C++ proqramlaşdırma dili

Əhməd Sadıxov

3 – cü buraxılış

10.02.2013

Önsöz

C++ proqramlaşdırma dili kitabının 3-cü buraxılışında hamınızı xoş gördük. Bu dərslik C++ proqramlaşdırma dilini tam başlanğıcdan, heç bir proqramlaşdırma biliyi olmayan səviyyədən orta təcrübəli səviyyəyə qədər sərbəst örgənənlər üçün nəzərdə tutulub. 3-cü buraxılış əvvəlki buraxılışlardan məzmun, struktur v.b. məsələlərə görə ciddi fərqlənir. Bir çox başlıqlar demək olar ki, təmamilə yenidən işlənib.

C++ dilinin örgənilməsini çətinləşdirən ən əsas amil göstəricilərdir. Digər C++ dərsliklərində və bu dərsliyin əvvəlki buraxılışlarında göstəricilər mövzusu dərsliyin əvvələrində daxil olunur və dərslik boyu göstəricilərin digər proqramlaşdırma elementləri ilə əlaqəsi paralel izah olunur. Bütün bu isə yeni başlayanlar üçün ciddi çətinliklər yaradır. Bunları nəzərə alıb dərsliyin hazırkı braxlışında aşağıdakı dəyişiklikləri etdik. Göstəricilərlə bağlı bütün məsələlər dərsliyin əsas bölmələrindən təmamilə yığışdırıldı. Bu mövqenin yeni başlayanlar üçün ilkin proqramlaşdırma təcrübəsi toplamaqda göstəricilərin yaratdığı maneəni aradan qaldırdığı qənaətindəyik.

İlkin proqramlaşdırma təcrübəsi dedikdə dəyişənlər, operatorlar, cərgələr, sətirlər, strukt tiplər və siniflər nəzərdə tutulur. Bütün bu anlayışlarla iş təcrübəsi topldıqdan sonra artıq orta səviyyəli proqramçı öz proqramlaşdırma təcrübəsini PHP, JAVA və digər bu tipli göstərici tələb etməyən dillər istiqamətində inkişaf etdirə bilər.

Göstəricilərlə bağlı bütün məsələlər 9 – cu paraqrafda daxil edilir. Burada göstəricilər, onların digər proqramlaşdırma elementləri ilə əlaqəsi, göstəricilər üzərində hesab əməlləri, göstəricilərin funksiyalarla əlqəsi v.s. məsələlər tam ətraflı izah olunur. Etiraf etməliyəm ki, göstəricilər hal-hazırda istifadəçi proqramlaşdırmasında(məlumatlar bazası, web v.s.) demək olar ki istifadə olunmur və göstəricilər mövzusunu örgənmək bu halda sizə sadəcə proqramın yaddaşı ilə bağlı müəyyən qədər təcrübə qazandıracaq. Lakin gələcəkdə Unix sistemlərinə keçmək istəyən proqramçılar(sistem proqramlaşdırma) üçün göstəriciləri örgənmək mütləq vacibdir.

Bu buraxılışda digər bir yenilik isə Buta_PM proqramlaşdırma mühitinin hazırlanması oldu. C++ və digər dilləri örgənmək istəyənlərin ən başlanğıcda qarşılaşdıqları problem düzgün kompilyatorun seçilməsi, quraşdırılması və istifadəsi problemidir. Bir qayda olaraq yeni başlayan proqramçılar bu mərhələdə təcrübə, dil, v.s. problemlər səbəbindən hal-hazırda geniş istifadə olunan və daha çox peşəkar proqramçılar üçün nəzərdə tutulmuş kompilyatorların quraşdırılmasında çətinlik çəkirlər. Bunları nəzərə alıb biz yeni başlayanlar üçün Azərbaycan dilində olan, çox asanlıqla quraşdırılan və olduqca sadə istifadəçi interfeysinə malik Buta_PM C++ proqramlaşdırma mühitini tərtib etdik. Buta_PM proqramlaşdırma mühitini ***asadikhov.net*** səhifəsindən yükləyə bilərsiniz.

Müəllif hüquqları:

Əgər sizin C++ proqralaşdırma təcrübəniz varsa və kitabın gələcək versiyalarının hazırlanmasında köməklik göstərmək istəyirsinizsə müəlliflə aşağıdakı ünvandan əlaqə saxlaya bilərsiniz.

ahmed.sadikhov@gmail.com

Mündəricat

$1 Giriş.

Bu mətndə C++ dilində proqram tərtibindən bəhs olunur. Bu mətndən istifadə edə bilmək üçün ilkin olaraq heç bir proqramlaşdırma dilini bilmək tələb olunmur.
Hər bir paraqrafın sonunda verilmiş 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 Unix sistem proqramlaşdırmanı örgənmək istəyənlər $9 – Göstəricilər bölməsinə xüsusi ilə diqqət yetirməlidirlər.

1.1 Ümumi mənzərəyə baxış.

C++ dilinin öyrənilməsinə keçmədən öncə əvvəlcə istərdim ki, kompüter ,proqram, proqramlaşdırma dili kimi məsələlərə bir balaca aydınlıq gətirək. Bugünkü günümüzdə kompüterlərin həyatımıza inteqrasiyasının getdikcə daha da gücləndiyini hiss edirik. Kompüterlər demək olar ki, bizi hər tərəfdən əhatə edir.

Kompüterlər əsasən idarəetmə işlərində istifadə olunur. Son dərəcə dəqiq, səhvsiz və həddən artıq sürətli olmaları onların bütün keçmiş rəqiblərini durmadan sıxışdırmaqdadır. Ən maraqlı məqam isə bu qədər müxtəlif işləri görən kompüterin əsas istifadə predmetinin məlumat olmasıdır. Kompüterin hər-hansı bir işi görməsi üçün ona lazım olan ən əsas şey məlumatdır.

Məlumatsız kompüter heç-bir işi yerinə yetirə bilməz. Kompüter bütün işlərini məlumat vastəsilə həyata keçirir. Bundan əlavə günümüzdə bizə xidmət edən kompüterlər təkbaşına deyil, şəbəkə halında fəaliyyət göstərirlər. Və belə halda məlumat nəinki hər-hansı kompüterin bu və ya digər işi yerinə yetirməsini həmdə, kompüterlərin şəbəkə vastəsilə əlaqə qurmasını təmin edir.

Kompüterin məlumat üzərində gördüyü işlərin son nəticəsinin kifayət qədər rəngarəng bir spektr əhatə etməsinə baxmayaraq , ümumi halda kompüter məlumat üzərində aşağıdakı işləri həyata keçirir, bunu hər bir proqramçı əzbər bilməlidir:

1) Kompüter məlumatı qəbul edir. Kompüter məlumatı istifadəçidən, hər-hansı qurğudan və ya şəbəkədəki digər kompüterdən qəbul edə bilər.
2) Kompüter məlumatı emal edir. Kompüter məlumat üzərində müxtəlif çevrilmə, axtarış v.s. əməliyyatları apara bilər.
3) Kompüter məlumatı ötürür. Kompüter məlumatı istifadəçiyə, hər-hansı qurğuya və ya şəbəkə vastəsilə başqa bir kompüterə ötürə bilər.
4) Kompüter məlumatı yadda saxlayır.

Bütün bu deyilənlərdən isə məntiqi olaraq aşağıdakı sual yaranır: Bəs kompüter hər-hansısa bir məlumat üzərində, konkret hansı əməliyyatın aparılmalı oluğunu necə müəyyən edir?
Əlbətdə ki, heç bir kompüter, heç-bir məlumat üzərində necə gəldi, hansısa əməliyyat apara bilməz. Bütün bunlar əvvəlcədən dəqiq ölçülüb-biçilmiş, düşünülmüş, dəfələrlə test edilmiş instruksiyalar ardıcıllığı vastəsilə həyata keçirilir.
Bu instruksiyalar ardıcıllığı adlanır PROQRAM və ya PROQRAM TƏMİNATI.

Kompüterin hər-hansı bir işin görülməsi üçün tələb olunan instruksiyalar ardıcıllığını tərtib etmək isə adlanır proqramlaşdırma.

