilərik ki, `#include<iostream>` sətri istifadə edəcəyimiz bütün proqramlarda daxil olunur.

Proqrama bu sətri daxil etmək bizə `std::cout, std::cin` funksiyalarından istifadə etməyə imkan verir.
Növbəti sətir `int _tmain(int argc, _TCHAR* argv[])` sətridir.
Bu proqramın əsas funksiyasıdır.

C++ silində yazılmış proqramlarda müxtəlif işlər görən standart(std::cout) və proqramçı tərəfindən yaradılan funksiyalardan ($5) istifadə olunur.

main funksiyası (_tmain unikodu dəstəkləmək üçün MS Vstudionun əlavəsidir)isə xüsusi funksiyadır. Bütün C++ proqramlarında bu funksiya mütləq olmalıdır və bütün C++ proqramları icraolunmağa main funksiyasından başlayır.

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

Ən sonda yerləşən } simvolu isə main funksiyasının bitdiyini göstərir.

Çalışmalar:

1. prog1.cpp proqramında `std::cout<<"Salam dunya \n";` sətrindən sonra `std::cout<<"Hey bu yeni proqramdir \n";` sətrini yerləşdirin, faylı yadda saxlayın, kompilyasiya və icra edin.

2. C++ dilində Mən C++ dilini örgənirəm sətrini 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 yaddaşda 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.

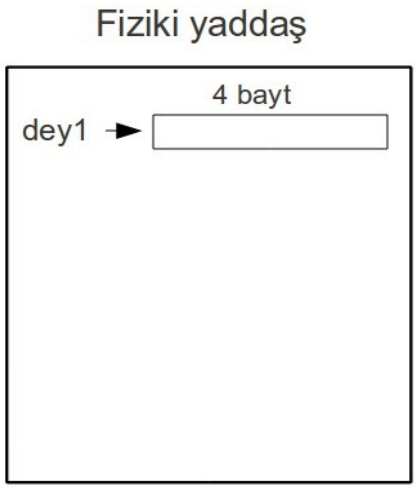
Bu yaddaşın sahəsinin həcmini və adını dəyişəni elan edərkən biz özümüz(proqramçılar) müəyyən edirik.

Misal üçün aşağıdakı kimi:

```
int dey1;
```

Yuxarıdakı kod hissəsində biz fiziki yaddaşda (RAM) 4 bayt yer ayırırıq və həmin yerə `dey1` adını verdik. Artıq bundan sonra proqramda `dey1` üzərində apardığımız bütün əməliyyatlar birbaşa `dey1` - ə aid yaddaş sahəsi üzərində aparılacaq(şəkil1).

```
int dey1;
```



şəkil 1

Yadda saxladığı məlumatın növünə və yaddaşda 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 tip, struktur tiplər, siyahılar və klasslardan da çox geniş istifadə olunur ki, bunlarla da uyğun olaraq 4, 8, 9, 10-cu paraqraflarda məşğul olacayıq.

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 olacayıq.

Beləliklə cəm proqrmında 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 adlandıraq.

Bu dəyişənləri təyin etmək üçün proqram kodu aşağıdakı kimi olacaq.

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

Qeyd edək ki, eyni tipdən 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 məna bildirir).

Proqramımız belə olacaq:

```
// prog1.cpp : Defines the entry point for the console application.
#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;
}
```

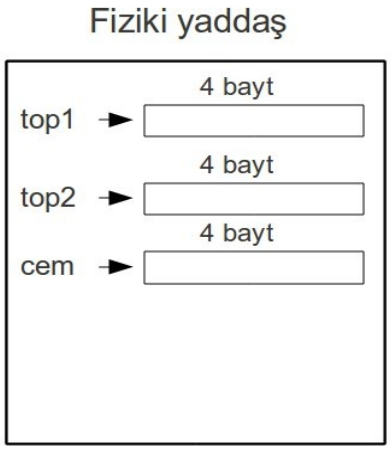
Proqramı yadda saxlayaq , kompilyasiya edək və yerinə yetirək.

Proqramın izahı

Program icra olunmağa `int _tmain(int argc, _TCHAR* argv[])` sətrindən başlayır.

Daha sonra proqramda `int top1, top2, cem;` sətri gəlir. Burada biz `int` tipli `top1, top2` və `cem` dəyişənlərini elan edirik.

Yaddaşın vəziyyəti bu zaman aşağıdakı kimi olar(şəkil 2).



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.

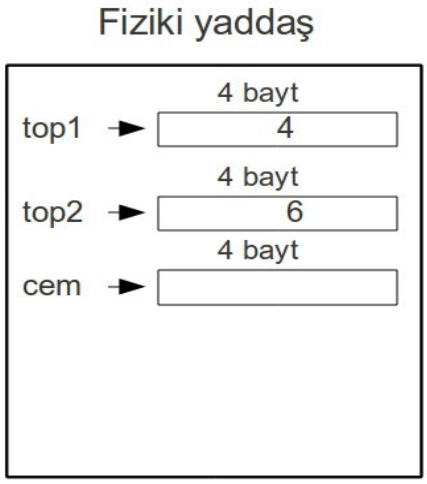
Yuxarıdakı kod sətrində mənimsətmə operatorundan solda `top1` dəyişəni, sağda isə `4` qiyməti durur.

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 vardısa o silinir.

Uyğun olaraq `top2 = 6;` proqram sətri də yaddaşın `top2` - yə aid olan hissəsinə `6` qiymətini yazır.

Bu zaman yaddaşın vəziyyəti aşağıdakı kimi olar:



şəkil 3

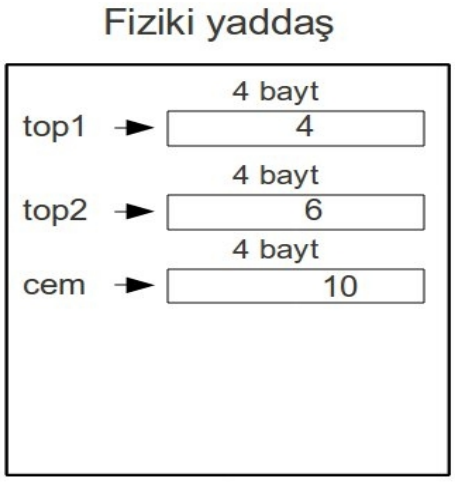
Proqramın növbəti sətrinə keçək:

```
cem = top1 + top2;
```

Burada biz yenə də = mənimsətmə opratorundan istifadə edərik.

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 (şəkil 4).



şəkil 4

Proqramın növbəti sətri belədir:

```
std::cout<<" 4 ile 6 -nin cemi ="<<x;
```

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 danışacayıq.

`std::cout` funksiyasının geniş izahı Əlavə A -da verilib.

Proqram nümunələri:

1. 3 ədədin cəmin 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;
```

Indi isə bu 3 dəyişənin cəmini `h` dəyişəninə mənimsədək.

```
h = x + y + z;
```

Nəticəni çapa verək:

```
std::cout<<x<<" , "<<y<<" , "<<" , "<<z<<"ededlerinin cemi ="<<h;
```

Proqram aşağıdakı kimi olar:

```
// prog1.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include <iostream>

int main(int argc, _TCHAR* argv[])
{
int x,y,z,h;
x=8;
y=45;
z=3;
h = x + y + z;
std::cout<<x<<" , "<<y<<" , "<<" , "<<z<<"ededlerinin cemi ="<<h;

int x;
std::cin>>x;
return 0;
}
```

2.2 Dəyişənlərin elanı qaydaları.

C++ dilində dəyişənlər proqramın istənilən yerində elan oluna bilər. Dəyişənlərə müraciət isə yalnız dəyişənlərin elanından aşağı hissədə edilə bilər.

Eyni tipdən olan bir neçə dəyişən ayrı-ayrı sətirlərdə və ya eyni elan sətrində elan oluna bilər.

Aşağıdakı kimi:

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

və ya

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

Dəyişənlərə birbaşa elan sətrində 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 ifdədə onun öz qiymətindən istifadə olunur.

Proqramlaşdırma nöqteyi 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 azaltmq üçün istifdə olunur.

Bunlar uyğun olaraq aşağıdakılardır:

İnkrement - artırma `++`, Dekremetn azaltma `--` .

Misal üçün inkrementdən istifdə edərək `x` -in qiymətin `1` vahid artırmaq istəsək aşağıdakı kimi yaza 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 dəyişənin qiymətini 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.

Indi isə biraz daha maraqlı, *interaktiv* proqramlar tərtibi ilə məşğul olaq.

Baxdığımız proqramlarda biz hesab əməlləri üçün statik qiymətlərdən istifadə etdik, indi isə elə proqramlar yazaq ki, dəyişənlərin qiymətlərinin istifadəçi tərəfindən daxil edilməsi mümkün olsun.

C++ dilində isitfadəçinin daxil etdiyi məlumatı proqramdakı dəyişənlərə mənimsətmək üçün əsasən `std::cin` funksiyasından istifadə olunur.

`std::cin` funksiyası barədə Əlavə A -da geniş izh 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 sətri 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 qovluğu **cpp** adlandıraq.

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;
}
```

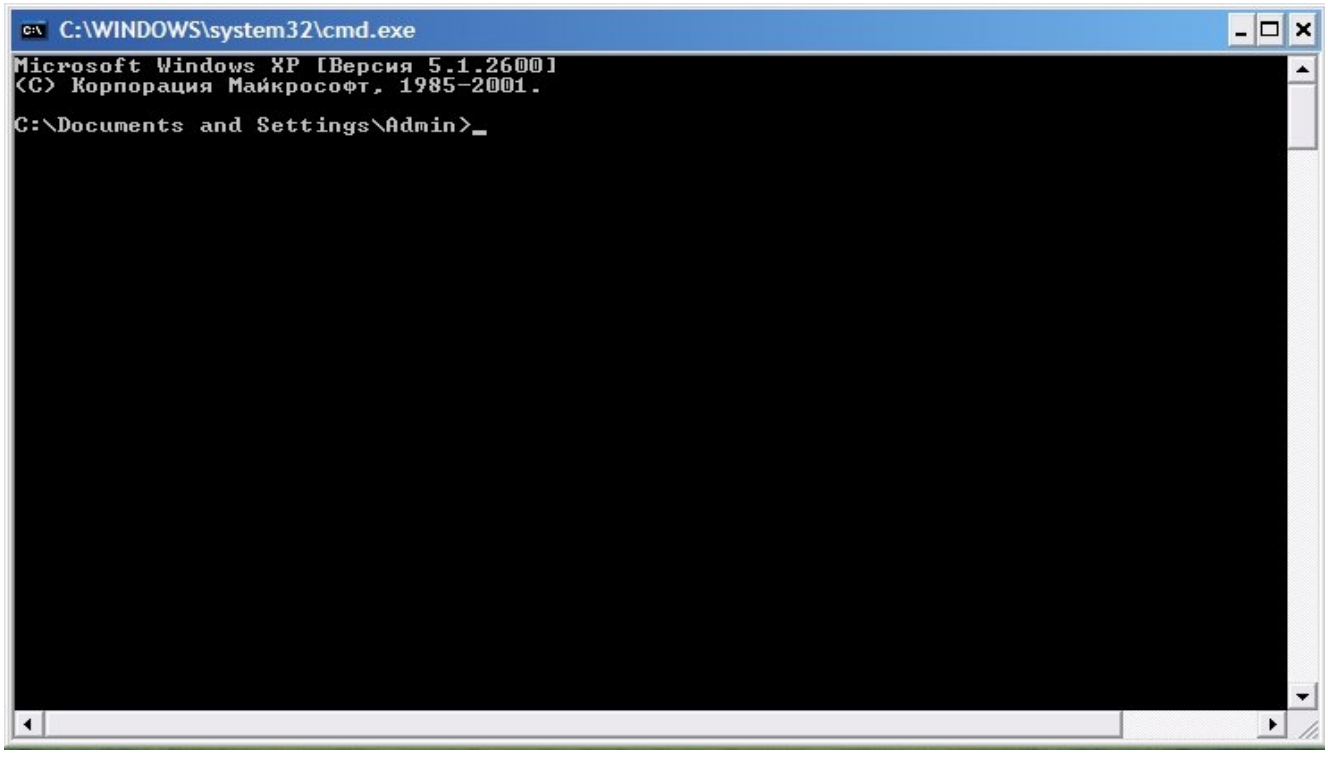
Yadda saxlayıb, kompilyasiya edirik.

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\Degub
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:

```
// prog2.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
int x,y;
std::cout<<"Zehmet olmasa her hansi 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 (F7).

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Zehmet olmasa her hansi 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` finksiyası 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, istifadıçidən 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:

```
#include <iostream>

int main(){
int en, uz, sahe;

std::cout<<"Zəhmət olmasa düzbucaqlının enini daxil edin \n";
std::cin>>en;

std::cout<<"Zəhmət olmasa düzbucaqlının uzunluğunu daxil edin \n";
std::cin>>uz;

sahe = en*uz;
std::cout<<"Düzbucaqlının 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:

Proqram nümunəsi.

Aşağıdakı proqramı kompilyasiya edib yerinə yetirin:

```
#include <iostream>

int main(){
char x,y;
x = 'a';
y = 'B';

std::cout<<"x ve y -in qiymetleri: "<<x<<" , "<<y<<"\n";
return 0;
}
```

Proqramı 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 saviyyədə ədədlərlə ifadə olunur.

Sadəcə onlaraq ədədlərə müxtəlif tipdən olan dəyişən kimi yanaşmaq bizə istifadə rahatlığı yaradır.

Başqa sözlə char tipindən olan dəyişəni ədədə mənimsədə bilərik .

Yuxarıda baxdığımız proqramın bir qədər dəyişilmiş variantına baxaq:

```
#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;
}
```

Proqramı 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 proqramda 'x' , 'y' yazırıqsa kompiyator bunu ingilis əlifbasının müvafiq simvolları 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 ASCİİ simvollar cədvəlidir.

ASCİİ cədvəli 128 simvoldan ibarətdir. Onların siyahısı və ədəd qarşılığı Əlavə B -də göstərilib.

Bu siyahıdan olan bəzi ASCİİ simvolları daha çox istifadə olunur.

Bunlar aşağıdakılardır:

`'\n'` yeni sətir simvolu.

`'\t'` tabulyasiya simvolu.

`'\0'` sətrin sonunu bildirmək üçün istifadə olunan somvol.

2.5 Proqramda Şərhlər

C++ dilində yazılmış hər – hansı proqramı şərhsiz təsəvvür 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 şəhrlərin heç bir aidiyyəti yoxdur. Belə ki, kompilyator proqramı kompilyasiya edərkən birinci gordüyü iş şərhləri proqram kodundan silməkdir.

C++ dilində 2 cür şərhlərdən istifadə olunur: çoxsətirli - `/*` **və** `*/` və təksətirli - `//` .

Çoxsətirli şərhlərdən istifadə etdikdə kompilyator `/*` və `*/` arasında qalan bütün proqram kodun şərh kimi qəbul edəcək .

Təksətirli şərhlərdən istifadə edən zaman kompilyator `//` simvollarından həmin sətrin 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 izə 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 şərhlərdən 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;
```

Indi tutaq ki, mən istəyirəm `z` -ə `x` -in `y` -ə nisbətin mənimsədim.
Yəni `x` -i `y` -ə böləndə alınan tam ədədə.

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;
```

Proqram nümunəsi.