Biz dedik ki, kompüterin hər-hansı işi görməsi üçün tələb olunan ən əsas şey məlumatdır. Proqram təminatı da istisna deyil. Yəni proqram təminatı da, başqa sözlə kompüterin məlumat üzərində icra etdiyi instruksiyalar ardıcıllığı da öz növbəsində kompüter üçün bir məlumatdır, kompüterə hansı işləri görməli olduğunu bildirir. Bu nöqteyi nəzərdən biz məlumatları iki tipə ayıra bilərik : kompüter instruksiyaları və istifadəçi məlumatları.

1.2 Məlumatın ifadə olunma formaları

Tipindən, növündən, təyinatından asılı olmayaraq kompüter bütün məlumatları yalnız bir gözlə görür - 0 və 1 -lər ardıcıllığı şəklində. Misal üçün 011010101011 kimi verilmiş məlumat kompüter üçün hər-hansı ədəd, hərif, toplama əməliyyatı, hər-hansı şəklin hansısa bir nöqtəsinin rəngi v.s. ola bilər. Başqa sözlə həm kompüter proqramları, həm də istifadəçi məlumatları kompüterin yaddaşında 0 və 1 -lər ardıcıllığı şəklində verilir.

Müasir kompüterdə iki ədədin cəmini hesablayan proqramın bir hissəsinə nəzər salaq:

```
01110100 11001110 10011110 10110110
00101110 10000110 01000110 00000000
01110100 11001110 00101110 01001110
00101110 10000110 01000110 00000000
01110100 11001110 00010110 11001110
00101110 01001110 00101110 10000110
01000110 00000000 01110100 00101110
10100110 00011110 00101110 00000000
01110100 00100110 10000110 00101110
```

Bu 0 və 1 -lər kompüter prosessoru instruksiyalarının ikili ifadəsidir. İlkin vaxtlarda kompüter proqramları bu qayda ilə tərtib olunurdu. Bu zaman proqramçı kompüterin mərkəzi prosessorunun bütün instruksiyalarının ikili ifadəsini əzbərdən bilməli idi. Əlbəttdə görüntüdən də aydın olduğu kimi bu iş olduqca mürəkkəb bir prosesdir. Səhv etmə ehtimalı həddən artıq çox olması, proqramın gördüyü işi anlamağın çətinliyi v.s. proqramçıları daha asan və təkmil yollar axarmağa vadar edir. Nəticədə proqramçılar kompüter instruksiyalarının ikili formalarının özlərinin nisbətən asan başa düşdükləri formasından istifadə etməyə başlayırlar. Həmin forma adlanır assembler.

Misal üçün yuxarıdakı proqramın assembler kodunun bir hissəsi aşağıdakı kimi olar:

```
movl x, %eax
movl y, %ebx
addl %eax, %ebx
movl %ebx, z
```

Bu forma proqramçılar üçün ikili formaya nisbətən daha anlaşılan variantdır. Bu zaman proqramlar assembler dilində yazılır və daha sonra xüsusi assemblyasiya proqramları vastəsilə proqram assembler variantından ikili formaya çevrilir. Proqramın icraolunması üçün onun ikili formaya çevrilməsi mütləqdir, çünki yalnız ikili formada prosessor onu başa düşə bilər, assembler forması yalnız proqramçıların özləri üçündür.

Bir müddət sonra isə artıq assembler dili də proqramçılar üçün çətin gəlməyə başladı və

proqramçılar 3-cü nəsil dil adlandırılan proqramlaşdırma dillərini işləyib hazırladılar. Bunlara misal olaraq Paskal, C, C++ v.s. proqramlaşdırma dillərini göstərə bilərik. C dilində iki ədədin cəmi proqramı aşağıdakı kimi olar:

```
  int main(){

   int x,y,z;

    x=10;
    y=15;
    z=x+y;

}
```

Göründüyü kimi bu daha başadüşülən formadır. Proqramların mətni bu cür dillərdə yığıldıqdan sonra kompilyator adlandırılan xüsusi proqramlar vastəsilə əvvəlcə müvafiq assembler formasına, daha sonra isə ikili formaya çevrilirlər. Bir daha qeyd edim ki, hansı dildə tərtib olunmasından asılı olmayaraq proqramın kompüter tərəfindən icra edilə bilməsi üçün onun ikili formaya çevrilməsi mütləqdir.

Ümumi olaraq proqramın kompüter tərəfindən icra olunan formasına ikili kod, icraolunabilən kod və ya obyekt kod deyirlər. Proqramçılar tərəfindən tərtib və istifadə olunan yüksək səviyyəli dildə olan formasına isə proqramın mənbə kodu deyirlər. Biz bu kitabda yüksək səviyyəli dillərdən biri olan və 3-cü nəsil proqramlaşdırma dilləri arasında reytinqi daima yüksəkdə olan C++ dili ilə tanış olacayıq.

1.3 C++ dili ilə ilk tanışlıq.

Gəlin C++ dilində ilk sadə proqram nümunəsi ilə tanış olaq:

```
  int main ()
  {

  }
```

Gördüyünüz bu 3 sətirdən ibarət sadə kod bitkin bir kompüter proqram kodudur. Bu proqram icra olunduqda heç bir iş görmur, sadəcə proqram icraya başlayır və dərhal başa çatır.

İndi isə ikinci proqram nümunəsi ilə tanış olaq. Elə proqram tərtib edək ki, icra olunduqda nəsə bir iş görsün, misal üçün ekranda “Salam Dunya” sətrini çap etsin:

```
  #include <iostream>

  int main ()
  {

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

  }
```

1.3.1 C++ proqramlarının strukturu

C++ dilində tərtib olunmuş proqramlar müxtəlif funksiyalardan, qarışıq kodlardan, müxtəlif

mənbə fayllarında yerləşən kodlardan, hətta ayrı kompilyasiya olunmuş ikili fayllarda yerləşən funksiyalardan ibarət olur. Bütün bunlarla biz gələcəkdə məşğul olacayıq. Hələlik isə (funksiyaları örgənənə kimi) bizim baxacağımız proqramların hamısı aşağıdakı strukturdan ibarət olacaq:

```
1)  #include <iostream>

2)  int main(){

3)  PROQRAM KODU HİSSƏSİ:

4)  }
```

1) #include <iostream> sətri daxiletmə direktividir(direktivləri (10 Makroslar)bölməsində örgənəcəyik). Onun vastəsilə proqrama iostream faylı əlavə olunur harada ki, yalnız bəzi standart funksiyaların elanı yerləşir. Burada icra olunan heç bir kod yerləşdirilmədiyindən hələlik bizim üçün maraqlı deyil.

2) int main (){ sətri proqramın icraya başalama yeridir. C++ dilində tərtib olunmuş bütün proqramlar bu yerdən icra olunmağa başlayır.

3) Proqram Kodu Hissəsində biz operatorların və dəyişənlərin köməyi ilə proqramın görməli olduğu işi kodlaşdırırıq. Bura istədiyimiz qədər proqram kodu yerləşdirə bilərik və həmin kod əvvəldən sona doğru icra olunacaq.

4) } mötərizəsi proqram kodu hissəsinin sonunu bildirir.

1.3.2 C++ proqramlarının icrası

Bizim C++ dilində tərtib etdiyimiz kod, misal üçün

```
#include <iostream>

int main (){

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

}
```

kodu proqramın mənbə kodu adlanır. Bu mənbə kodu C++ dilinin sintaksis tələblərinə uyğun yazılmış adi mətndir və o icra oluna bilməz. Lakin xüsisu sistem proqramlar vastəsilə biz bu mətndən icra oluna bilən proqram kodu alırıq. Həmin xüsusi sistem proqramlar adlanır kompilyator. Proqramın mənbə kodundan icraolunabilən proqram kodunun alınması prosesinə isə deyirlər kompilyasiya və ya proqramın kompilyasiyası.

Deməli biz C++ dilində tərtib etdiyimiz proqram kodunu icra etmək üçün əvvəlcə onu kompilyator vastəsilə kompilyasiya edib ondan icraolunabilən proqram kodu almalıyıq. Daha sonra artıq icraoluna bilən proqram kodunu adi proqramları icra etdiyimiz kimi icra edə bilərik (üstündə 2 dəfə klik etməklə və ya proqramın adını terminaldan daxil etməklə).