Aşağıdakı proqramı kompilyasiya edib yerinə yetirin:

```
#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 paraqrafda biz proqramlaşdırmanın əsas elementlərindən biri - DƏYİŞƏNLƏR , onların tipləri və dəyişənlər üzərində hesab əməlləri ilə tanış olduq.

Bu paraqrafda isə biz C++ dilində yazılmış proqların digər əsas elementləri - OPERATORLAR ilə tanış olacayıq.

OPERATORLAR proqramlaşdırmada hər hansı şərtdən asılı olaraq proqramın növbəti icra istiqamətini müəyyənləşdirir.

Misal üçün hər hansı şərtdən asılı olaraq bu və ya digər kod hissəsi icra olunar, hər hansı kod hissəsi bir neçə dəfə təkrar olunar və ya bir neçə kod hissəsindən biri icra olunar.

3.1 Şərt operatorları

Biz 3 qrup operatorlarla tanış olacağıq : Şərt, Dövr, Seçim.

Şərt opretaru çox sadədir. Əvvəlki proqram nümunələrində biz müxtəlif proqram kodları icra etdik.

Misal üçün:

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

kod hissəsi ilə biz `x` -in qiymətini ekranda çap edirdik və bu zaman heç bir şərtdən istifadə etmirdik.

Şərt opratoru bizə bu imkanı verir, yəni biz istədiyimiz şərtdən asılı olaraq `x` -in qiymətini çap edərik və ya heç bir iş görmərik.

Tutaq ki, məsələ belədir:

Əgər `x` -in qiyməti `234` -dən böyükdüsə 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  əməliyatlar
  }
 else {
   digər əməliyatlar   }
```

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

Izahı:

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

Digər misal:

```
  int x;
    if (x<5)
std::cout<<x<<"  5 -den kiçikdir "<<"\n";
    else
std::cout<<x<<"  5 -den boyukdur "<<"\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:

```
#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>
```

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 sintakisisi aşagıdakı kimidir:

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

Nümunə proqram:

```
#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;
}
```

programı 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>
C:\cpp\prog2\Debug>
```

Tapşırıq:

Yuxarıdakı proqramda `for (k=0; k<10; k=k+1)` sərtini `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 ödənir dövrdə 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 ödənir, belə ki, `0 < 10` ifadəsi doğrudur və avtomatik olaraq dövr opratorunun əməliyatları icra olunur(ekranda “salam dunya” sətri çap olunur).

Daha sonra növbə sayğacın qiymətinin dəyişməsinə gəlir.

Bunun üçün biz `for` -da `k = k + 1;` yazmışıq.

Bu əməliyyat `k` -ın əvvəlki qiyməti nə idisə onu `1` vahid artırır.

Baxdığımız hal üçün (`k` -nın qiyməti `0` olan hal) `k` -nın 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 opratoru

`for` operatorunda biz sayğac təyin etdik, dövrün başa çatması şərtini və sayğacın dəyişmə qaydasını verdik. Bu zaman biz dövrün neçə dəfə təkrar olunacağını dəqiq bilirik.

Bəzən isə elə olur ki, dövrün başa çatması şərtinin nə vaxt ödənəcəyi əvvəlcədən bilinmir.
Bu zaman `while` operatorundan istifadə olunur.

`while` operatorunun sisntaksisi aşağıdakı kimidr.

```
while(şərt){
əməliyatlar; }
```

nümunə:

```
char x;
x='b';
while (x!='a'){
std::cout<<"Salam dunya\n";
std::cout<<"  yeni  simvol daxil edin\n";
std::cin>>x;
       }
```

Bu kod icra olunduqda ekranda Salam dünya sətri çap olunacaq və proqram isitfadəçinin hər-hansı simvol daxil etməsini gözləyəcək. Əgər bu simvol `'a'` -dırsa 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` opratorundan 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 şərtdən asılıdırsa bu zaman `if` opratoru ilə bu şərtlərin mürəkkəb konfiqurasiyasından istifadə etmək əvəzinə switch operatorundan istifadə edirlər.

`switch` operatorunun sintaksisi aşağıdakı kimidir:

```
   switch ( dəyişən ) {
case qiymət1:
  yerinə yetirilməli proqram hissəsi  /*  əgər dəyişənin qiyməti == qiymət1 */
  break;
case qiymət2:
  yerinə yetirilməli proqram hissəsi  /*  əgər dəyişənin qiyməti == qiymət2  */
  break;
...
default:
yerinə yetirilməli proqram hissəsi     /*  yuxarıdakı şərtlərin heç biri ödənmədikdə  */
  break;
}
```

`switch` opratoru 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` opratoru işini dayandırır və proqramda `switch` -dən sonra gələn oprator 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` seçimindən istifadə etmək vacib deyil. Əgər qiymətlərdən heç biri ödənməsə onda `default:` seçimində göstərilən operatorlar yerinə yetiriləcək.

nümunə proqram:

```
#include <iosteram>

int main(){
int color = 0;

std::cout<<"Her hansi reng sechin(qirmizi=1,yashil=2,qara=3):\n";
std::cin>>color;

switch(color){
case 1:  std::cout<<"siz qirmizi rengi seçdiniz\n";
break;
case 2:  std::cout<<"siz yaşıl rengi seçdiniz\n";
break;
case 3:  std::cout<<"siz qara rengi seçdiniz\n";
break;
default: std::cout<<"siz hech bir reng sechmediniz\n";
}
return 0;
}
```

Proqramı kompilyasiya edib yerinə yetirək.

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

3.4 continue və break

c`ontinue` 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 belə yazılış doğrudur:

Bu kod icra olunduqda heç nə yerinə yetirilməyəcək, çünki şərt ödənmir (0-dır)

```
if (0)
{
std::cout<<"Bakida havalar yaxshi kecir \n";
}
```

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

```
while(1)
{
std::cout<<"Bakida havalar soyuq kecir \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. Ele 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.

Bir çox proqramçılar məhs göstəricilər mövzusunda C++ dilini orgənməkdən imtina edir.

4.1 Dinamik və Statik dəyişənlər

Bax qeydlərə.

Proqramda istifadə olunan dəyişənlər iki cür olur, `statik` və `dinamik` .

Bizim indiyə qədər istifadə etdiyimiz dəyişənlər hamısı `statik` dəyişənlərdir.

`Statik` dəyişənlərin `dinamik` dəyişənlərindən heç bir üstünlüyü yoxdur.

Çatışmamazlıqları isə həddən artıq çoxdur.

Statik 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 ünvana istinad edirlər, bu dəyişənlərin istinad etdiyi ünvanı dəyişdirmək olmaz.

Dinamik dəyişənlər :

Dinamik dəyişənləri proqramın icrasının istənilən anında yaratmaq olar,
Onlara ayrılmış yaddaşı proqramın icrasının istənilən anında geri alıb həmin yeri istənilən digər məqsəd üçün istifadə etmək olar,
Dinamik dəyişənlərin yaddaşda istinad etdikləri ünvanı istənilən digər ünvana dəyişdirmək olar, hətta digər proqramın və ya nüvənin yaddaş sahəsinə.
Ancaq buna etmək istədiyimiz ilk cəhddə nüvə proqramımızı təmamilə söndürər(Başqa proqramların məlumatlarına icazəsiz müraciət etmək olmaz, əgər etsəniz deməli siz - hakersiniz).

Ünvan dəyişənlərinin özəlliyi odur ki, onlar özlərində heç vaxt heç bir məlumat saxlamırlar.

Məlumat yaddaşda olur, onlar isə sadəcə bu yaddaşın ünvanın özlərində saxlayırlar.

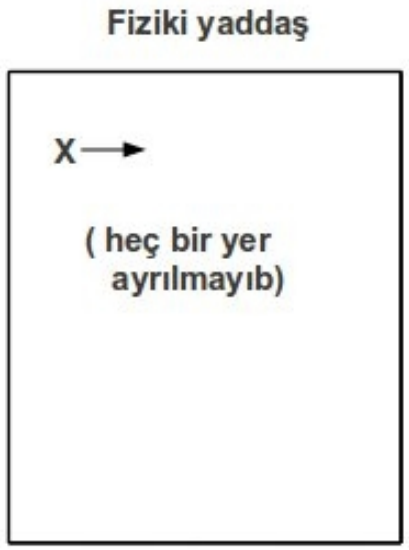
Əgər biz bu ünvanı dəyişsək onda onlar ayrı məlumata istinad edəcəklər.

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.

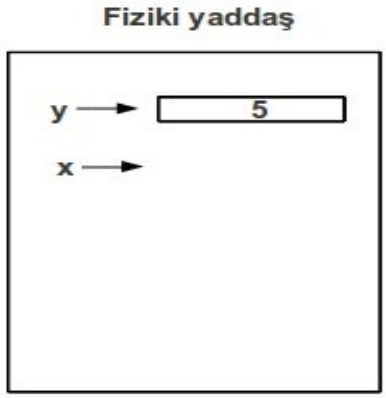
Bu zaman yaddaşın vəziyyəti belədir(şəkil 5):



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:



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

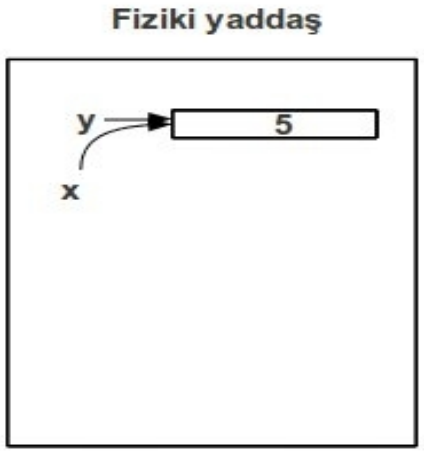
Bunun üçün & ünvan operatorundan istifadə edəcəm.

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

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

```
x = &y ;
```

Yaddaşın vəziyyəti:



4.2 Dinamik yaradılma

Indi isə gəlin yaddaşda 4 bayt əlavə yer ayıraq və x -i bu yerin ünvanına mənimsədək.

Bunun üçün new funksiyasından istifadə edəcəyik.

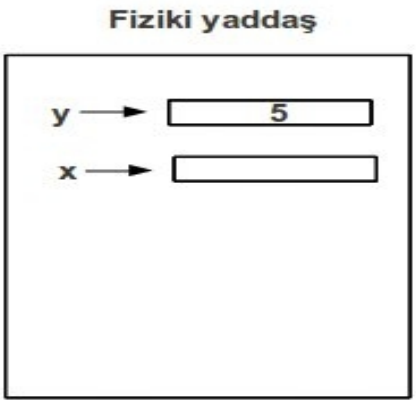
Sintaksis aşağıdakı kimidir:

```
unvan_deyisheni = new tip;
```

Aşağıdakı kimi:

```
x = new int;
```

Bu zaman yaddaşın vəziyyəti belə olar:



4.3 Dinamik silinmə

Biz bu yerdən istədiyimiz qədər istifadə edə bilərik. Artıq bu yerə ehtiyacımız qalmadıqda onu silə bilərik.
Beləliklə də həmin sahə başqa məqsədlər üçün istifadə oluna bilər.

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

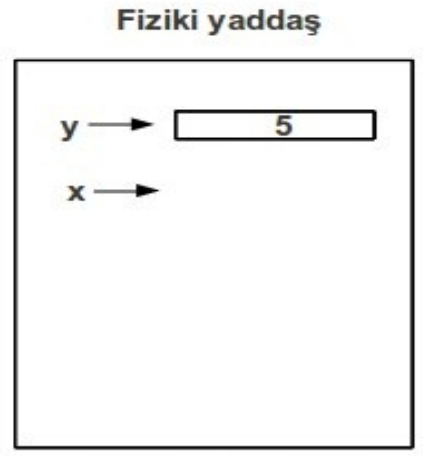
Sintaksis belədir:

```
delete unvan_deyisheni
```

Misal üçün :

```
delete x;
```

Yaddaşın vəziyyəti:



Biz ünvan dəyişənləri üçün yaddaşda yer ayırma və ünvan dəyişənlərinin digər yaddaş sahələrinə ünvanlanması ilə tanış olduq.

İndi gəlin ünvan dəyişənlərinin istinad etdikləri məlumatın oxunması və dəyişdirilməsi ilə məşğul olaq.

Adi dəyişənlərin yaddaşdakı qiymətini oxumaq və ya dəyişdirmək üçün biz onların adından istifadə edirik.

Ünvan dəyişənlərinin istinad etdikləri məlumata müraciət də eyni qayda ilə olunur, sadəcə ünvan dəyişənlərinin əvvəlinə `*` simvolunu artırmaq tələb olunur.

Elanda olduğu kimi.

Misal üçün: x ünvan dəyişəni elan edək, bu dəyişənə yer ayıraq , bu yerə 8 qiyməti yazaq və daha sonra həmin qiyməti başqa adi dəyişənə mənimsədək.

```
// adi ve unvan deyishenlerini eyni setirde elan etmek olar
int y, *x;

// x-e yer ayiririq
x = new int;

// bu yere 8 qiymeti yaziriq
*x = 8;

// x -in istinad etdiyi qiymeti y-e menimsedirik
y= *x;
```

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

İrəlidəki paraqraflarda biz bu biliklərimizi daha da möhkəmləndirəcəyik.

İndi isə sizi proqram nümunələri ilə tanış olaq. Əvvəl həlli ilə, sonra sərbəst işləmək üçün.

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

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

Proqram 1:

```
#include <iostream>
int main(int argc, char *argv[]){
int x;
int *y;     // tam tipli unvan deyisheni

x=5;
y=&x;       // & unvan operatorudur
std::cout<<"x-in qiymeti = <<x<<" x-in unvani "<<y;
return 0;
}
```

Proqram 2:

```
#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;
}
```

Proqram 3:

```
#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;
}
```

Çalışmalar

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

2. Ancaq ünvan dəyişənlərindən istifadə etməklə iki ədədin maksimumunu hesablayan proqram 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_adı ( 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_adı` 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 mötərəzə 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 funksiyanın_adı ( tip1 , tip2 , ...);
```

Nümunə:

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

Burada biz `int` tipli nəticə qaytaran və `int` tipli iki arqument qəbul edən cəm funksiyası elan elədik.

Biz bunu aşağıdakı kimi də yaza bilərik, harada ki arqumentlərin adları göstərilmir.

```
int cem (int , int );
```

5.2 Funksiyanın mətn kodunun tərtibi

Funksiyanın mətn kodunu tərtib etməklə biz onun görəcəyi işi proqramlaşdırmış oluruq.

Bunun üçün aşağıdakı qaydadan istifadə edirik:

```
nəticənin_tipi funksiyanın_adı ( tip1 arg1, tip2 arg2) {

proqram kodu

return nəticə;

 }
```

Burada ilk sətir funksiyanın elanı sətridir. Fərq yalnız odur ki, mötərəzədən 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:

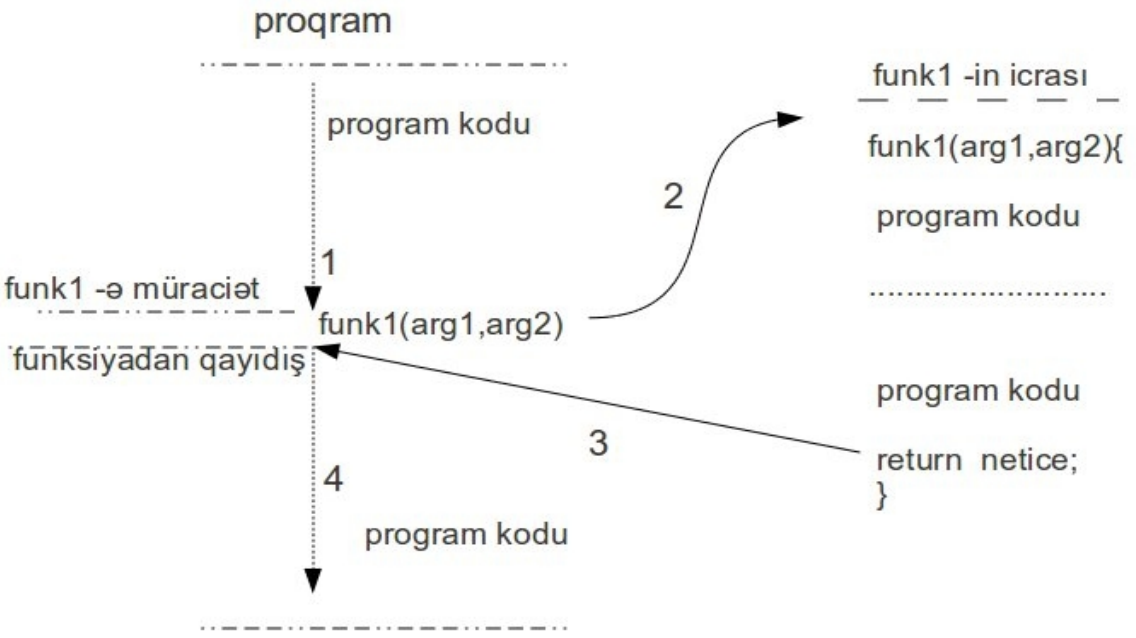
```
int cem ( int x, int y)
{
int z;
z = x + y;
return z;
}
```

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 çağrılan funksiyanın kodu yerləşən hissəyə ötürülür.

Funksiya öz işini yekunlaşdırdıqdan sonra isə icraolunma yenidən proqramın funksiya çağrılan yerinə qaytarılır(həmin yerin ünvanı yaddaşa yerləşdirilmişdi).

Funksiyanı çağırmaq üçün biz onun adından istifadə edirik(aşağıdakı proqram nümunəsinə bax).

Bəs icraolunma funksiyadan onu çağıran kod hissəsinə geri necə ötürülür.

Funksiyanı 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 vastəsilə həyata keçirilir.

Bu zaman funksiyanın qaytaracağı məlumatı `return` operatoruna arqument 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:

```
#include <iostream>

/*cem finksiyanin elani  */
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 funksiyasin cagiririq  */
z = cem(x,y);

std::cout<<"x ile y -in cemi = "<<z<<"\n";
return 0;   }
```

```
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

/* cem finksiyasinin 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 sətri yerləşir, daha sonra isə proqramın əsas funksiyası `main` funksiaysı 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 demək olduğunu bilmək üçün biz `blok` anlayışını daxil etməliyik.

C++ dilində `{` və `}` mötərəzələri arasında qalan hissə `blok` adlanır.

Əgər diqqət yetirsəniz, görərsiniz 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 */
}
```

İndi biz `lokal` və `qlobal` dəyişənlərin izahını verə bilərik.

Dəyişən, elan olunduğu `blok` -dan *"üst"* -də yerləşmiş bloklar üçün `lokaldır` , yəni *"üst"* -dəki bloklar bu dəyişəni görmür.

Dəyişən, elan olunduğu `blok` -un *"içəri"* -sində yerləşmiş bloklar üçün `qlobaldır` , yəni *"içəri"* -dəki bloklar bu dəyişəni görür.

Biraz əvvəl daxil etdiyimiz proqrama nəzər salsaq görərik ki, `cem` funksiyası daxilində biz `int` tipli `dey3` dəyişəni elan etmişik.

Aydındır 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 arqument 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 əməlyyatlar funksiya bitdikdən sonra qüvvədə qalır.

Dəyişənlərin ünvana görə ötürmək üçün biz funksiyanı aşağıdakı kimi elan etməliyik:

```
nəticənin_tipi funksiyanın_adı ( tip1 *arg1, tip2 *arg2, ...);
```

Dəyişənləri bu funksiyaya parametr 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);
```

Çalışmalar:

1. Funksiyalardan istfadə etməklə iki ədədin maksimumunu hesablayan proqram tətrib edin.

2. Elə funksiya qurun ki, istifadəçidən 10 tam ədəd daxil etməsini istəsin və onların cəmini qaytarsın. Bu funksiyadan 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 funksiyasından istfadə etməklə proqram qurub icra edin.

4. Çalışma 3-dəki kvadrat funksiyasını elə dəyişin ki, tərəflərinin sayı və təşkil olunduğu simvol bu funksiyaya parametr kimi ötürülsün.
Bu funksiyadan 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 funksiyadan istifadə etməklə proqram tərtib edib, icra edin.

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

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.

8. Kvadrat, romb və ucbucaq funksiyalarından istifadə etməklə elə proqram tərtib edin ki, əvvəl istifadəçidən tərəfin uzunluğun, tərəfin və fiqurun daxilin 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.

9.(*) 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.

$6 Cərgələr və ya Massivlər.

Biz baxdığımız nümunə proqramlarda elə də çox sayda dəyişəndən istifadə etmirdik.

1,2 və ya 3 dəyişən maksimum halda.

Aydındır 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, univeristetdə bu il bütün fakultələr 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 (əlbəttdə 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 eləmək üçün cərgənin adı və `0` indeksindən istifadə olunur.

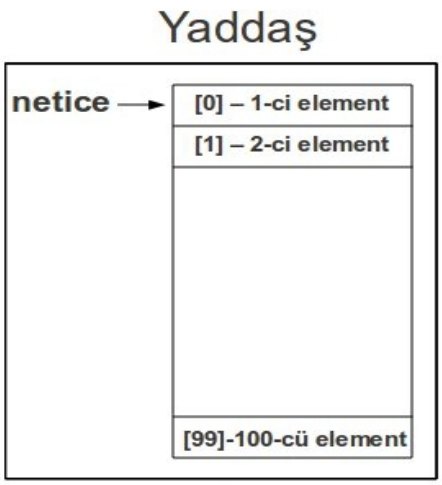
Cərgə elan eləmək üçün aşağıdakı çox sadə sintaksisdən istifadə edirik:

```
tip cərgənin_adı[elementlərin_sayı];
```

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

```
int netice[100];
```

Yaddaşa nəzər salaq:



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_adı[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.

Proqram nümunəsi:

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

```
#include <iosteram>


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;
}
```

6.2 Cərgələrin elan olunma qaydaları

Cərgələr aşağıdakı kimi elan oluna bilər:

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

İkinci qayda:

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

Bu zaman cərgənin elementlərinin sayı `[]` mötərəzələri arasında verilmir, onun təşkil olunduğu elementlər `{}` mötərəzələri arasında göstərilir.

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 isitfadə olunur.

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

Aşağıdakı kimi adi 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 ünvanlana bilməz.

Bu deyilənə əsasən `z` cərgəsinin ilk elementinə `z[0]` -dən başqa aşağıdakı kimi də müraciət etmək olar.

```
*z;
```

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

`x = z[0];` və ya `x = *z;`

Ancaq ilk elenemtə əlbəttdə ki;

Digər tərəfdən `z` ünvan tipli dəyişən oluğundan 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 etişik.

```
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:

```
#include <iostream>

int main(int argc, char *argv[]){
int i, x[100], say;

std::cout<<"100 -den kicik her hansi 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;
}
```

Proqramı kompilyasiya edib icra edək:

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

Proqram 2. max.c

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

```
#include <iostream>

int main(int argc, char *argv[]){
int i, x[100], say, max;

std::cout<<"100 -den kicik her hansi 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;
}
```

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
100 -den kicik her hansi bir eded daxil edin
7
7sayda 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>
```

Proqram 3. max_func.c

Proqram 2 -ni funksiyadan istifadə etməklə tərtib edək.

```
#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 hansi 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;
}
```

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
100 -den kicik her hansi 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>
```

Çalışmlar

1. max proqramını ele dəyişin ki, istifadəçinin daxil etdiyi ədələrin içində ən kiçiyini tapsın.

2.(*) Elə proqram qurun ki, istifadəçinin daxil etdiyi ədədləri artan sıra ilə düzsün.

3. Elə proqram 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 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.

5. 4 - cı proqramı elə dəyişin ki, istifadəçi say olaraq 0 daxil edənə kimi proqram təkrar olunsun.

6. Elə funksiya qurun ki, verilmiş siyahının elementlərini artan sıra ilə düzsün. Bu funksiyadan istifadə etməklə elə proqram 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 ardıcıllığı olduğundan, sətir kimi elan etdiyimiz yerə istənilən məlumat yaza bilərik.

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:

Adi biz bildiyimiz cərgənin hər bir elementinə qiymət mənimsətmə yolu ən az istifadə olunandır.

Misal üçün, `20` simvoldan ibarət sətir elan edək, daha sonra bu sətrə `"Azerbaycan"` sözünü yerləşdirək.

Qeyd eliyirəm bir daha ki, bu qayda ilə sətrə 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 sətrin sonuncu elementinə mənimsətdiyimiz `'\0'` simvolu xüsusi simvoldur və sətirlərin sonunu bildirmək üçün istifadə olunur.

Aşağıda daxil edəcəyimiz sətir funksiyaları sətrin sonunu bu simvola görə təyin edir. Beləki sətir funksiyaları sətirdə `'\0'` simvoluna rast gəldikləri yerdən sətrin qalan hissəsini inkar edirlər.

Bu misalda sətirlərin adi `char` tipindən olan cərgə olmaları aydın görülür.

Yuxarıdakı `set` sətrinə `"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 Oveflow (buferi daşırmaq) və digərlərini misal göstərmək olar.

C++ dilində yazılmış proqramların zəif tərəflərindən biri `strcpy()` funksiyasıdır.

Beləki, pis məqsədli hər kimsə, `strcpy()` funksiyasına parametr olaraq böyük olçülü sətir verə bilər.

Nəticədə `strcpy()` həmin sətri göstərilən ünvandan proqramın daxilinə köçürəndə proqramın bütün ehtiyat yaddaşın doldurar və nəticədə proqram dayanmağa məcbur olar.

Çıxış yolu `strcpy()` əvəzinə `strncpy()` finksiyasından istifadə etməkdir.

`strncpy()` funksiyası verilmiş ünvandan ən çoxu verilmiş sayda simvol köçürməyə icazə verir.

Sadə məsələlərdə `strcpy()` -dən istifadə edə bilərsiniz, amma ciddi proyektlərdə mütləq `strncpy()` -dən istifadə etmək lazımdır.

`strcpy()` funksiyasının sintaksisi aşağıdakı kimidir:

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

Burada `menbe` köçürüləcək sətrin ünvanı, `menseb` sətri köçürmək istədiyimiz ünvandır.

Proqram nümunəsi:

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

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

strcpy(set,"Ekvator");

std::cout<<set;
return 0;    }
```

strlen() fuksiaysı.

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

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

strncpy() fuksiaysı.

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

`s2` sətrinin ilk `n` elementini `s1`-ə köçürür(`s1`-in əvvəlindən başlayaraq).

Qeyd: bu funksiya sətrin sonuna `'\0'` simvolunu yazmır.

strcmp() fuksiaysı.

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

Bu funksiyadan sətirlərin müqaisəsi üçün istifadə edirlər. Əgər `s1` sətri `s2` sətri ilə eynidirsə onda funksiya `0` qiymətini qaytarır.

Əgər `s1`-in elementlərinin sayı `s2`-den azdırsa onda `<0` əks halda `>0` qiymətini qaytarır.

strcat() fuksiaysı.

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

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

Proqram nümunələri:

Proqram 1:

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

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

int main(){
char  set[1024];
int k;

std::cout<<"Zehmet olamsa her hansi setir daxil edin \n";
std::cin>>set;

k = strlen(set);

std::cout<<"Sizin daxil etdiyiniz setrin simvollarinin sayi
"<<k<<" \n";
return 0;
}
```

Proqramı icra edək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Zehmet olamsa her hansi setir daxil edin
Olimpiada
Sizin daxil etdiyiniz setrin simvollarinin sayi 9
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

Proqram 2.

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

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

int main(){
char  set[1024];
int k;

std::cout<<"Zehmet olamsa her hansi setir daxil edin \n";
std::cin>>set;

k = strlen(set);

if (k<10)
std::cout<<set<<"\n";
else {
char *p = set;
p+=20;
std::cout<<p<<"\n";
      }
return 0;
}
```

Proqramı icra edək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Zehmet olamsa her hansi setir daxil edin
SerguzeshtiVeziriXaniLenkeran
iLenkeran
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>
```

Çalışmalar:

1. Elə proqram qurun ki, istifadəçinin daxil etdiyi sətrin 5-ci simvolu ilə 15-ci simvolu arasında qalan hissəsini çap etsin. Əgər sətrin 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 ardıcıllığında birləşdirərək tam sətir kimi çap etsin.

4.* Elə proqram tərtib edin ki, istifadəçidən 5 sətir qəbul etsin və bu sətirləri uzunluqlarının artma ardıcıllığı ilə alt-alta çap etsin.

$8 Struct tiplər.

İndiyə qədər biz dəyişən elan edərkən `int, double, char, long` kimi standart tiplərdən istifadə edirdirk. Əlbətdə bu tiplər çox əlverişlidir, lakin çox vaxt məsələnin şərtinə uyğun olaraq proqramçının özü yaratdığı tiplərdən istifadə etmək lazım gəlir.

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.

Yeni `struct` tipi təyin etmək üçün sintaksis aşağıdakı kimidir.

```
struct yeni_tipin_adı {
tip  dey1;
tip  dey2;
tip  dey3;
.
.
tip  deyn;     };
```

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 tipdən adi tiplər kimi dəyişənlər və ünvan dəyişənləri elan edə bilərik.

Məsələn: `ishci dey1, ishci1, *muhendis, yeni_ishciler[100];`

Yuxaridakı elanda biz `ishci` tipindən olan `dey1` və `ishci1` dəyişənləri, `muhendis` unvan 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 ishciler[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 oların 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 proqram tərtibinin vaxtı çatıb.