1.3.3 C++ İDE – İİM -in quraşdırılması

C++ proqramlarının mənbə kodunu tərtib etmək, kompilyasiya etmək və icra etmək üçün müvafiq olaraq mətn redaktoru, kompilyator və əmrlər pəncərəsi (kansol) tələb olunur. Bir çox hallarda bu üç komponenti bir-birinə inteqrasiya edib vahid proqramlaşdırma mühiti yaradırlar. Bu cür proqramlaşdırma mühiti xarici ədəbiyyatlarda İDE adlanır, açılışı İntegrated Development Environment deməkdir.

Bizim dilə İİM kimi tərcümə etmək olar, İnteqrəolunmuş İnkişafetdirmə Mühiti. Hal-hazırda geniş istifadə olunan müxtəlif C++ proqramlaşdırma İDE -ləri mövcuddur. Bu cür İDE -lər əsasən təcrübəli proqramçılar üçün nəzərdə tutulduğundan (professional tətbiqlər hazırlamaq üçün) onlardan istifadə yeni başlayanlar üçün müəyyən çətinliklər yaradır.

Bu çətinliyi aradan qaldırmaq məqsədilə kitabın müəllifləri yenibaşlayanlar üçün istifadəsi və quraşdırılması asan olan sadə interfeysli Buta_PM adlı C++ İDE tərtib etmişlər. Həmin İDE -ni ***asadikhov.net*** səhifəsindən yükləyə bilərsiniz.

1.4 İlk proqramlaşdırma təcrübəsi - Ekranda çap etmə

Proqramlaşdırmaya təzə başlayarkən ən asan işlərdən biri ekranda nəyisə çap etməkdir. Ümumiliyi pozmadan biz də ilk proqramlaşdırma təcrübəmizə ekranda çap etməyi öyrənməklə başlayaq. Bunun üçün **std::cout** operatorundan istifadə olunur. Proqramda hər-hansı əməliyyat yerinə yetirən kod hissəsinə operator deyirlər.

std::cout operatorunun istifadə qaydası aşağıdakı kimidir

std::cout << Çap olunmalı məlumat ;

Bu qayda ilə biz ekranda sətir, simvol, dəyişənin qiyməti, dəyişənin yaddaşdakı ünvanı v.s. çap edə bilərik. Bütün bunlar sonrakı bölmələrin mövzusu olduğundan hələlik ekranda adi sətirlər çap edəcəyik. C++ dilində cüt dırnaq arasında verilən ifadə sətir kimi qəbul olunur. Misal üçün ***"Salam"***, ***"Baki"***, ***"Proqram dili"*** sətirlərə nümunədir. Beləliklə biz artıq ilk operatorumuzu tərtib edə bilərik.

std::cout << *"Salam"* ;

Bu operator icra olunduqda ekranda “Salam” sözü çap olunacaq. Bunun üçün operatorumuzu yuxarıda proqramın strukturu bölməsində izah edilən qayda ilə proqrama yerləşdirməliyik .Yekun proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main (){

      std::cout<<"Salam";

}
```

Baxdığımız bu proqramda PROQRAM KODU HİSSƏSİ std::cout<<"Salam"; operatorundan ibarətdir və biz onu tələb olunduğu kimi **int main (){** ilə **}** -nin arasına yerləşdirmişik. Əgər bu proqramı icra etsək ekranda ***Salam*** sözü çap olunar.

Çalışma 1. Ekranda "Salam dunya" sətrini çap edən proqram tərtib edin.

Həlli. Ekranda **"Salam dunya"** çap etmək üçün **std::cout << "Salam dunya" ;** operatorundan istifadə edə bilərik. Müvafiq proqram kodu aşağıdakı kimi olacaq:

```
#include <iostream>

int main (){

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

İzahı: Biz proqramımızın Proqram Kodu Hissəsində yalnız bir operator yerləşdirmişik - **std::cout << "Salam dunya" ;** operatorunu. Proqram icra olunduqda təkcə bu operator icra olunur, nəticədə ekranda **"Salam dunya"** sətri çap olunur və proqram başa çatır.

1.4.1 Yeni sətirdən çap etmə

Tutaq ki, biz std::cout operatoru ilə ekranda əvvəlcə **"Salam dunya"** sətrini, daha sonra isə **"Men C++ dilini orgenirem"** sətrini çap etmək istəyirik. Bunun üçün **std::cout<<"Salam dunya";** və **std::cout<<"Men C++ dilini orgenirem";** operatorlarından istifadə etməliyik. Nəticədə ekranda aşağıdakı kimi yazı çap olunar:

Salam dunyaMen C++ dilini orgenirem

Buna səbəb **std::cout** operatorunun bütün məlumatları bir-birinin ardınca eyni sətirdən çap etməsidir. Bəzən bizə elə bu formada çap etmək tələb olunur, lakin bir-çox hallarda növbəti məlumatın yeni sətirdən çap edilməsi tələb olunur, aşağıdakı kimi:

Salam dunya
Men C++ dilini orgenirem

Bu zaman bu iki məlumat arasında yeni sətir simvolu çap olunmalıdır. Nəticədə hazırki sətirin yerdə qalan boş hissəsi buraxılır və çap olunma növbəti sətirdən davam edir. C++ dilində yeni sətir simvolu **"\n"** kimi işarə olunur. Yeni sətir simvolunu **std::cout** operatoru vastəsilə aşağıdakı kimi çap edə bilərik:

std::cout << "\n" ;

Çalışma 2. Ekranda alt-alta "Komputer Elmi" və "Suni Intellekt" sətirlərini çap edən proqram tərtib edin.

Həlli: Ekranda "Komputer Elmi" sətrini çap etmək üçün std::cout<< "Komputer Elmi"; operatorundan, "Suni Intellekt" sətrini çap etmək üçün isə std::cout<<"Suni Intellekt"; operatorundan istifadə edə bilərik. Bu iki operatoru alt-alta çap etmək tələb olunduğundan std::cout<< "Komputer Elmi"; operatorundan sonra std::cout << "\n" ; yeni sətir simvolunu çap etməliyik. Yekun proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main (){

std::cout<<"Komputer Elmi";
std::cout<<"\n" ;
std::cout<<"Suni Intellekt";

}
```

1.4.2 Birləşdirmə operatoru

Çap zamanı **std::cout** operatoru ilə yanaşı istifadə etdiyimiz digər operator birləşdirmə operatorudur. Birləşdirmə operatoru **<<** kimi işarə olunur. Onun funksiyası bir neçə məlumatı "birləşdirərək" bir **std::cout** operatoru vastəsilə çap olunmasını təmin etməkdir. İndiyə qədər baxdığımız nümunələrdə biz **std::cout** ilə cəmi bir məlumat (adi sətir) çap etdiyimizdən birləşdirmə operatorunun işi hiss olunmurdu. İndi gəlin bir **std::cout** -dan istifadə etməklə birləşdirmə operatorunun köməyi ilə bir neçə məlumatın çap edilməsi ilə tanış olaq. Birləşdirmə operatorunun istifadə qaydası aşağıdakı kimidir:

```
<<məlumat1<<məlumat2<<...<<məlumatn;
```

və ya

```
<<məlumat1
<<məlumat2
<<...
<<məlumatn;
```

Bu zaman **məlumat1**, **məlumat2** ... v.s. hamısı eyni bir axına bitişdirilir.

Çalışma 3.
Yalnız bir std::cout operatorundan istifadə edərək ekranda "Nevrokibernetika" və "Proqramlashdirma" sətirlərini ardıcıl çap edən proqram tərtib edin.

Həlli. "Nevrokibernetika" və "Proqramlashdirma" sətirlərini çap etmək üçün std::cout<< "Nevrokibernetika"; və std::cout<<"Proqramlashdirma"; operatorlarından istifadə etməliyik, lakin birləşdirmə operatorunun köməyilə ilə bu aşağıdakı kimi də yazıla bilər:

std::cout<< "Nevrokibernetika"<<"Proqramlashdirma";

Proqram aşağıdakı kimi olar:

```
#include <iostream>

int main (){

std::cout<<"Nevrokibernetika"<<"Proqramlashdirma";