Proqram 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;
}
```

Proqramı 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 Strukut tipindən olan ünvan dəyişənləri

strukt tipindən aşağıdakı kimi ünvan dəyişəni təyin edək:

```
ishci *muhendis;
```

Əgər biz `strukt` tipdən adi yox unvan 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 unvan 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 operatordan 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;
}
```

```
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` simvollu sətir tipli `soz` dəyişənlərindən ibarət olan `str` adlı yeni `struct` tipi yaradın.
Bu yeni yaratdığınız tipdən `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.

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 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 elemntlərinə mənimsətsin.
Daha sonra `max_el` funksiyası tərtib edin, hansı ki, `strler` cərgəsinin elementləri arasında `x-i` ən böyük olanın qiymetlerini (`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 tipdən olan elementlər ardıcıllığını 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 bilməriksə, 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) ardıcıllığıdır və bu elementlərin(obtektlərin) hər birində müxtəlif məlumatlar yerləşdirmək olar.

İndi biz həmin 1 obyekti yaratmağa çalışırıq, daha sonra bir neçə bu cür obyekti bir-biri ilə əlaqələndirib siyahı yaratma qaydasını örgənəcəyik.

Baxdığımız sadə halda yaratmaq istədiyimiz obyekt özündə bir məlumat - `int` tipindən olan dəyişən saxlayır.

Bu obyekti yaratmaq üçün biz `struct` tipdən istifadə edəcəyik.

`Aşağıdakı kimi:`

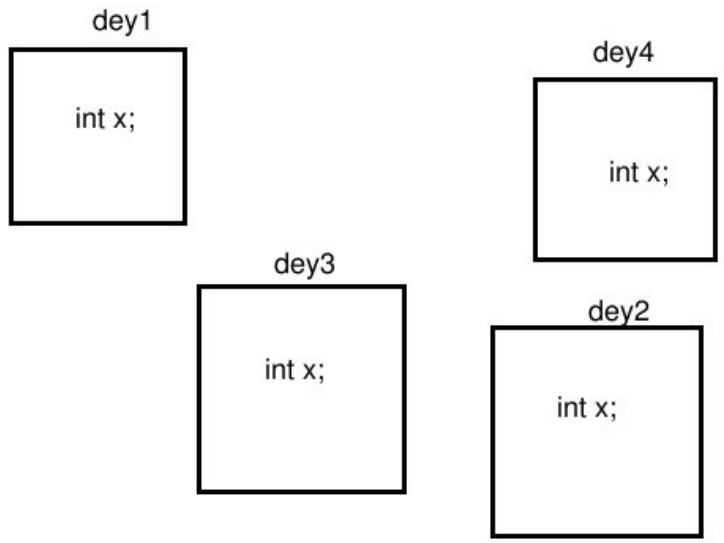
```
struct syh_el {
int x;
}
```

Yuxarıda biz özündə `int` tipindən olan dəyişən, `x` -i saxlayan yeni `struct` tipi yaratdıq.

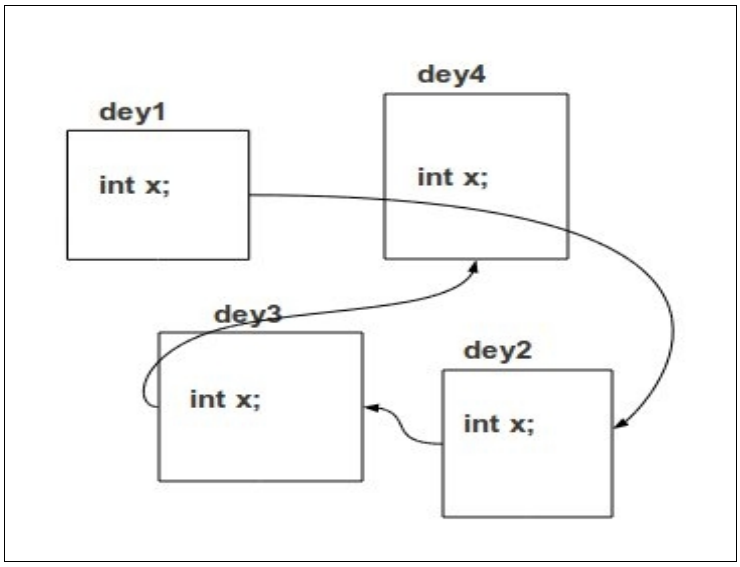
İndi bu tipdən olan bir neçə dəyişən elan edək.

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

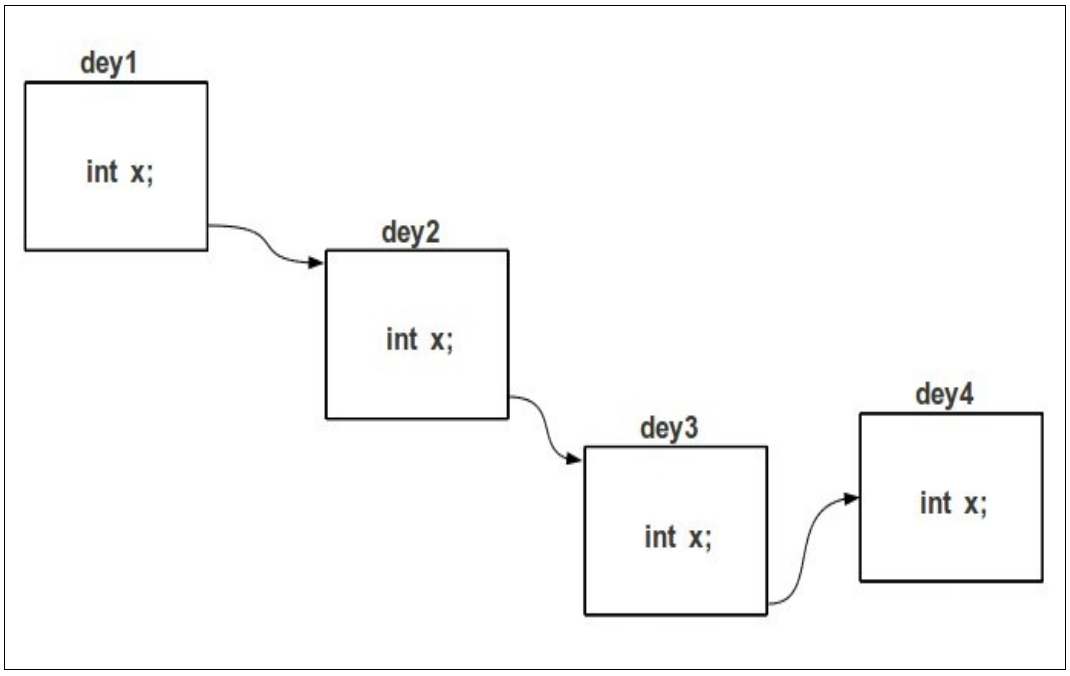
Hal-hazırda yaddaşda vəziyyət aşağıdakı kimidir:



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



Bir az daha səliqəli göstərsək, aşağıdakı kimi:



Şəkildə çəkmək asandır, amma praktikada obyektlər arasında bu əlaqəni yaratmaq elə də asan deyil, ən azından ilk cəhddə.

Siyahının yaradılması üçün aşağıda daxil ediləcək qayda biraz çətin ola bilər.

Bu qaydanı birinci dəfədən tam dəqiqliyi ilə qavramaq üçün onun üzərində baş sındırmaq məsləhət deyil.

Məsləhər görərdim ki, əvvəl aşağıda daxil edəcəyimiz hazır proqram kodlarından istifadə edib nəticələri yoxlayasınız və özünüz bəzi testlər aparasınız.

Siyahılarla işləmə vərdişi yarandıqdan sonra mahiyyəti qavramaq da asan olacaq.

Siyahı yaratma qaydası:

Əsas məsələ bir obyekti başqa obyekt ilə *əlaqələndirməkdir* .

Əgər yaratmış olduğumuz obyektlərdən birini, digəri ilə əlaqələndirə bilsək, daha sonra yeni obyekti digər başqa biri ilə əlaqələndirə bilərik.

Beləliklə də siyahıya istənilən sayda yeni obyekt əlavə edib siyahımızı istədiyimiz qədər uzada bilərik.

Bəs bir obyekti 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 ünvanın bilməlidir.

Bunun üçün obyektin daxilində onun öz tipindən olan ünvan dəyişəni yerləşdirirlər və bu ünvan dəyişəninə siyahının növbəti obyektinin ünvanını mənimsədirlər.

Bir balaca bizim üçün yeni oldu hə?, obyektin daxilində onun öz tipindən ünvan dəyişəni elan etmək. Ancaq nə etməli :)!..

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.

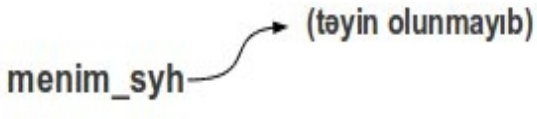
Artıq bu bizə siyahı yaratmağa imkan verir.

Əvvəlcə biz siyahımızı elan etməliyik. Bu zaman biz prinsip etibarı ilə `struct syh_el` tipindən olan ünvan dəyişəni elan etmiş oluruq.

Aşağıdakı kimi:

```
syh_el *menim_syh;
```

Yaddaşın vəziyyəti:



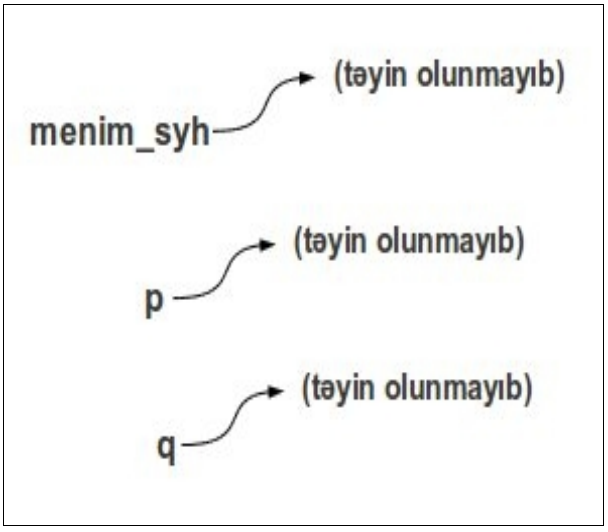
Siyahı yaratmaq üçün `syh_el` tipindən olan daha iki ünvan dəyişəninə ehtiyacımız olacaq. Bunları `p` və `q` ilə işarə edək.

Bunlardan biri - `p` , yeni obyektlərin yaradılması və inisializasiyası (ilkin qiymətin mənimsədilməsi), digəri - `q` isə iterasiya üçündür(siyahı boyu hərəkət etmək).

Gəlin bu dəyişənləri də elan edək:

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

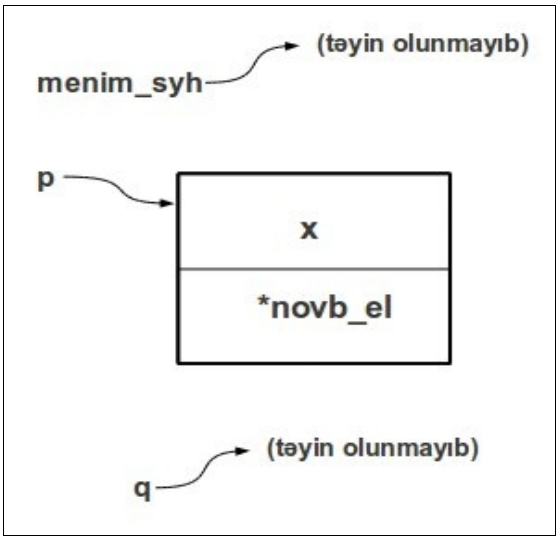
Yeri gəlmişkən yaddaşa da bir ötəri nəzər salaq:



Hər şey hazırdır, siyahının ilk elementini yarada bilərik.
Bunun üçün yazırıq:

```
p = new syh_el;
```

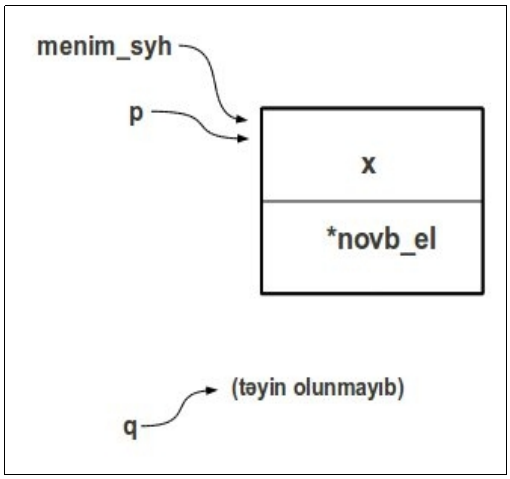
Bu zaman yaddaşın vəziyyəti aşağıdakı kimi olar:



Növbəti addım `menim_syh` dəyişənini bu yeni yaratdığımız obyektə mənimsətməkdir.
Bunun üçün sadəcə yazırıq:

```
menim_syh = p;
```

Yaddaşın vəziyyəti aşağıdakı kimidir:



Artıq `menim_syh` dəyişənindən istifadə etməyəcəyik.

O siyahının ilk elementinə istinad edir və siyahıya müraciət etmək üçün bu dəyişəndən istifadə edəcəyik.

Əlbətdə gələcəkdə siyahı üzərində əməliyyat aparsaq, əvvəlki obyektlərdən bəzilərini silsək onda `menim_syh` parametri üzərində də çevrilmələr aparmalıyıq.

Hələlik isə siyahının yaradılmasını davam etdirək.

İkinci elementi yaratmaq üçün hazırlıq işləri görək.

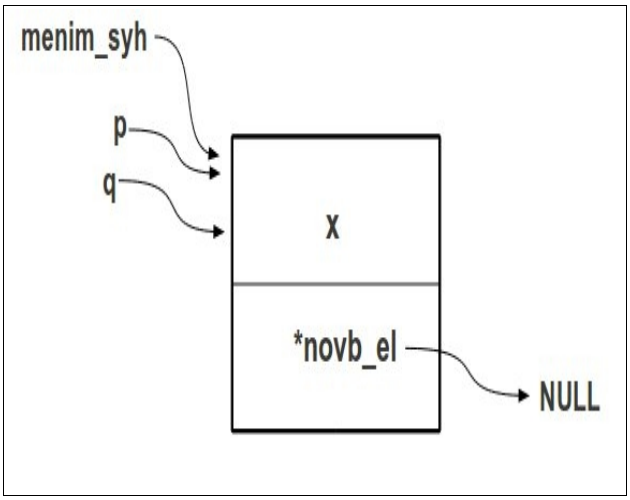
Bunun üçün iterasiya dəyişənini siyahının ilk elementinə (`p` -yə) , ilk elementin ikinci elementlə əlaqələndirmə həddini(`novb_el`) isə `NULL` qiymətinə mənimsətməliyəm.

`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;
```

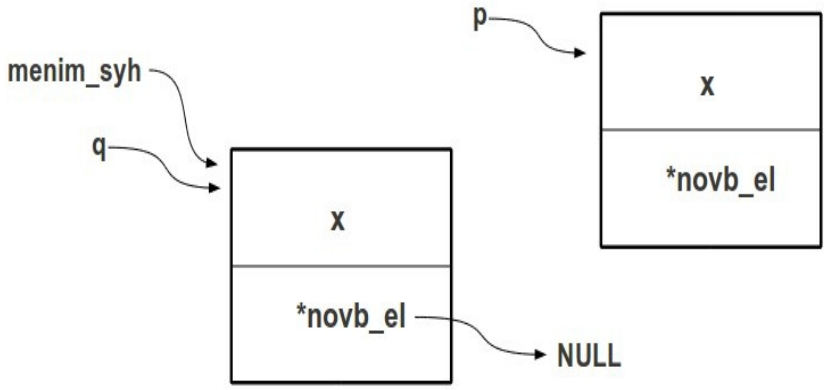
Yaddaşın vəziyyəti aşağıdakı kimidir:



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

```
p = new syh_el;
```

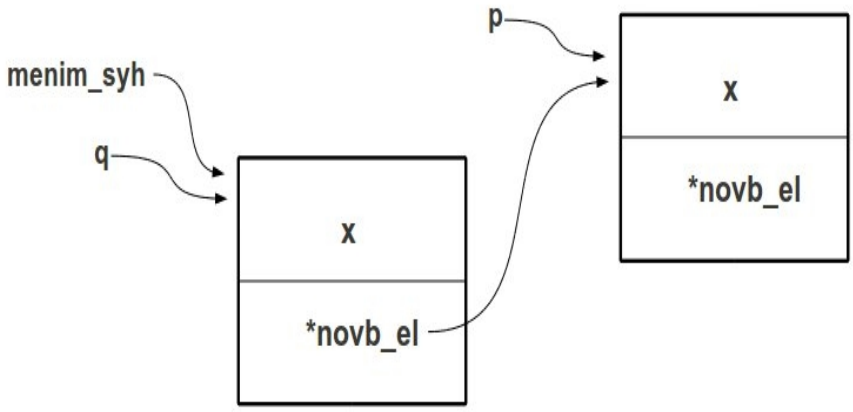
Yaddaşın vəziyyətinə baxaq:



Hey diqqət! Əsas anlardan biri. İlk element ilə yeni yaratdığımız elementi birləşdiririk. Bunun üçün sadəcə yazırıq:

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

Yaddaşa nəzər salaq:

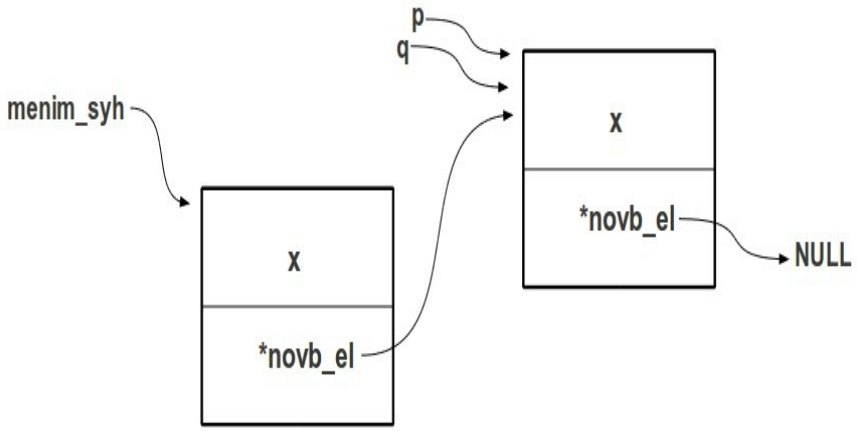


Əla! Əlaqəmiz yarandı.

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

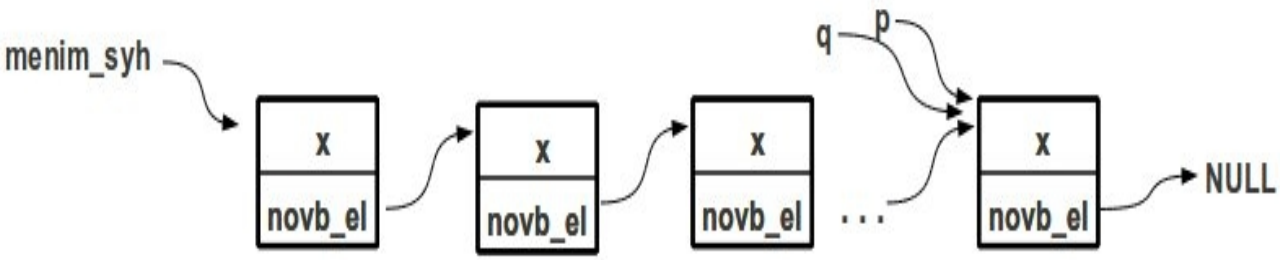
```
q = p;
p->novb_el = NULL;
```

Yaddaşa baxırıq:



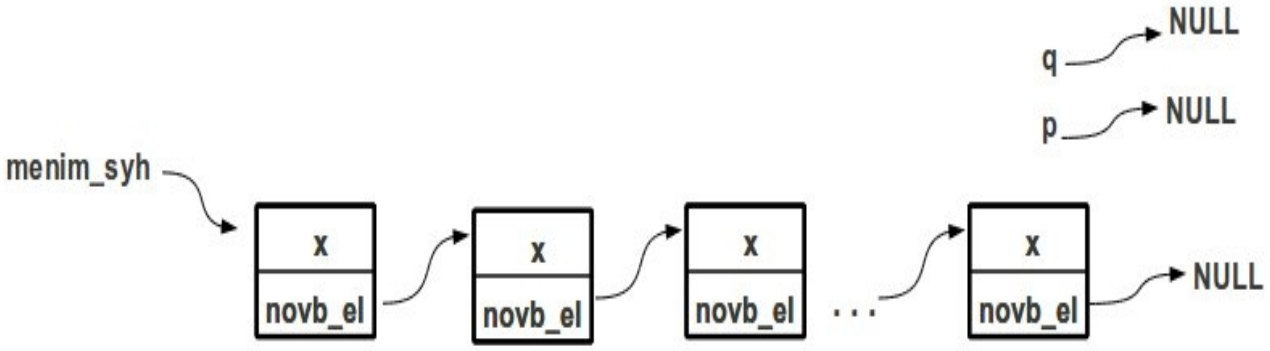
Artıq bu şəkildə davam edərək siyahıya istədiyimiz qədər yeni element əlavə edə bilərik.

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 əvəzedilməzdir və çox praktikdirlər.

Aşağıdakı nümunə proqramları yerinə yetirdikdən sonra siyahılarla işləmə bacarığımız bir qədər də artmış olar.

Proqram nümunələri:

Gəlin yuxarıda daxil etdyimiz, nümunəyə uyğun proqram hazırlayaq.

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

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

Proqram 1.

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

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

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

// ilk elementi yaradaq
p = new syh_el;

// lazimi menimsetmeleri aparaq
menim_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 menim_syh deyisheni yeni yaratdigimiz siyahinin ilk
elementine istinad edir ve onun vastesile siyahini butun obyektlerine
muraciet ede bilerem*/

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

// Yene p ye ehtiyacimiz olacaq
p = menim_syh;

// indi p dayanib siyahinin evvelinde , ashagidaki koda diqqet eleyin
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 obyekti uzerine surushmek ucun ashagidaki
qaydadan istfade 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 menimsetdik, indi onlari cap edek
// p-ni siyahinin ilk ilk elementine menimsedek, bunu etmeyi  bilirik
p = menim_syh;

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

//iknici elementin uzerine susrushek
p = p->novb_el;
dey = p->x;
std::cout<<"Ikinci 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;
}
```

Proqramı yerinə yetirək:

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Siyahinin heddlerinin qiymetlerini daxil edin.
Siyahini birinci heddinin qiymetini daxil edin.
65
Siyahini ikinci heddinin qiymetini daxil edin.
345
Siyahini ucuncu heddinin qiymetini daxil edin.
78
Siayhini elementleri ashagidakilardir:

Birinci element 65
Ikinci element 345
Ucuncu element 78

Siayhinin elementleri 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 proqramın funksiyaların tətbiqi ilə olan variantını nəzərdən keçirək.

Funksiyalardan istifadə etməklə yuxarıda baxrdığımız proqram 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 `siyahi_yarat` və `siyahini_cap_et` kimi adlandıraq.

Birinci funksiya 2 parametr qəbul edir : `syh_el *` tipindən olan dəyişən - yaratmaq istədiyimiz siyahı və `int` tipli dəyişən - siyahıya daxil etmək istədiyimiz elementlərin sayı.

İkinci funksiya isə bir parametr qəbul edir: çap etməli olduğumuz siyahıya istinad, `struct syh_el *` tipli.

Funksiyaların elanları aşağıdakı kimi olar:

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

Proqramı daxil edək:

Proqram 2.

```
#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
menim_syh = NULL;
int say;

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

```
menim_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";
     }
```

Proqramı yerinə yetirək:

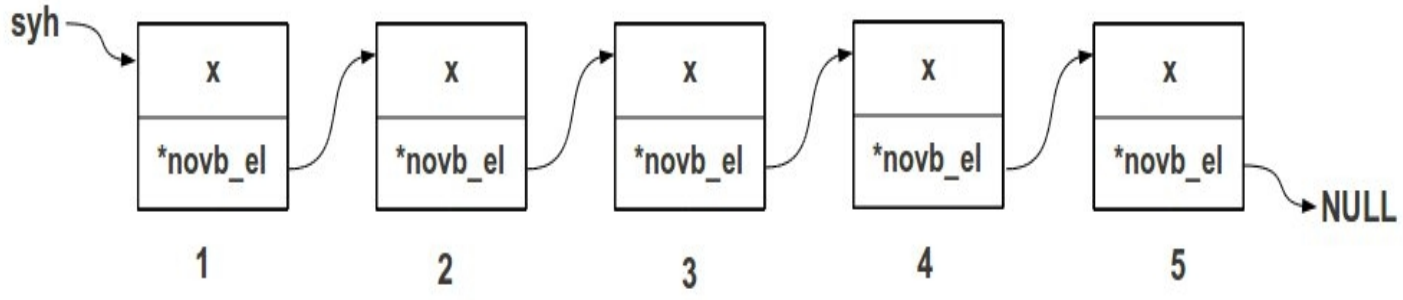
```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Siyahinin elementlerinin sayini daxil edin
5
siyahinin 1 -ci elementini daxil edin
23
siyahinin 2 -ci elementini daxil edin
45
siyahinin 3 -ci elementini daxil edin
567
siyahinin 4 -ci elementini daxil edin
1
siyahinin 5 -ci elementini daxil edin
789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

9.2 Siyahılardan elementlərin silinmsəi.

Biz qeyd etdik ki, siyahılara proqramın icrası boyu istədiymiz 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 obyekti olan aşağıdakı kimi siyahımız var:

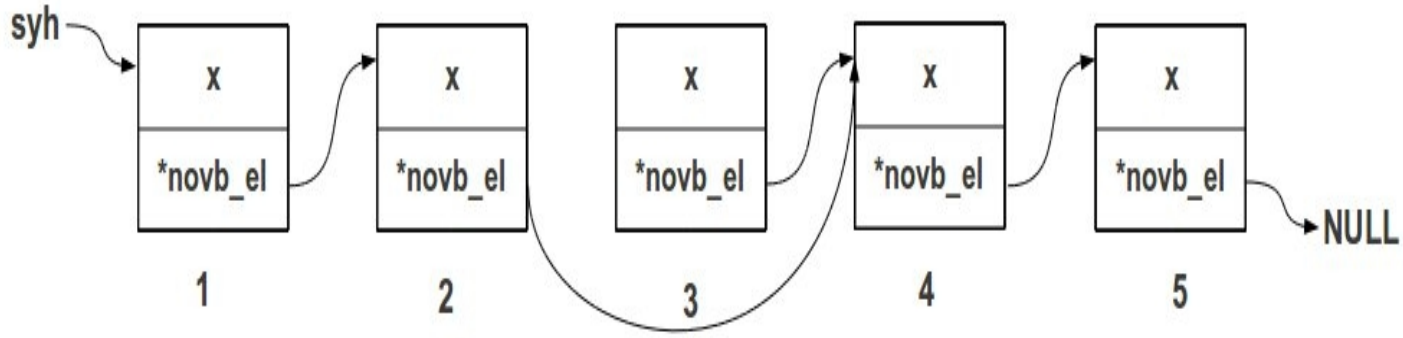


Əlbəttdə siyahının aşağısında göstərdiyimiz nömrələrin siyahı ilə heç bir əlaqəsi yoxdur, sadəcə konkret obyektlərə müraciəti asanlaşdırmaq üçün göstərmişik.

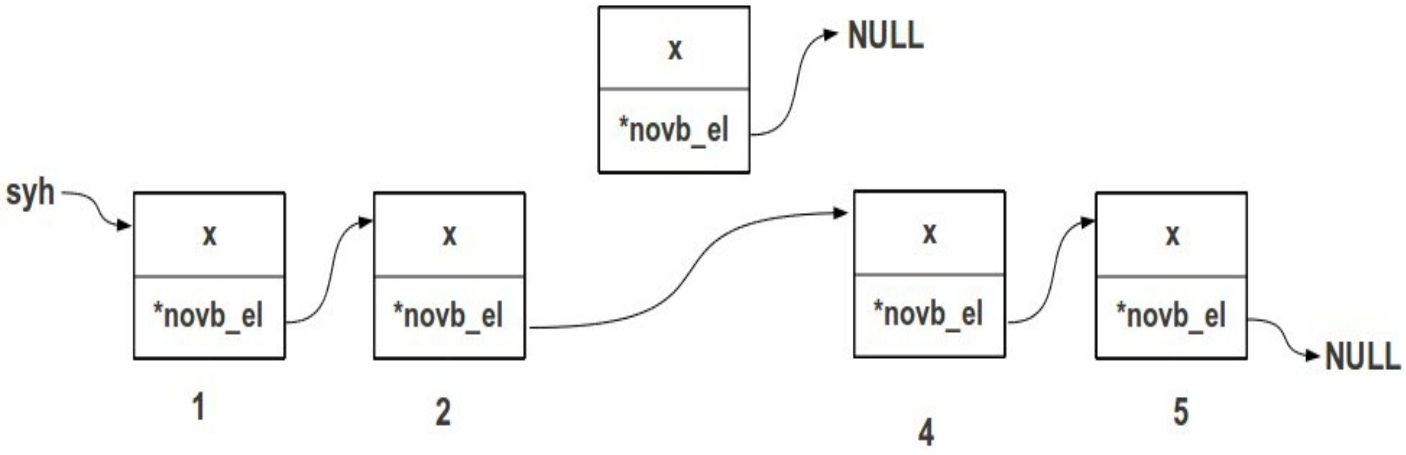
Biz bu siyahıdan üçüncü obyekti silmək istəyirik.

Əvvəlcə qrafik şəkildə görəcəyimiz işi təsvir edək, daha sonra müvafiq proqram kodunu daxil edərik.

Bunun üçün ilk olaraq 2-ci obyektin 3-cü obyektə olan istinadın 4-cü obyektə mənimsədirik:

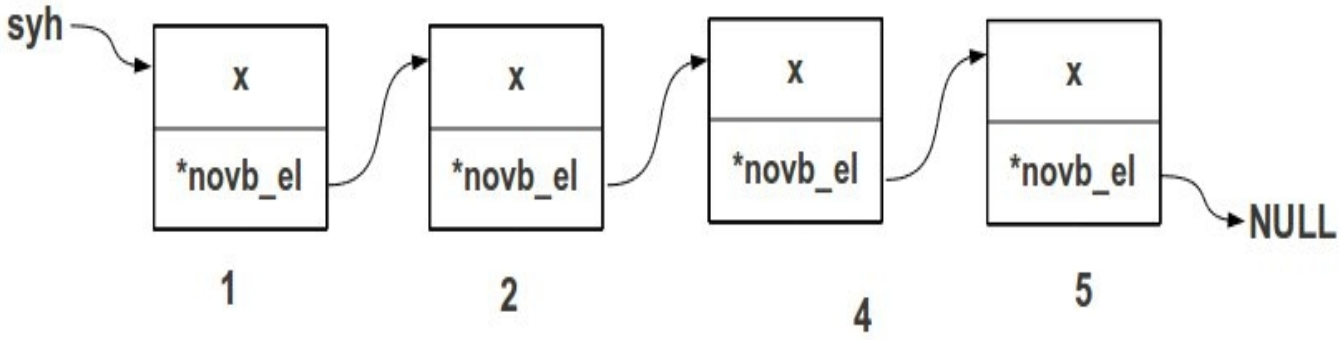


Daha sonra 3-cü obyektin 4-cü obyektə olan istinadın ləğv edirik.



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.



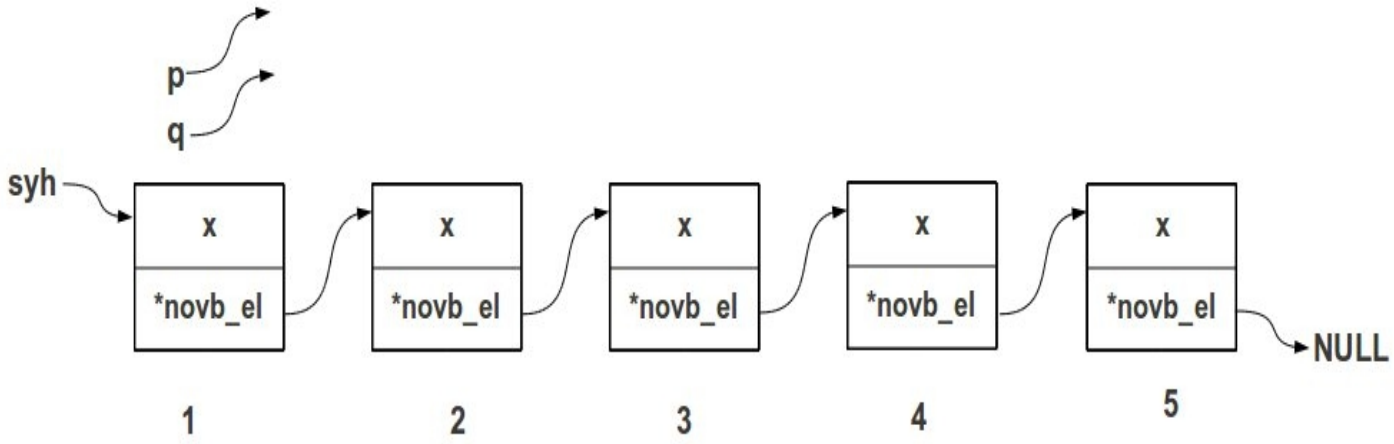
Görəcəyimiz iş ilə tanış olduq, indi isə gəlin bu işi yerinə yetirən müvafiq proqram kodunu daxil edək.

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.

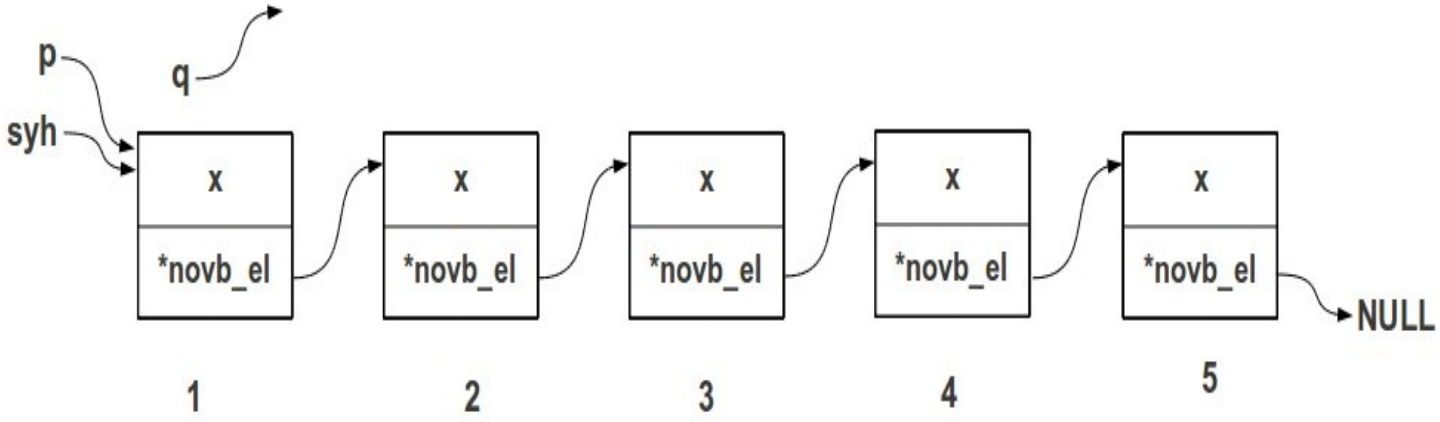
`syh_el *` tipindən olan `p` və `q` dəyişənlərini elan edək.

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



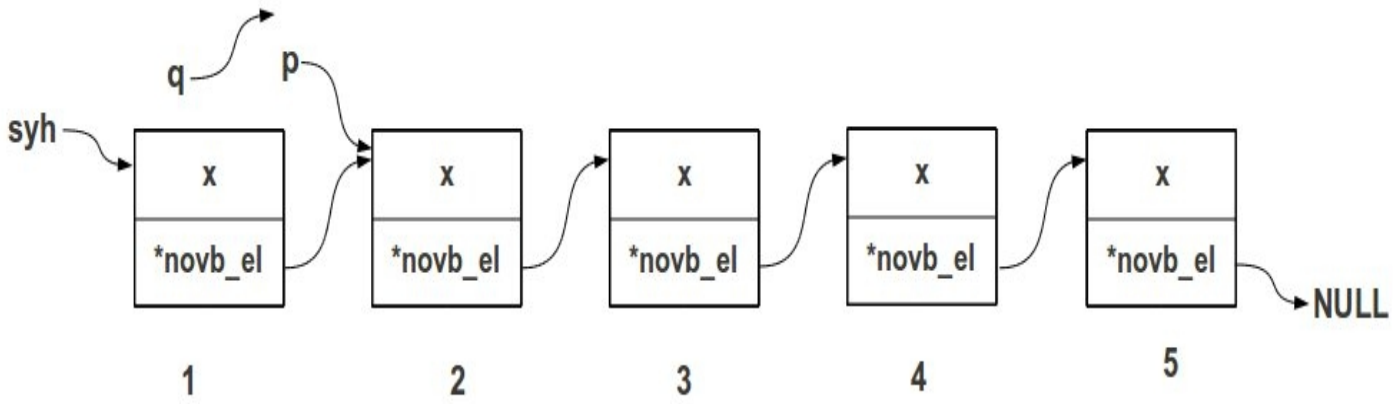
`p` -ni siyahının başlanğıcına mənimsədək.