}
```

Bəzən çap elədiyimiz məlumatları oxumaq asan olsun deyə bir-birindən aralı çap etmək tələb olunur. Bu zaman yalnız məsafə(probel) simvollarından ibarət sətirdən istifadə edə bilərik, misal üçün " " şəklində.

Çalışma 4. Ekranda "Nevrokibernetika" və "Proqramlashdirma" sətirlərini ardıcıl çap edən proqram tərtib edin. Sətirləri bir-birindən fərqləndirmək məqsədilə onları bir qədər aralı çap edin.

Həlli. Bunun üçün "Nevrokibernetika" sətrindən sonra məsafə buraxmalıyıq. std::cout<<" "; operatoru ilə həmin məqsədə nail ola bilərik. Daha sonra isə "Proqramlashdirma" sətrini çap etmək olar. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>
```

```
int main (){

std::cout<<"Nevrokibernetika";
std::cout<<"    ";
std::cout<<"Proqramlashdirma";

}
```

Çalışma 5. Yalnız bir std::cout operatorundan istifadə edərək ekranda "Nevrokibernetika" və "Proqramlashdirma" sətirlərini ardıcıl çap edən proqram tərtib edin. Sətirləri fərqləndirmək məqsədilə onları bir-birindən aralı çap edin.

Həlli. Bunun üçün birləşdirmə operatorundan aşağıdakı kimi istifadə edə bilərik std::cout<<"Nevrokibernetika" <<" "<<"Proqramlashdirma"; . Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main (){

std::cout<<"Nevrokibernetika" <<"    "<<"Proqramlashdirma";

}
```

Çalışma 6. Ekranda "Nevrokibernetika" və "Proqramlashdirma" sətirlərini alt-alta çap edən proqram tərtib edin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main (){

std::cout<<"Nevrokibernetika" <<"\n"<<"Proqramlashdirma";

}
```

Çalışma 7. Ekranda ardıcıl olaraq (arada boş məsafə buraxmaqla) "Elm","Tehsil","Tedqiqat","Inkishafetdirme" sətirlərini çap edən proqram tərtib edin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main (){

std::cout<< "Elm"       << "   "
         << "Tehsil"    << "   "
         << "Tedqiqat"  << "   "
         << "Inkishafetdirme" ;
}
```

Çalışma 8. Ekranda alt-alta "Elm","Tehsil","Tedqiqat","Inkishafetdirme" sətirlərini çap edən proqram tərtib edin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```cpp
#include <iostream>

int main (){

std::cout<< "Elm"       << "\n"
         << "Tehsil"    << "\n"
         << "Tedqiqat"  << "\n"
         << "Inkishafetdirme" ;
}
```

1.5 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 proqramkodundan 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).

Çalışmalar

1. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar?

```cpp
#include <iostream>

int main (){

std::cout<<"Proqramlashdirma";

}
```

2. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar?

```cpp
#include <iostream>

int main (){

std::cout<<"C++";

}
```

3. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar?

```
#include <iostream>

int main (){

std::cout<<"x = 5";

}
```

4. Ekranda "informatika" sətrini çap edən proqram tərtib edin.

5. Ekranda "İkili say sistemi" sətrini çap edən proqram tərtib edin.

6. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar?

```
#include <iostream>

int main (){

std::cout<<"CPU";
std::cout<<"\n" ;
std::cout<<"RAM";

}
```

7. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar?

```
#include <iostream>

int main (){

std::cout<<"seher";
std::cout<<"\n" ;
std::cout<<"gunorta";
std::cout<<"\n" ;
std::cout<<"axsham";

}
```

9. Yeni sətir simvolundan istifadə etməklə ekranda alt-alta

```
"Fizika"
"Riyaziyyat"
```

sətirlərini çap edən proqram tərtib edin.

10. Yeni sətir simvolundan istifadə etməklə ekranda alt-alta

```
"Fizika"
"Riyaziyyat"
"İnformatika"
```

sətirlərini çap edən proqram tərtib edin.

11. Yeni sətir simvolundan istifadə etməklə ekranda alt-alta

```
"Fizika"
"Riyaziyyat"
"İnformatika"
"Xaricidil"
```

sətirlərini çap edən proqram tərtib edin.

12. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar

```
#include <iostream>

int main (){

std::cout<<"İnter"<<"Proses";

}
```

13. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar

```
#include <iostream>

int main (){

std::cout<<"İnter"<<"    "<<"Proses";

}
```

13. Bir std::cout operatorundan istifadə etməklə ekranda eyni sətirdə bir-birindən 3 məsafə simvolu (probel) aralı

```
"Ikili"    "Say"  "Sistemi"
```

sözlərini çap edən proqram tərtib edin.

14. Aşağıdakı proqram icra olunduqda ekranda nə çap olunar

```
#include <iostream>

int main (){

std::cout<<"Sade"<<"\n"<<"Murekkeb";

}
```

15. Bir std::cout operatorundan istifadə etməklə ekranda alt-alta

```
"Ana dili"
"Edebiyyat"
```

sətirlərini çap edən proqram tərtib edin.

16. Ekranda ulduz - '*' simvolu çap edən proqram tərtib edin.

16. Ekranda ardıcıl 5 ulduz "*****" çap edən proqram tərtib edin.

17. Ekranda alt-alta 2 sətir 5 ulduz

```
"*****"
"*****"
```

çap edən proqram tərtib edin.

18. Ekranda 5 ulduzlu kvadrat çap edən proqram tərtib edin.

```
*****
*****
*****
*****
*****
```

çap edən proqram tərtib edin.

18. Ekranda 5 ulduzlu içiboş kvadrat çap edən proqram tərtib edin.

```
*****
*   *
*   *
*   *
*****
```

çap edən proqram tərtib edin.

$2 Dəyişənlər.

Bu paraqrafda proqramın ən vacib elementlərindən biri dəyişənləri örgənəcəyik. Dəyişənlər nədir, nə məqsəd üçün istifadə olunur, onlara nə cür müraciət edilir v.s. kimi məsələlərə baxacayıq.

2.1 Dəyişənlər, dəyişən tipləri.

Dəyişənlər yaddaşda hər-hansı məlumat yerləşdirmək, həmin məlumata müraciət etmək, onu dəyişdirmək v.s. üçün istifadə olunur.

Şərti olaraq dəyişənlər yadda saxladıqları məlumatın müxtəlifliyinə görə tiplərə bölünür. Ən çox istifadə olunan tiplər tam ədədlər, kəsr ədədlər və simvollar tipidir. Bu tiplərdən olan dəyişənlərdə müvafiq olaraq tam ədəd, kəsr ədəd və simvol yerləşdirə bilərik. Bundan əlavə C++ dili proqramçılara öz istədikləri yeni tip təyin etməyə və proqramda istifadə etməyə imkan verir.

2.1.1 Tam tip

C++ dilində bir neçə tam tip mövcuddur. Bunlara unsigned, short, int və long tiplərini misal gətirə bilərik. Bu tiplərin biri digərindən yeganə fərqi onlara ayrılan yaddaşın həcmidir. Yaddaş həcmi nə qədər böyük olsa o qədər də böyük ədəd yadda saxlamaq olar. short tipinin ölçüsü 2 baytdır. Bu tipdən olan dəyişənlərdə 32767 -dən böyük, –32768 -dan isə kiçik tam ədədləri yerləşdirmək olmaz. int və long tipləri yaddaşda uyğun olaraq 4 və 8 bayt yer tuturlar.

Əgər proqramda böyük ədələrdən istifadə olunmursa onda niyə yaddaşda çox yer tutan tiplərdən istifadə edib yaddaşı boş yerə israf etməliyik?!
Sadəlik xətrinə biz proqramlarımızda tam ədədlərlə işləyərkən adətən int tipinə üstünlük verəcəyik.

2.1.2 Kəsr tipi

C++ dilində kəsr tipləri float və double kimi işarə olunur. Bunların da fərqi yalnız yaddaşın həcmindədir. float yaddaşda 4, double isə 8 bayt yer tutur. Bu tiplərdən kəsr ədədlərlə işləmək üçün istifadə olunur. Tam və kəsr hissə nöqtə ilə ayrılır misal üçün 5.7, 78.9, 0.1 .