```
p = syh;
```



`p` -ni siyahının ikinci obyekti üzərinə sürüşdürək.

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



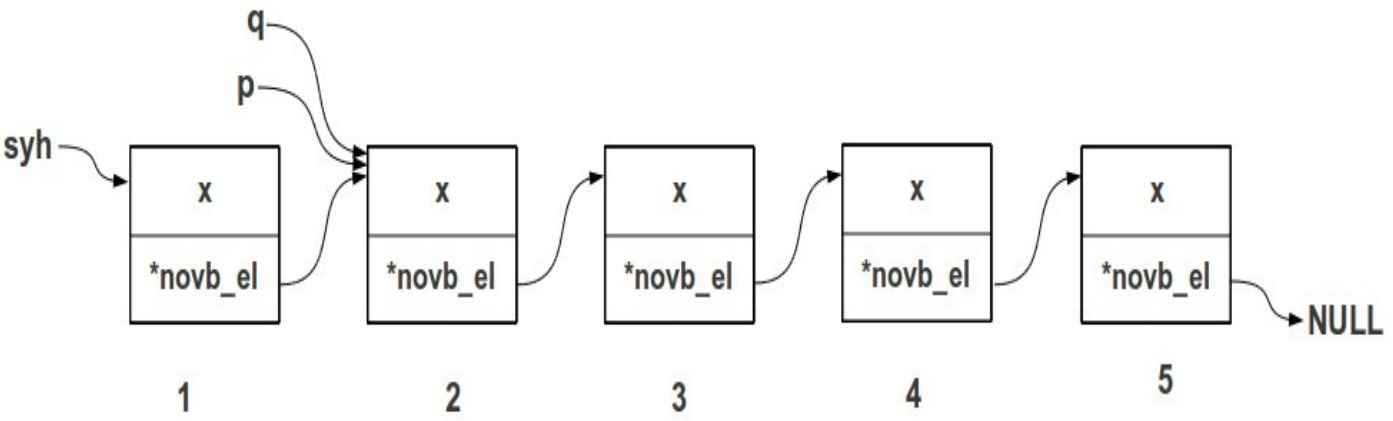
q -nü də ikinci obyektə mənimsətməliyik.

Bu bizə ikinci obyektin `novb_el` həddinə müraciət etməyə imkan verəcək.

Hal-hazırda p ikinci obyektə istinad etdiyindən daha siyahının əvvəldən başlamağa ehtiyac yoxdur.

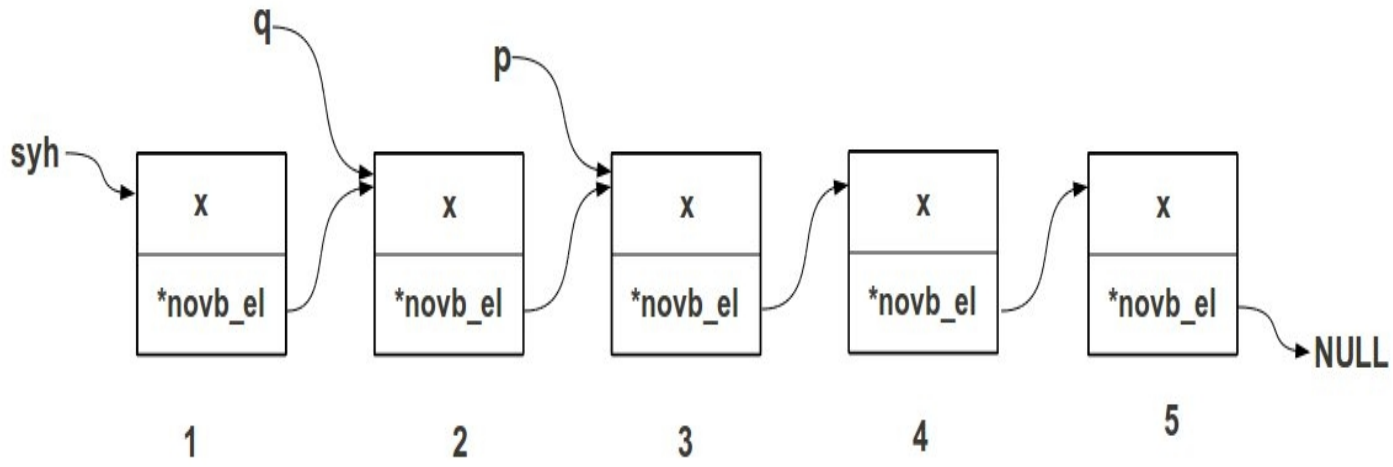
Sadəcə yazırıq:

```
q = p;
```



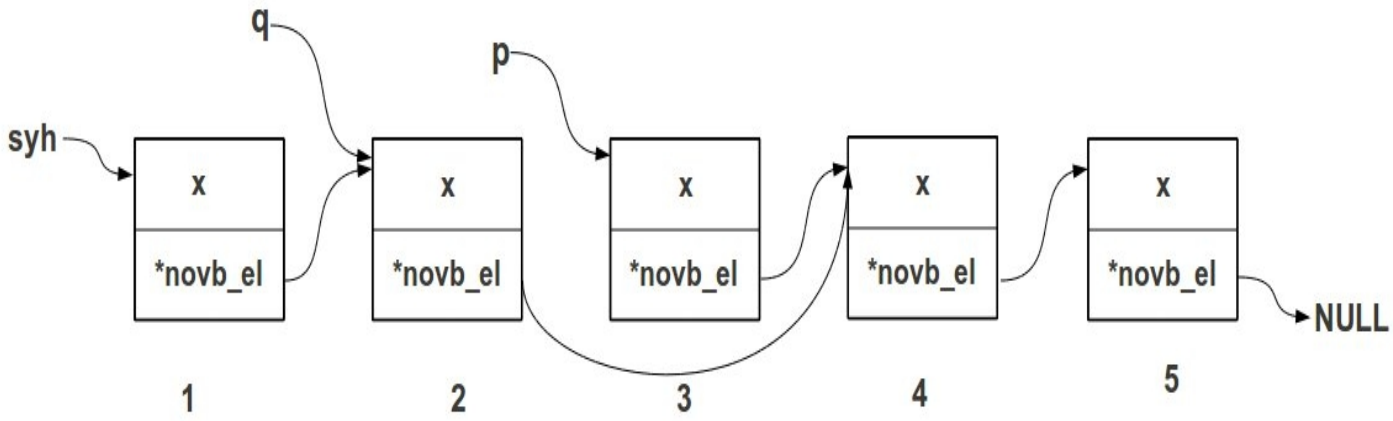
p -ni siyahının 3-cü obyektinin üzərinə sürüşdürək.

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



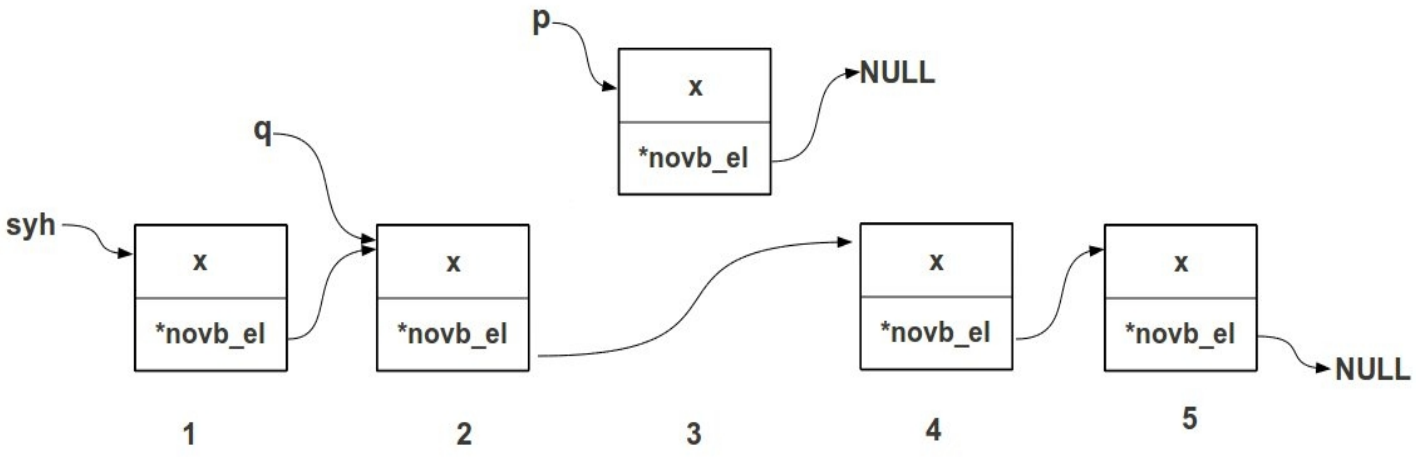
Siyahının ikinci obyektinin `novb_el` həddini 4 -cü obyektə mənimsədək:

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



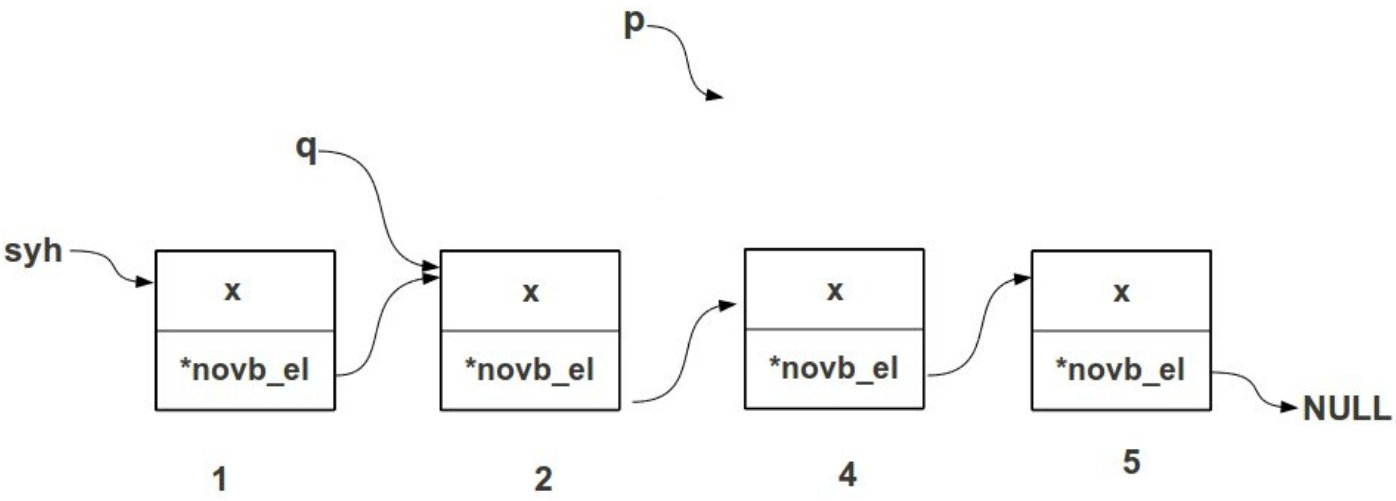
3-cü obyektin siyahı ilə əlaqəsini ləğv edirik :

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



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