2.1.3 Simvol tipi

Simvol tipi char ilə işarə olunur və yaddaşda 1 bayt yer tutur. C++ dilində simvollar tək dırnaq - ' arasında verilir və böyük və kiçik hərflər bir-birindən fərqləndirilir, misal üçün 'A' - böyük a simvolu, 'c' - kiçik c simvolu.

2.2 Dəyişənlərin elan olunması

Biz proqramın istənilən yerində, istədiyimiz tipdən , istədiyimiz sayda dəyişən elan edə bilərik. Bunun üçün dəyişənin tipini və adını qeyd etməliyik, aşağıdakı kimi:

tip ad;

Hər-bir operatorun sonunda olduğu kimi dəyişənlərin elanı da mütləq nöqtə-vergül ilə bitməlidir. Əgər eyni tipdən bir neçə dəyişən elan etmək istəsək onları bir-birindən vergül ilə ayırmaqla eyni sətirdə elan edə bilərik, aşağıdakı kimi:

```
tip dey1, dey2, dey2;
```

2.3 Dəyişənlərin adlandırılması

Dəyişənlərə ad verərkən aşağıdakı qaydalara əməl etməliyik:

1. Dəyişən adında yalnız ingilis əlifbasının simvolları, rəqəmlər və _, $ simvollarından istifadə edə bilərik.

2. Dəyişən adı mütləq ingilis əlifbasının simvolu ilə başlamalıdır.

3. Əvvəlcədən elan olunmuş dəyişən, funksiya və operator adlarını dəyişən adı kimi istifadə etmək olmaz.

Düzgün dəyişən adlarına misal: x, y, dey, dey1, dey2, cem, deyişen, S, s, vurma v.s.

Yanlış dəyişən adlarına misal: dəy(ə simvolundan istifadə olunur), 5f (rəqəm ilə başlayır), vur%ma (% simvolundan istifadə olunur) v.s.

Qeyd: Dəyişənləri adlandırmada səhvə yol vermə yeni başlayanlar üçün tipik səhvlərdən sayılır. Nəticədə proqram uğurlu kompilyasiya olunmur.

Məsləhət: Dəyişənə ad verərkən mümkün qədər qısa və dəyişəndə yadda saxlanılan məlumata uyğun ad vermək lazımdır. Bu sizə həmin dəyişəndə hansı məlumatı yerləşdirdiyinizi yadda saxlamağa kömək edr.

2.3.1 Dəyişənlərin elan olunmasına aid misallar

Calışma 1. int tipindən x adlı dəyişən elan edin.

Həlli: Dəyişənin tipi int, adı isə x olduğundan int x; operatoru ilə int tipindən x adlı dəyişən elan etmiş oluruq.

Çalışma 2. int tipindən dey1, cem və x adlı 3 müxtəlif dəyişən elan edin.
Həlli: int dey2, cem, x;

Çalışma 3. int tipindən x,y,z, float tipindən q,r, char tipindən isə S,s, c, dd dəyişənlər elan edin.
Həlli:

```
int x,y,z;
double q,r;
char S,s,c;
```

2.4 Dəyişənə qiymət mənimsədilməsi

Dəyişən elan edərkən onun üçün yaddaşda yer ayrılır. Bu yerdə dəyişənin tipinə müvafiq məlumat yerləşdirmək olar. Bu əməliyyata dəyişənə qiymət mənimsətmə və ya qısa olaraq mənimsətmə deyilir. Dəyişənə qiymət mənimsətmək üçün müxtəlif vasitələrdən istifadə olunur. Misal üçün mənimsətmə operatorundan istifadə etməklə, cin operatorndan istifadə etməklə, dəyişənin ünvanına başqa ünvanda olan məlumatı köçürməklə. Bunlardan ilk ikisi ilə (mənimsətmə və cin operatorları) bu paraqrafda tanış olacayıq, ünvandan köçürmə qaydası ilə isə paraqraf 9) göstəricilər mövzüsunda tanış olacayıq.

2.4.1 Mənimsətmə operatoru

C++ dilində mənimsətmə operatoru bərabərlik simvolu ilə işarə olunur, = kimi. Bu operator vastəsilə dəyişənə qiymət mənimsətmək üçün aşağıdakı qaydadan istifadə olunur:

```
dəyişən = qiymət;
```

Son mütləq ; simvolu ilə bitməlidir. Dəyişənə qiymət mənimsətmədən öncə o mütləq elan olunmalıdır. Bu zaman dəyişənin əvvləki qiyməti silinir və onun qiyməti yenisi ilə əvəzlənir.

Tipik proqramlaşdırma səhvi: Yeni başlayan proqramçılar dəyişəni elan etməmiş, proqramda ona qiymət mənimsədirlər və ya elanda dəyişənə bir ad verirlər, qiymət mənisədəndə isə səhvən bir qədər fərqli addan istifadə edirlər. Nəticədə kompilyator "xxx" adlı dəyişən tanınmadı səhv mesajını çap edir.

Məsləhət: Dəyişənə qiymət mənimsədərkən onun elanı sətrini bir daha nəzərdən keçirin və adın eyni olduğuna əmin olun.

Çalışma 4. İnt tipli x adlı dəyişən elan edin. Mənimsətmə operatorundan istifadə etməklə x dəyişəninə 5 qiyməti mənimsədin.

Həlli: int tipli x dəyişəni elan etmək üçün int x; operatorundan, x dəyişəninə 5 qiyməti mənimsətmək üçün isə x = 5; operatorundan istifadə etməliyik. Yekun proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main(){

int x;

x = 5;

}
```

2.5 Dəyişənlər üzərində əməllər

Dəyişənlərdə yerləşdirilmiş məlumatları məsələnin tələbinə uyğun şəkildə emal etmək üçün onun üzərində müxtəlif əməliyyatlar icra etmək mümkündür. Bu əməliyyatları hesab,

müqaisə, köçürmə, məntiq, bit v.s. kimi qruplaşdıra bilərik.

2.5.1 Dəyişənlər üzərində hesab əməlləri

C++ dili tipindən asılı olmayaraq istənilən dəyişən üzərində aşağıdakı hesab əməliyyatlarını aparmağa imkan verir:

```
Toplama: "+",
Çıxma:   "-",
Vurma:   "*",
Bölmə:   "/",
Qalıq:   "%"
```

2.5.2 Dəyişənlər üzərində müqaisə əməlləri

```
Böyükdür:           ">"
Kiçikdir:           "<"
Böyük bərabərdir:   ">="
Kiçik bərabərdir:   "<="
Bərabərdir:         "=="
Fərqlidir:          "!="
```

2.5.3 Dəyişənlər üzərində məntiq əməlləri

```
Və       "&&"
Və Ya    "||"
Inkar    "!"
```

Hesab əməllərinin hamısının istifadə qaydası eyni olduğundan yalnız toplama əməliyyatının izahını verəcəyik.

2.5.4 Toplama əməliyyatı

Toplama əməliyyatı 2 və daha artıq dəyişənin qiymətini toplamağa imkan verir. İsitifadə qaydası aşağıdakı kimidir:

```
  dey = dey1 + dey2 + ... + deyn;
```

Burada dey1, dey2, ... , deyn dəyişənlərinin qiymətləri toplanır və yekun nəticə dey dəyişəninə mənimsədilir.

Çalışma 5. C++ dilində proqram tərtib edin. Tam tipli x,y və z adlı 3 dəyişən elan edin, x-ə 5, y-ə 12, z -tə isə x ilə y in cəmini mənimsədin.

Həlli: Əvvəlcə int x,y,z; sətri ilə tələb olunan dəyişənləri elan edək. Daha sonra mənimsətmə operatoru ilə x -ə 5, y -ə 12 qiymətlərini mənimsədək. x = 5; y = 12; . Sonda isə toplama operatoru vastəsilə x və y -in cəmini z -tə mənimsədək:
z = x + y; Müvafiq proqram kodu aşağıdakı kimi olacaq:

```
// 5 ilə 12 nin cəmini hesablayan proqram

#include <iostream>

int main(){
```

```
    // deyishenleri elan edek
    int x,y,z;