```
delete p;
```



Indi isə gəlin bütün bu dediklərimizi icra edən konkret proqram nümunəsi daxil edək.

Proqram 3.

```
#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<<"Siayhidan silmek istediyiniz 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 istediyimiz 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 istediyimiz element sonuncu elementdir
q->novb_el=NULL;
delete p;
p=NULL;
q=NULL;
return syh;
}

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

return syh;
       }
```

```
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
Siayhidan silmek istediyiniz 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 siyahi yaratma və cap_et funksiyalarinda 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 funksiyasın elə dəyişdirin ki, siyahının verilmiş indeksli obyektindən başlayaraq verilmiş sayda obyekti siyahıdan silsin.

$10 Klasslar.

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 *`klasslar`* anlayışıdır.

Burada qeyri-adi, yeni heç bir şey yoxdur.

Sadəcə olaraq klasslara strukt tiplərin biraz fərqli və inkişafetmiş növü kimi baxmaq olar.

Belə ki, strukt tipinin elementlərini tərtib edərkən biz standart tiplərdən və əvvəl yaratdığımız struct tiplərdən istifadə edirdik.

Bu qayda klasslar üçün də keçərlidir.

Lakin klassların struct tiplərdən fundamental üstünlüyü ondadır ki, biz klassların tərkibinə nəinki hər hansısa tipdən olan dəyişən hətta funksiyaları da daxil edə bilərik.

Bu imkan klasslara çox fərqlilik verir və hal-hazırda C++ dilinin ən məşhur dillərdən biri olmasında müstəsna rol oynayır.

Klassdan istfadə etməklə tərtib olunmuş sadə nümunə proqram fikirlərimizə daha da aydınlıq gətirər.

Proqram nümunəsi:

```
#include <iostream>

// sade_klass tipini elan edirik
class sade_klass{
public:
int x;
int y;
int cem(void);
};

// sade_klass -in cem funksiyasinin metn kodu
int sade_klass::cem(void){
return x+y;
}
```

```
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
int main(){
//yeni tertib etdiyimiz klass tipinden dey1 adli deyishen elan edirik
// klasin tipinden olan deyishen numunelerine "obyekt" deyilir.
sade_klass dey1;

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

//dey1 obyektinin cem funksiyasina 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 klasslarla bağlı digər proqram nümunələrinə baxarıq.

Yeni klass tipi yaradarkən `class` sözündən istfadə edirik.

Daha sonra isə yeni yaratmaq istədiyimiz klass tipinin adını yazırıq.

Aşağıdakı kimi:

```
class klasın_adı
```

Klası adi dəyişənləri adlandırdığımız kimi adlandıra bilərik.

Daha sonra isə `{` mötərəzəsini yerləşdiririk.

Nümunə proqramda bu `class sade_klass{` sətrinə uyğun gəlir.

`{` mötərəzəsindən sonra klası təşkil edən elementlər: dəyişənlər və funksiyalar yerləşdirilir.

Klasın sonun bildirmək üçün `}` mötərəzəsindən və sintaksis tələbi olan **;** -dən istifadə olunur.

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

```
class sade_klass{
public:
```

```
int x;
int y;
int cem(void);
};
```

`public` ifadəsinin mənasını irəlidə daxil edəcəyik.

Beləliklə biz artıq `sade_klass` adlı yeni `class` tipi elan etmişik.

Yeni klass tipi yaradarkən yerinə yetirməli olduğumuz ikinci iş klassın funksiya həddlərinin mətn kodunu tərtib etməkdir.

Klasın funksiya həddlərinin mətn kodu adi funksiyaların mətn kodu kimi tərtib olunur, sadəcə funksiyanın hansı klasa məxsus olduğunu bildirmək üçün funksiyanın adının əvvəlinə `klasın_adı::` ifadəsi yazırıq.

Nümunə proqramında bu `int sade_klass::cem(void){` sətrinə 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 parametr 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 klassların aşağıdakı xassəsindən istifadə olunur:

Klasın funksiya həddləri onun dəyişən həddlərinə birbaşa müraciət edə bilər .

`cem` funksiyası daxilində istifadə olunan `x` və `y` dəyişənləri məhs klasın müvafiq həddləridir.

Klasın elementlərini daxil etdikdən və funksiya həddlərinin mətn kodunu tərtib etdikdən sonra biz artıq yeni yaratdığımız klass tipindən dəyişənlər elan edə bilərik, adi tilplərdən olduğu kimi.

Nümunə proqramın `main` funksiyasında biz `sade_klass` tipinidən `dey1` adlı dəyişən elan edirik, aşağıdakı kimi:

```
sade_klass dey1;
```

`Struct` tipində olduğu kimi `class` tipində də həddlərə müraciət etmək üçün nöqtə (**.**) operatorundan istifadə olunur.

Misal üçün `kvadrat` klasının `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 nətcə olaraq qaytarır.

Bu baxdığımız sadə nümunədə biz yeni klass tipi elan etməyi, onun funksiya həddlərinin mətn kodunu tərtib etməyi və eləcə də klasın dəyişən və funksiya həddlərinə müraciət etməni örgəndik.

Bu klasslar barəsində bilməli olduğumuz zəruri anlayışlar idi.

İndi isə klasslarla bağlı digər anlayışları və istifadə qaydalarını daxil edək.

10.1 açıq və gizli həddlər

Class tipinin strukt tipindən digər fərqi də odur ki, klass tipinin həddləri açıq və gizli ola bilər.

Klasın açıq həddlərinə proqramın istənilən funsiyasından müraciət etmək olar, gizli həddlərə isə yalnız və yalnız klasın öz həddləri müraciət edə bilər.

Klasın hər-hansı həddini açıq elan etmək üçün public: ifadəsindən istifadə edirlər.

Bu zaman klasın daxilində public: ifadəsindən sonra yerləşdirilmiş həddlər açıq olurlar.

Klasın həddlərini gizli elan etmək üçün private: ifadəsindən istfadə olunur.

Klasın elanı daxilində açıq və gizli həddlərin hansı ardıcıllıqla yerləşdirilməsi ilə bağlı heç bir məhdudiyyət yoxdur.

Gəlin daxil etdiyimiz bu yeni anlayışlara aid proqram nümunəsi ilə tanış olaq.

Proqram nümunəsi, class2.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 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 duzb1;

//duz1 -in en ve uzunluguna qiymetler menimsedek
duzb1.terefler(5,8);

//duzb1 -in sahesini cap edek
std::cout<<"duzbucaqlinin sahesi = "<<duzb1.sahe()<<"\n";
return 0;
}
```

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

10.2 Konstruktor və Destruktor

Yuxarıda baxdığımız class2.cpp proqramında biz `duzbucaqli` klasından `duzb1` obyekti yaradırıq.

Daha sonra bu obyektin `en` və `uzunlug` -nu mənimsətmək üçün `terefler` funksiya həddini çağırırıq.

Obyekt yönümlü proqramlaşdırmada bir çox hallarda obyektin həddlərinin qiymətlərini obyekt yaradılarkən mənimsətmək tələb olunur.

Bunun üçün `kanstruktor` adlandırılan funksiyadan istifadə olunur, hansı ki, obyekt yaradılarkən avtomatik çağırılır.

Obyekt yaradılarkən deyəndə verilmiş klass tipindən hər- hansı dəyişənin elan olunması nəzərdə tutlur.

Misal üçün yuxarıdakı nümunə proqramda `duzbucaqli duzb1;` sətrində `duzb1` obyekti yaradılır və əgər biz `duzbucaqli` klası üçün kanstruktor təyin etsəydik onda o avtomatik çağrılardı.

10.3 Konstruktorun təyin olunması

Hər hansı klasın kanstruktorunu təyin etmək üçün bu klass daxilində həmin klasın adı ilə üst-üstə düşən funksiya təyin etmək lazımdır.

Bu funksiyaya klasın kanstruktoru deyilir.

Kanstruktor funksiyasının məqsədi klasın həddlərinin ilkin qiymətlərini mənimsətməkdir.

Gəlin yuxarıda daxil etdiyimiz class2.cpp proqramının kanstruktordan istfadə etməklə variantını daxil edək.

Proqram nümunəsi, class3.cpp

```
#include <iostream>

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

//kanstruktor
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 duzb1;
//kanstruktor avtomatik cagrilir ve heddlere qiymetler menimsedir
//bunu sahe funksiyasini cagirmala yoxlaya bilerik

std::cout<<"duzbucaqlinin sahesi = "<<duzb1.sahe()<<"\n";

//duzb1 -in tereflerine ayri qiymetler menimsedek
duzb1.terefler(5,8);

std::cout<<"duzbucaqlinin sahesi = "<<duzb1.sahe()<<"\n";
return 0;
}
```

```
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 kanstruktorun əksinə olaraq obyekt yaddaşdan silindiyi zaman çağırılır.

Bu funksiya da klasın adı ilə eyni adlandırılır, yalnız adın əvvəlinə ~ işarəsi artırılır.

Destruktorlar əsasən dinamik yaradılan obyektlər (ünvan tipli klass dıyişənləri) yaddaşdan silinərkən avtomatik çağırlır, lakin əgər biz obyekti özümüz istədiyimiz vaxt məhv etmək istəsək onda onun destruktorunu çağırırıq.

10.5 Obyektlərin dinamik yaradılması və silinməsi

Növbə çatdı ünvan dəyişənləri barəsində örgəndiyimiz biliklərin klasslara tətbiqinə.

Onu deyim ki, institutda bu dərsi müəllim elə çətin dildə izah eləmişdi ki, mən düzün desəm indi ***Obyektlərin dinamik yaradılması və silinməsi*** kəlməsini xatırlayanda canıma qorxu düşür.

Əslində isə proqramçı C++ dilinin bütün gözəlliyi və sadəliyini məhs dinamik obyektlərlə işləyən zaman hiss eləyir və nəticədə bu dilə olan məhəbbəti daha da artır.

$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 klasından 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 həddlərinə müraciət etmək üçün `.` yox `->` operatorundan istifadə olunur.

Proqram nümunəsi, class4.cpp

```
#include <iostream>

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

//kanstruktor
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 *duzb1;
//heleki yaddashda duzb1 ucun hec bir yer ayrilmiyib

//duzb1 -i yaradaq
duzb1 = new duzbucaqli;
//kanstruktor avtomatik cagrilir

std::cout<<"duzbucaqlinin sahesi = "<<duzb1->sahe()<<"\n";

//duzb1 -in tereflerine ayri qiymetler menimsedek
duzb1->terefler(5,8);

std::cout<<"duzbucaqlinin sahesi = "<<duzb1->sahe()<<"\n";
return 0;
}
```

```
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 proqramlarada

`#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 veriri 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 olunduğu başlıq fayllar və bu dəyişən və funksiyalara müraciət edən proqram kodu faylları.

Proqram 1.

menim_faylim.h faylının mətni

```
/*bashliq 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` sətrini 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` direktivlədən istifadə olunaraq yaradılır.

Şərt makrosları bizə imkan verir ki, müəyyən şərtdən asılı olaraq proqramın hər-hansı hissəsinin kompilyator tərəfindən nəzərə alınmamasını təmin edir.

Sintaksis belədir:

```
# if şərt

 proqram kodu

#endif
```

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

Əlavələr

Əlavə A – bəzi standanrt funksiyalar

sdt::cout funksiyası.

`sdt::cout` funksiyası yaddaşın müxtəlif məlumatları ekrana çap etmək üçün istifadə olunur.

Misal üçün əgər ekranda "Salam dunya" sətrini çap etmək istəyiriksə onda aşağıdakı kimi yazırıq:

```
std::cout<<"Salam dunya";
```

Əgər `sdt::cout` vastəsilə ekrana müxtəlif məlumatlar göndərmək istəyiriksə onda bir neçə müxtəlif məlumatı `"<<"` vastəsilə birləşdirə bilərik.

Misal 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` vastəsilə biz dəyişənlərin qiymətlərin ekrana çap edirdiksə, `std::cin` vastəsilə biz istifadəçinin klaviaturadan daxil etdiyi qiymətləri dəyişənlərə mənimsədirik.

Misal 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.

Misal üçün əgər mən `x,y,z` dəyişənlərinə istifadəçi tərəfindən daxil olunan qiymət mnimsətmək istəyirəmsə onda yaza 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.

	+0	+1	+2	+3	+4	+5	+6	+7
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
8	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
24	CAN	EM	SUB	ESC	FS	GS	RS	US
32		!	"	#	$	%	&	’
40	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7
56	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G
72	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W
88	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g
104	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL

Bəzi çalışmaların həlləri

1. Giriş

5 - ci çalışma.

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

```
#include < iostream >
int main(){
std::cout<<"Men C++ silini orgenirem \n";
}
```

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

```
#include < iostream >
int main(){
int x1,x2,x3,x4,x5,y;

/*en primitiv variant*/
std::cout<<"Zəhmət olmasa  1-ci ədədi daxil edin \n";
std::cin>>x1;
```

```
std::cout<<"Zəhmət olmasa  2-ci ədədi daxil edin \n";
std::cin>>x2;

std::cout<<"Zəhmət olmasa  3-cu ədədi daxil edin \n";
std::cin>>x3;

std::cout<<"Zəhmət olmasa  4-cu ədədi daxil edin \n";
std::cin>>x4;

std::cout<<"Zəhmət olmasa  5-ci ədədi daxil edin \n";
std::cin>>x5;

y = x1 + x2 + x3 + x4 + x5;
std::cout<<"Daxil etdiyiniz ədədlerin cemi = "<<y<<"\n";
}
```

```
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 ədədlerin cemi = 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.

```
#include < iostream >
int main(){
```

```
int ter1, ter2, ter3, perim;


std::cout<<"Zəhmət olmasa ucbucaqin 1-ci terefinin uzunlugunu daxil edin \n";
std::cin>>ter1;


std::cout<<"Zəhmət olmasa ucbucaqin 2-ci terefinin uzunlugunu daxil edin \n";
std::cin>>ter2;


std::cout<<"Zəhmət olmasa ucbucaqin 3-cu terefinin uzunlugunu daxil edin \n";
std::cin>>ter3;
perim = ter1 + ter2 + ter3;
std::cout<<"Ucbucaqin perimetri = "<<perim<<"\n";
}
```

```
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 proqram yazın ki, istifadəçidən 2 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.