    // qiymetleri menimsedek
    x = 5;
    y = 12;

    // z -te x ile y-in cemini menimsedek
    z = x + y;

}
```

Proqramın ədədləri düzgün cəmlədiyini bilmək üçün nəticəsini yoxlamaq tələb olunur. Bunun üçün ekranda çap etmə - cout operatorundan istifadə edə bilərik. cout operatorunu keçəndə qeyd etdik ki, cout ilə ekranda sətir, dəyişən qiyməti, unvan v.s. çap edə bilərik. Bir daha yada salaq ki, C++ dilində sətir cüt dırna işarəsi arasında yerləşən ifadə hesab olunur, misal üçün "Baki", "Salam dunya","cem", "5", "5 ile 12 -nin cemi", " x = ", v.s. Bütün bunlar cüt dırnaq arasında verilmiş müxtəlif simvollar ardıcıllığıdır.

cout ilə dəyişənin qiymətini çap etdikdə həmin dəyişənin adı istifadə olunur, misal üçün cout<<x; operatoru ekranda artıq "x" somvolu deyil, x dəyişəninin qiymətini çap edəcək, hansı ki, proqramın icrası zamanı müxtəlif qiymətlər ala bilər.

Çalışma 6. Tam tipli hər-hansı dəyişən elan edin və ona bir qiymət mənimsədin. Daha sonra cout operatoru ilə dəyişənin qiymətini ekranda çap edin.

Həlli: Müvafiq proqram kodu aşağıdakı kimi olar

```
 #include <iostream>

 int main(){

    int x;

    x = 45;

    std::cout<<x;
}
```

Çalışma 7. 60 ilə 12 ədədlərinin nisbətini hesablayan proqram tərtib edin, nəticəni ekranda çap edin:

Həlli: Proqram aşağıdakı kimi olar:

```
#include <iostream>

int main(){

   int x,y,z;

   x = 12;
   y = 60;

   z = y/x;

   std::cout<<z;
}
```

2.6 Istifadəçi ilə əlaqə, cin operatoru

Yuxarıda qeyd etdik ki, dəyişənə qiymət mənimsətmənin digər üsulu cin operatorudr. cin və cout operatorları istifadəçi ilə proqram arasında əlaqə yaradır. cout operatoru proqramda olan hər-hansı məlumatı ekranda çap edir, cin operatoru isə əksinə istifadəçinin klaviaturadan daxil etdiyi məlumatı verilmiş dəyişənə mənimsədir. cin operatorunun istifadə qaydası aşağıdakı kimidir:

```
  cin::>>deyishen;
```

Çalışma . Aşağıdakı işlər görən proqram tərtib edin. Tam tipli hər-hansı dəyişən elan edin, istifadəçinin daxil etdiyi qiyməti həmin dəyişənə mənimsədin.

Həlli:

```
 #include <iostream>

 int main(){

   int x;

   std::cin>>x;

}
```

2.6.1 Proqramda cout və cin operatorlarından birgə istifadə

Əgər biz ekranda məlumat çap edə və istifadəçinin daxil etdiyi məlumatı qəbul edə biliriksə, deməli biz artıq interaktiv proqramlar, yəni istifadəçi ilə əlaqə quran proqramlar tərtib edə bilərik.

Çalışma 8. Elə proqram tərtib edin ki, əvvəlcə istifadəçidən hər-hansı ədəd daxil etməsini bildirən sətir çap etsin. Daha sonra istifadəçinin daxil etdiyi ədədi hər-hansı dəyişənə mənimsətsin. Daha sonra proqram istifadəçinin klaviaturadan daxil etdiyi qiyməti təkrar ekranda çap etsin və bu barədə müvafiq məlumat sətri çap etsin.

Həlli: Əvvəlcə istifadəçiyə hər-hansı ədəd daxil etməli olduğunu bildirməliyik. bunun üçün cout operatoru ilə ekranda "Zehmet olmasa her-hansi eded daxil edin" sətrini çap edə bilərik. cout<<"Zehmet olmasa her-hansi eded daxil edin"; . İstifadəçinin daxil etdiyi ədədi yadda saxlamaq üçün int tipli x dəyişəni elan edək, int x; . cin operatoru ilə daxil olunan qiyməti x -də yerləşdirmək üçün cin::>>x; operatundan istifadə etməliyik. Bu zaman istifadəçinin daxil etdiyi qiymət x dəyişəninə mənimsədiləcək. Daha sonra həmin qiyməti təkrar ekranda çap etmək üçün cout<<x; operatorundan istifadə etməliyik.

Proqram aşağıdakı kimi olacaq:

```
#include <iostream>

int main(){

   int x;

   // ekranda adi setir cap edirik
   std::cout<<"Zehmet olmasa her-hansi eded daxil edin";
```

```
   // istifadecinin daxil etdiyi qiymeti x-e menimsedirik
   std::cin>>x;

   //ekranda adi setir cap edirik
  std::cout<<"Siz ashagidaki qiymeti daxil etdiniz  ";

  // istifadecinin daxil etdiyi qiymeti ekranda cap edirik
  std::cout<<x;

}
```

Proqramın icrası:

```
Zehmet olmasa her-hansi eded daxil edin
45
Siz ashagidaki qiymeti daxil etdiniz  45
```

2.6.2 İki ədədin cəmi proqramı

Çalışma . C++ dilində iki ədədin cəmini hesablayan proqram tərtib edin.

Həlli: Proqram aşağıdakı kimi olacaq:

```
// iki ededin cemini hesablayan proqram

#include <iostream>

int main(){

  int x,y,z;

  std::cout<<"Iki ededin cemini hesablayan proqram \n";

  // birinci ededin daxil olmasini isteyirik
  // daxil olunan ededi x-e menimsedirik
  std::cout<<"Zehmet olmasa birinci ededi daxil edin \n";
  std::cin>>x;

  // ikinci ededin daxil olmasini isteyirik
  // daxil olunan ededi y-e menimsedirik
  std::cout<<"Zehmet olmasa ikinci ededi daxil edin \n";
  std::cin>>y;

  // x ile y -in cemini hesablayib z -te meinmsedirik
  z = x + y ;

  std::cout<<"sizin daxil etdiyiniz ededlerin cemi = ";
  // cemi ekranda cap edirik
  std::cout<<z;

  // ekranda bosh setir cap edirik
  std::cout<<" \n ";

}
```

Proqramın icrası:

```
Iki ededin cemini hesablayan proqram
Zehmet olmasa birinci ededi daxil edin
34
Zehmet olmasa ikinci ededi daxil edin
78
```

```
sizin daxil etdiyiniz ededlerin cemi = 112
```

Çalışma 9. Verilmiş ədədin kvadratını hesablayan proqram tərtib edin.

Həlli:

```
#include <iostream>

int main()
{
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 icra edək:

```
Zehmet olmasa her hansi eded daxil edin
67
67 in kvadrati = 4489
```

İzahı:
Proqramda int tipli **x** və **y** dəyişənləri elan edirik. Daha sonra **std::cin** finksiyası ilə istifadəçinin daxil etdiyi qiyməti **x** dəyişəninə mənimsədirik. y -ə x -in kvadratını mənimsədirik və çap edirik.

Çalışma 10. Elə proqram tərtib edin ki, istifadəçidən düzbucaqlının enini və uzunluğunu daxil etməsini istəsin. Daha sonra proqram düzbucaqlının sahəsini ekranda çap etsin.

Həlli:

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

Proqramı icra edək:

```
Zəhmət olmasa düzbucaqlının enini daxil edin
56
Zəhmət olmasa düzbucaqlının uzunluğunu daxil edin
23
Düzbucaqlının sahəsi = 1288
```

Çalışma 11. Proqram tərtib edin. Proqramda double tipindən 3 dəyişən elan edin. Bu dəyişənlərin ikisinə istifadəçinin daxil etdiyi qiymətlər mənimsədin, 3 -yə isə istifadəçinin daxil etdiyi birinci qiymət ilə ikincinin fərqini mənimsədin.

Həlli:

```
 /* prg_2_4.cpp  */

#include <iostream>

int main(){
   //deyishenler elan edek

    double x,y,z;

    std::cout<<"x -in qiymetini daxil edin \n";
    std::cin>>x;

    std::cout<<"y -in qiymetini daxil edin \n";
    std::cin>>y;

    // x ve y -in cemini z-te menimsedek

   z = x-y;

   // z - in qiymetini cap edek

  std::cout<<x<<" ile "<<y<<" -in ferqi  "<<z<<" -dir";

}
```

Proqramı icra edək:

```
x -in qiymetini daxil edin
2.45
y -in qiymetini daxil edin
1.23
2.45 ile 1.23 -in ferqi 3.68 -dir
```

Çalışma 12. Proqram tərtib edin, proqramda simvol tipindən hər-hansı dəyişən elan edin və ona qiymət mənimsədin. Dəyişənin qiymətini çap edin.

Həlli:

```
/* prg_2_5.cpp  */

#include <iostream>

int main(){

   // simvol tipli x deyisheni elan edirik
   char x;

   // x deyishenine 'R' qiymetini menimsedirik
   x = 'R';

   // x deyisheninin qiymetini cap edirik
   std::cout<<"x = "<<x<<"\n";

}
```

2.7 Sivol tipi ilə ədədlər arasında əlaqə

C++ dilində simvol tipinə ədəd tipinin oxşarı kimi baxılır. Yəni hər bir simvolun bir ədəd qarşılığı var. Bu qarşılıq ASCİİ adlandırılan cədvəl vastəsilə verilir(bax Əlavəyə). Misal üçün 'a' simvolunun ədəd qarşılığı 97, 'R' simvolununku 82, '+' işarəsinin isə ədəd qarşılığı 43 -dür. C++ dilində somvol tipindən olan dəyişənləri tam tipli ədədlər kimi, hətta hər iki tipdən olan dəyişənləri qarışıq hesablamalarda istifadə edə bilərik.

Misal üçün qeyd etdik ki, 'a' simvolunun ədəd qarşılığı 97 -dir. Belə isə onda int tipindən hər-hansı y dəyişəni elan edib ona 3 ilə 'a' -nın cəmini mənimsədib nəticəni çap etsək ekranda 100 qiyməti çap olunmalıdır. Gəlin bunu test edək.

Çalışma 12. Proqram tərtib edin. Proqramda tam tipli y dəyişəni elan edib ona 3 ilə 'a' -nın cəmini mənimsədin. Nəticəni çap edin.

Həlli.

```
/* prg_2_6.cpp  */

#include <iostream>

int main(){

  int y;

   y = 3 + 'a';

   std::cout<<y;

}
```

2.8 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 **++**, **Dekrement** 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 qiyməti dəyişməkdirsə onda işarəni sağda və ya solda yazmağın fərqi yoxdur. Lakin əgər inkrement və ya dekrement hansısa ifadənin daxilindədirsə onda sağ,solun fərqi var. Belə ki, işrə solda olanda ifadədə dəyişənin ilkin qiyməti, sağda olanda isə 1 vahid dəyişdirilmiş yeni qiyməti hesablanır.

Çalışmalar.

1. İstifadəçinin daxil etdiyi ədədin kvadratını ekranda çap edən proqram tərtib edin.

2. İstifadəçinin daxil etdiyi 2 ədədin fərqini çap edən proqram tərtib edin.

3. 'A', 'c', 'F' simvollarının ədəd qarşılığını ekranda çap edən proqram tərtib edin.

4. 77, 81, 116 ədədlərinin ASCİİ simvol qarşılığını ekranda çap edən proqram tərtib edin.

5. İstifadəçinin daxil etdiyi 3 ədədin cəmini hesablayan proqram tərtib edin.

6. İstifadəçinin daxil etdiyi 3 ədədin hasilini hesablayan proqram tərtib edin.

$3 Operatorlar.

Proqramda hər-hansı əməliyyat icra edən kod hissəsinə operator deyirlər. Əvvəlki paraqrafda mənimsətmə, hesab, çap v.s. operatorlar ilə tanış olduq. Bu paraqrafda isə şərt, dövr və seçim operatorları ilə tanış olacayıq.

3.1 Şərt operatoru.

Şərt operatoru proqramda hər-hansı əməliyyatı müəyyən şərtdən asılı olaraq yerinə yetirməyə imkan verir. Misal üçün proqramda bizə istifadəçinin daxil etdiyi şifrənin düzgünlüyün yoxlamaq lazım ola bilər və istifadəçinin şifrəni düzgün daxil edib-etməməsindən asılı olaraq proqram ona resurslardan istifadəyə icazə verməlidir, və ya tam əksinə istifadəçinin sistemə daxil olmasının qarşısını almalı və inzibatçını insidentlə bağlı məlumatlandırmalıdır. Bu zaman proqramın atacağı addım, yəni icra olunmalı kodun istiqaməti şərt operatoru ilə tənzimlənir.

Şərt operatorunun sintaksis aşağıdakı kimidir:

```
if (Şərt)
    { Proqram kodu }
```

```
else
    { Digər proqram kodu }
```

Bu zaman əgər Şərt ödənərsə onda Proqram kodu yerinə yetirilər, əks halda isə Digər proqram kodu icra olunacaq. Misal üçün yuxarıda daxil etdiyimiz məsələni proqramlaşdırmaq istəsək, kod belə olar:

```
if (İstifadəçi_şifrəni_düzgün_daxil_edib)
   {  Ona Sistemdən İstifadəyə icazə ver;   }
else
   {  Sistemdən istifadəni qadağan elə;
      Inzibatçını məlumatlandır;  }
```

Sadə proqram nümunəsi ilə tanış olaq.

Çalışma 1. Elə proqram tərtib edin ki, istifadəçidən hər-hansı ədəd daxil etməsini istəsin. ƏGƏR İSTİFADƏÇİNİN DAXİL ETDİYİ ƏDƏD 10 -DAN BÖYÜKDÜRSƏ, onda ekranda "BOYUKDUR" sətrini çap etsin, ƏKS HALDA isə "KİCİKDİR" sətrini çap etsin.

Həlli. Əvvəlcə ekranda bildiriş sətri çap edək ki, istifadəçi nə iş görməli olduğunu bilsin. Misal üçün cout ilə "Zehmet olmasa her-hansi eded daxil edin" sətrini çap etməklə istifadəçiyə ədəd daxil etməli olduğun bildirərik. Daha sonra cin ilə istifadəçinin daxil etdiyi ədədi əvvəlcədən elan etdiyimiz x dəyişəninə yerləşdirərik. Daha sonra if operatorunun köməyi ilə x -in 10 -dan BÖYÜK olub-olmadığını yoxlayarıq. ƏGƏR BÖYÜKDÜRSƏ onda ekran "BOYUKDUR" sətrini çap edərik, ƏKS HALDA isə "KICIKDIR" sətrini çap edərik. Burada x -in 10-dan böyük olması şərtini yoxlamaq üçün müqaisə operatoru olan "böyükdür" - ">" operatorundan istifadə edəcəyik, aşağıdakı kimi:

```
   if ( x > 10 )
       { std::cout<<"BOYUKDUR";  }
   else
       { std::cout<<"KICIKDIR";  }
```

Proqram kodu:

```
   #include <iostream>

   int main(){

      int x;

      // istifadeciden eded daxil etmesini xahish edek
      // ve ededi x-e yerleshdirek
      std::cout<<"Zehmet olmasa her-hansi eded daxil edin";
      std::cin>>x;

      // sherti yoxlayaq ve teleb olunan kodu icra edek
      if ( x > 10 )
         { std::cout<<"BOYUKDUR";  }
      else
         { std::cout<<"KICIKDIR";  }
}
```

İzahı: Proqram icra olunanda istifadəçidən hər-hansı ədəd daxil etməsini xahiş edəcək.

İstifadəçi daxil edən ədəd x -də yerləşdiriləcək. Daha sonra if operatoru ilə x-in 10-dan böyük olma şərti yoxlanacaq. Əgər istifadəçi 10-dan böyük ədəd daxil edərsə, misal üçün 11 onda if operatorunun şərti (x>10) ödənir və bu hal üçün nəzərdə tutulmuş kod icra olunur (Ekranda "BOYUKDUR" sətri çap olunur) else -nin daxilində verilmiş kod icra olunmadan proqram bitir. Əksə halda isə, yəni istifadəçi 10-dan kiçik ədəd daxil edərsə onda else -də gösətrilən operator icra olunar cout<<"KICIKDIR"; .