```
#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";
}
```

```
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 proqram yazın ki, istifadəçidən 3 ədəd qəbul etsin və bunların ən böyüyünü çap etsin.

```
#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";
}
```

```
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 - ci çalışma.

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

```
#include < iostream >
int main(){
int i,dey;

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

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

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

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

5 - ci çalışma.

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.

```
#include < iostream >
int main(){
int i,dey;

std::cout<<"Her hansi 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";
}
```

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>prog2.exe
Her hansi 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 proqram yazın.

```
#include < iostream >
int main(){
int i;

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

```
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 proqram yazın.

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

```
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 proqram tərtib edin.

```
#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";
}
```

```
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

proqram tərtib edin.

```
#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";
}
```

```
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 istfadə etməklə iki ədədin maksimumunu hesablayan proqram tətrib edin.

```
#include <iostream>
// en_boyuk funksiyasinin 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 funksiyasinin proqram kodu
int en_boyuk ( int dey1, int dey2) {
if (dey1 > dey2)
return dey1;
else
return dey2;
}
```

```
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 funksiyadan istifadə etməklə proqram qurun və onu icra edin.

```
#include <iostream>
// cem funksiyasinin 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 funksiyasinin proqram kodu
int cem ( void) {
int i, x, y;
// y-den umimi cemi yadda saxlamaq ucun istifade edeceyik
y=0;
for (i=1; i<=10; ++i){
std::cout<<i<<" -ci ededi daxil edin\n";
std::cin>>x;
y = y + x;
}
return y;
}
```

```
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 funksiyasından istfadə etməklə proqram qurub icra edin.

```
#include <iostream>
// kvadrat funksiyasinin elani
void kvadrat (void);

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

// kvadrat funksiyasinin 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";
}
}
```

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

4 - cü çalışma.

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

```
#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 reng simvolunu daxil edin \n";
std::cin>>x>>c;

kvadrat(x,c);
}

/* kvadrat funksiyasinin proqram 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";
}
}
```

```
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 funksiyanı elə dəyişin ki, istifadəçi kvadratın içinin rəngləndiyi simvolu da daxil edə bilsin.
Bu funksiyadan istifadə etməklə proqram tərtib edib, icra edin.

```
#include <iostream>
// kvadrat funksiyasinin elani
void kvadrat (int , char, char);

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

std::cout<<"kvadratin terefinin uzunlugunu , terefinin ve daxilinin
reng simvollarini daxil edin \n";
std::cin>>x>>c>>d;
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 olunanada 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.
# # # # #
# @ @ @ #
# @ @ @ #
# @ @ @ #
# # # # #

*/
}
```

```
C:\cpp\prog2\Debug> gcc 5_5.c -o 5_5
C:\cpp\prog2\Debug>
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.

```
#include <iostream>
// romb funksiyasinin elani
void romb (int , char, char);

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

std::cout<<"rombun terefinin uzunlugunu , terefinin ve daxilinin reng simvollarini daxil edin \n";
std::cin>>x>>c>>d;
romb(x,c,d);
}

// romb funksiyasinin proqram kodu
void romb (int teref, char t_reng, char d_reng) {
/*
  eger istifadeci 6 , =, +  parametrlerini 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";
}
```

```
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.

```
#include <iostream>
// ucbucaq funksiyasinin elani
void ucbucaq (int , char, char);

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

std::cout<<"ucbucaqin terefinin uzunlugunu , terefinin ve daxilinin
reng simvollarini daxil edin \n";
std::cin>>x>>c>>d;

ucbucaq(x,c,d);
}

// ucbucaq funksiyasinin proqram kodu
void ucbucaq (int teref, char t_reng, char d_reng) {
/*
  eger istifadeci 5 , &, =  parametrlerini 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";
}
```

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

```
#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 ve daxilinin
reng simvollarini daxil edin \n";
```

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


std::cout<<"Ekranda cekmek istediyiniz fiqurun nomresini daxil
edin\n";
std::cout<<"Ucbucaq ucun 1, romb ucun 2, kvadrat ucun 3 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 olunanada 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 , =, +  parametrlerini 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 proqram kodu */

void ucbucaq (int teref, char t_reng, char d_reng) {

/*
  eger istifadeci 5 , &, =  parametrlerini 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";
}
```

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


Fiqurun terefinin uzunlugunu , terefinin ve daxilinin reng simvollarini
daxil edin
9 - k
Ekranda cekmek istediyiniz 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.

```
#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 funksiya - proqramin esas funksiyasi
*/
int main(int argc, char *argv[]){
int k;

hazirliq_ishleri(&k);
/* gorduyunuz kimi men daxil etdiyim variantda
   ilkin_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 olunanada 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 , =, +  parametrlerini 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 proqram kodu */
```

```
void ucbucaq (int teref, char t_reng, char d_reng) {

/*
  eger istifadeci 5 , &, =  parametrlerini 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<<"Ekranda cekmek istediyiniz 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);
}
     }
```

```
C:\cpp\prog2\Debug>./prog2.exe
Ekranda cekmek istediyiniz 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 istediyiniz 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 istediyiniz 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 proqramını ele dəyişin ki, istifadəçinin daxil etdiyi ədələrin içində ən kiçiyini tapsın.

```
#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 hansi 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";
}
```

```
C:\cpp\prog2\Debug>./prog2.exe
100 -den kicik her hansi 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 proqram qurun ki, istifadəçinin daxil etdiyi ədədləri artan sıra ilə düzsün.

```
#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 hansi 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";
}
```

```
C:\cpp\prog2\Debug>./prog2.exe
100 -den kicik her hansi 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 funksiyadan 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.

```
#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 hansi 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;
}
```

```
C:\cpp\prog2\Debug>./prog2.exe
100 -den kicik her hansi 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 proqram qurun ki, istifadəçinin daxil etdiyi sətrin 5-ci simvolu ilə 15-ci simvolu arasında
qalan hissəsini çap etsin. Əgər sətrin uzunluğu 20-dən kiçik olarsa onda ekranda bu barədə məlumat çap etsin.

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

int main(int argc, char *argv[]){
 /*1024 elementden ibaret setir elan edirik*/
char  setir[1024];

std::cout<<"Her hansi setir daxil edin\n";
std::cin>>setir;

if (strlen(setir) < 20)
std::cout<<"Sizin daxil etdiyiniz setrin uzunlugu 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";
   }
}
```

```
C:\cpp\prog2\Debug>./prog2.exe
Her hansi 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 proqram tərtib edin ki, istifadəçidən 6 sətir qəbul etsin və bu sətirləri ardıcıl birləşdirərək tam sətir kimi çap etsin.

```
#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<<"Uzunlugu 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 yeniliyirik*/
memset(bufer,0,20);
    }
std::cout<<"\n"<<setir<<"\n";
}
```

```
C:\cpp\prog2\Debug>./prog2.exe
Uzunlugu 20-dan kicik olan 6 setir daxil edin
aaa
bbb
ccccc
dddddd
fffffffff
ggggggg

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

3 - cü çalışma.

Elə proqram tərtib edin ki, istifadəçidən 5 sətir qəbul etsin və bu sətirləri daxil olma sırasının
əksi ardıcıllığında birləşdirərək tam sətir kimi çap etsin.

```
#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<<"Uzunlugu 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";

}
```

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Uzunlugu 20-dan kicik olan 5 setir daxil edin
qqq
wwww
eeee
rrrrr
tttt

ttttrrrrreeeewwwwqqq
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** simvollu sətir tipli **soz** dəyişənlərindən ibarət olan **str** adlı yeni **struct** tipi yaradın.
Bu yeni yaratdığınız tipdən **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.

```
#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;
}
```

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

2 - ci çalışma.

Yuxarıdakı məsələdə `str` tipindən ünvan tipli `str_gst` dəyişəni elan edinvə məsələnin tələblərin yerinə yetirin.

```
#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;
}
```

```
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.

```
#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;
}
```

```
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 heddlerini daxil edin
6 ramin
strlerin 3 -ci elementinin x ve soz heddlerini daxil edin
7 elif
strlerin 4 -ci elementinin x ve soz heddlerini daxil edin
56 mithat
strlerin 5 -ci elementinin x ve soz heddlerini daxil edin
23 taleh
strler -in elementlerinin x ve soz heddleri
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, `strler` cərgəsinin elementlərinə mənimsədəcək, `cap_et` isə `strler` cərgəsinin elementlərinin qiymətlərini çap edəcək.

```
#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 heddlerini
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";
}
```

```
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 gilas
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 - gilas
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 elemntlərinə mənimsətsin.
Daha sonra `max_el` funksiyası tərtib edin, hansı ki, `strler` cərgəsinin elementləri arasında `x-i` ən böyük olanın qiymetlerini (`x` və `soz`) çap etsin.

```
#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];
}
```

```
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. Proqram 2 nümunəsini elə dəyişin ki, siyahının obyektləri özündə
`int` tipli `x` həddindən əlavə `char` tipli `ad[20]` həddi də saxlasın.
Müvafiq olaraq siyahi yaratma və cap_et funksiyalarinda lazımi dəyişikliyi edin.
Proqramı kompilyasiya və icra edin, siyahi yaradin , onun elementlərini çap edin.

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

int main(int argc, char *argv[]){
/* istifade edeceyimiz deyishenleri ilan edirik */
syh_el *menim_syh;
menim_syh = NULL;  /* siyahinin bosh oldugunu bildirmek ucun */
int say;
std::cout<<"Siyahinin elementlerinin sayini daxil edin \n";
std::cin>>say;

menim_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;
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";
}
```

```
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
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 5 -ci elementinin x ve ad heddlerini daxil edin
35 musiqi
```

```
siyahinin 6 -ci elementinin x ve ad heddlerini daxil edin
68 veten
Siyahinin elementleri
(12 kitab) -> (34 defter) -> (456 idman) -> (124 hefte) -> (35
musiqi) -> (68 veten)
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.

```
#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);

menim_syh = NULL;  /* siyahinin bosh oldugunu bildirmek ucun */
int say;

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

```
std::cin>>say;


menim_syh=siyahi_yarat(menim_syh,say);


std::cout<<"Siyahidan ad heddini tapmaq istediyiniz elementin x
heddini  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;
}
```

```
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 ad heddini tapmaq istediyiniz elementin x heddini daxil
edin
79
x heddi 79 -ye beraber olan elementin ad heddi shebeke -dir
C:\cpp\prog2\Debug>
```

3 - cü çalışma

İki siyahını birləşdirən funksiya tərtib edin. Funksiya parametr olaraq
iki siyahi qəbul edir, daha sonra bu siyahilardan birincini ikinci ilə birləşdirir.

```
#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);
void birleshdir(syh_el *, syh_el *);
void siyahini_cap_et(syh_el *);
```

```
int main(int argc, char *argv[]){
/* istifade edeceyimiz deyishenleri ilan edirik */
syh_el *syh1, *syh2 = NULL;

int x;
char bufer[30];
memset(bufer,0,30);

syh1 = syh2 = NULL;  /* siyahilarin bosh oldugunu bildirmek ucun */
int say;

std::cout<<"Birinci siyahinin elementlerinin sayini daxil edin \n";
std::cin>>say;
syh1=siyahi_yarat(syh1,say);

std::cout<<"Ikinci siyahinin elementlerinin sayini daxil edin \n";
std::cin>>say;
syh2=siyahi_yarat(syh2,say);

/*syh1 -i syh2 ilebirleshdiririk*/
birleshdir(syh1,syh2);
siyahini_cap_et(syh1);

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 birleshdir(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";
}
```

```
C:\cpp\prog2\Debug>
C:\cpp\prog2\Debug>./prog2.exe
Birinci siyahinin elementlerinin sayini daxil edin
3
siyahinin 1 -ci elementinin x ve ad heddlerini daxil edin
12 vaxt
siyahinin 2 -ci elementinin x ve ad heddlerini daxil edin
34 saniye
siyahinin 3 -ci elementinin x ve ad heddlerini daxil edin
45 elifba
Ikinci siyahinin elementlerinin sayini daxil edin
2
siyahinin 1 -ci elementinin x ve ad heddlerini daxil edin
56 telim
siyahinin 2 -ci elementinin x ve ad heddlerini 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 Proqram 3 -də daxil edilmiş silmə funksiyasın elə dəyişdirin ki, siyahının verilmiş indeksli obyektindən başlayaraq verilmiş sayda obyekti siyahıdan silsin.

```
#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 edeceyimiz deyishenleri ilan edirik */
```

```
syh_el *menim_syh;
menim_syh = NULL;  /* siyahinin bosh oldugunu bildirmek ucun */
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<<"Siayhidan silmek istediyiniz elementlerin \nbashlangic
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 istediyimiz 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 istediyimiz yerin bashlangicina istinad edir
   p -ni bu elementden bashlayaraq say qeder sona surushdurmeliyik
```

```
*/
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-den bashlayaraq 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;
}
```

```
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)
Siayhidan silmek istediyiniz 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

1. Statik dəyişənlər

Biz 4-cü paraqrafda dəyişənləri statik və dinamik olaraq 2 qrupa ayırdıq.

Bu zaman statik dedikdə biz adi qaydada elan etdiyimiz dəyişənləri nəzərdə tuturduq. Biz statik əvəzinə adi sözündən çaşqınlıq yaranmaması və dinamik elan olunan dəyişənlərdən fərqi daha qabarıq göstərmək üçün istifadə etdik.

Prinsip baxımından biz düzgün ifadə işlədirik "statik", yəni dinamik yaradıla və silinə bilmir. Amma C++ dilində *statik* xassəsi ilə elan olunan dəyişənlər də mövcuddur ki, digər məqsədlər üçün istifadə olunurlar.

C++ dilində dəyişənlər statik, const v.s. bəzi xassələrlə də elan oluna bilər.

Bunların hər birinin özünəməxsus özəllikləri var.

Kitabın hazırki versiyasında bu məsələrə toxunmağa hələlik imkan olmadı, İnşaallah növbəti versiyalarda bu deyilənlər barədə müvafiq məlumatlar, bölmələr əlavə olunar.