Şərt operatorundan istifadə edərkən hər-iki halı nəzərə almaq vacib deyil, yəni yalnız verilmiş şərtin doğruluğunu yoxlamaq tələb olunursa onda sadəcə

```
if (şərt)
    {proqram kodu}
```

sintaksisindən istifadə olunur. Bu zaman if operatoru daxilində verilmiş proqram kodu yalnız şərt doğru olduqda icra olunacaq.

3.1.1 Müqaisə operatorları

if operatorunun şərtini tərtib edərkən müqaisə operatorlarından istifadə olunur. C++ dilində aşağıdakı müqaisə operatorları təyin olunub:

```
Böyükdür:           ">"
Kiçikdir:           "<"
Böyük bərabərdir:   ">="
Kiçik bərabərdir:   "<="
Bərabərdir:         "=="
Fərqlidir:          "!="
```

Yuxarıdakı nümunədə > operatorundan istifadəyə aid proqram nümunəsi ilə tanış olduq. Digər müqaisə oparatorlarının istifadəsi də analojidir.

Çalışma 2. İstifadəçinin daxil etdiyi ədədin cüt və ya tək olduğunu təyin edən proqram tərtib edin.

Həlli: Bu proqramı tərtib etmək üçün istifadəçinin daxil etdiyi ədədin cüt və ya tək olmasını müəyyən edə bilməliyik. Əgər cüt olsa onda ekranda "Cutdur", əks halda isə "Tekdir" sətrini çap edərik. Ədədin cüt və ya tək olmasını bilmək üçün onu 2-yə bölüb alınan qalığı 0 qiyməti ilə müqaisə etməliyik. Əgər ədədi 2-yə böldükdə qalıqda 0 qalırsa onda ədəd cütdür, əks halda isə tək. Şərt operatoru belə olacaq

```
if (x%2 == 0)
   cout<<"Cut";
else
   cout<<"Tek";
```

Burada x%2 x -i 2-yə bölərkən alınan qalıqdır. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main(){

   int x;

   std::cout<<"Zehmet olmasa her-hansi eded daxil edin";
   std::cin>>x;
```

```
      if (x%2 == 0)
         cout<<"Cut";
      else
         cout<<"Tek";
}
```

3.1.2 Mürəkkəb şərtlərin qurulması, məntiq əməliyyatları

```
Və       "&&"
Və Ya    "||"
Inkar    "!"
```

Yuxarıdakı misallarda if operatorunda biz yalnız bir şərtin ödənib-ödənmədiyini yoxladıq. Misal üçün if (x > 10) ..., if (x % 2 == 0) ... v.s. Bir çox hallarda isə bizə bir neçə şərtin eyni anda ödənməsini yoxlamaq tələb olunur. Misal üçün

```
 əgər ( (Sabah yağış yağacaq) Və YA (Külək əsəcəksə) )  onda
     { Gödəkcə geyin }
 əks halda
     { Gödəkcə geyinmə  }
```

Bu misalda biz VƏ YA məntiq operatorunun köməyi ilə iki şərtin eyni anda ödənməsini yoxladıq. Məntiq operatorları ilə istənilən sayda şərti tələb olunan məntiqi ardıcıllıqla birləşdirib yekun mürəkkəb şərt ala bilərik.

Misal üçün

```
   əgər ( (Baki paytaxtdır) Və  (6 3-ə bölünür) Və ya ( 10 20 -dən kiçikdir))
      onda   {ekranda çap elə "Sabah hava isti olacaq"}
```

3.1.3 Və operatoru

Və operatoru bütün şərtlərin ödənməsini tələb edir. Misal üçün əgər bir neçə şərti Və operatoru ilə aşağıdakı kimi birləşdirmişiksə,

```
     əgər ( şərt1 Və şərt2 Və ... Və  şərtn  ) onda
               { Proqraq Kodu  }
```

Bu zaman Proqram Kodunun icra olunması üçün Şərt1, Şərt2, ... Şərtn -in hamısı DOĞRU olmalıdır, ödənməlidir. Əgər bu şərtlərdən heç olmasa biri YALAN qiyməti alarsa onda yekun şərt yalan qiyməti alacaq, nəticədə isə Proqram Kodu icra olmayacaq.

Misal üçün

((Bakı Paytaxtdır) VƏ (2 böyükdür 5)) mürəkkəb şərtinin qiymətnini hesablayaq. Bu mürəkkəb şərt 2 sadə şərtin VƏ operatoru ilə birləşməsindən ibarətdir. Bakı Paytaxtdır şərti doğru qiymət alır, 2 böyükdür 5 şərti isə yalan. Yekun nəticə YALAN qiyməti alır.

Və operatoru C++ dilində && kimi işarə olunur. Misal üçün şərt1 && şərt2 kimi.

Çalışma 3. İstifadəçinin daxil etdiyi ədədin 5 ilə 15 arasında olduğunu müəyyənləşdirən proqram tərtib edin.

Həlli. Bu məsələni həll etmək üçün biz istifadəçinin daxil etdiyi ədədin həm 5 -dən böyük,

həm də 15 -dən kiçik olduğunu yoxlamalıyıq. Tutaq ki, istifadəçinin daxil etdiyi ədədi yadda saxlamaq üçün x dəyişənindən istifadə edirik. Bu zaman x -in 5-dən böyük olması şərtini (x > 5) , 15 -dən kiçik olması şərtini isə (x < 15) kimi verə bilərik. Bu iki şərtin eyni anda ödəndiyini VƏ operatoru ilə bu şəkildə yoxlaya bilərik: ((x > 5) && (x < 15)). Proqram kodu aşağıdakı kimi olar:

```
  #include <iostream>

  int main(){

    int x;

    std::cout<<"Her-hansi eded daxil edin \n";
    std::cin>>x;

    if (( x > 5) &&  ( x < 15))
        std::cout<<"Sizin daxil etdiyiniz eded 5 ile 15 arasindadir";
    else
        std::cout<<"Sizin daxil etdiyiniz eded 5 ile 15 arasinda deyil";
}
```

3.1.4 Və Ya operatoru

Və Ya operatoru Və operatorundan fərqli olaraq heç olmasa bir şərt doğru qiymət aldıqda ödənir. Misal üçün ((Bakı Paytaxtdır) Və Ya (2 böyükdür 5)) şərti doğru qiymət alır. Çünki heç olmasa bir şərt - Bakı Paytaxtdır şərti doğru qiymət alır.

Və Ya operatoru C++ dilində || kimi işarə olunur, ((Bakı Paytaxtdır) || (2 böyükdür 5)).

Çalışma 4. İstifadəçinin daxil etdiyi ədədin 5-dən kiçik və ya 15-dən böyük olduğunu müəyyənləşdirən proqram tərtib edin.

Həlli . Bu zaman iki şərtdən biri ödəndikdə yəni istifadəçinin daxil etdiyi ədən 5 -dən kiçik və ya 15 -dən böyük olduqda nəticə doğru olur. Proqram kodu aşağıdakı kimi olar.

```
#include <iostream>

  int main(){

    int x;

    std::cout<<"Her-hansi eded daxil edin \n";
    std::cin>>x;

    if (( x < 5) ||  ( x > 15))
        std::cout<<"Sizin daxil etdiyiniz eded ya 5-den kicik, ya da 15 -den
boyukdur";
    else
        std::cout<<"Sizin daxil etdiyiniz eded 5 ile 15 arasindadir";
}
```

3.1.5 İnkar operatoru

İnkar operatoru verilmiş şərti inkar edir və verilmiş şərt yalan qiymət aldıqda doğru, əks halda isə yalan qiymət alır. İnkar operatoru ! somvolu ilə işarə olunur. Misal üçün (!Bakı Paytaxtdır) şərti yalan, (!2 böyükdür 5) şərti isə doğru qiyməti alır.

3.1.6 Şərtlərlə ədədlərin əlaqəsi

C++ dilində tam tipli ədədlərdən şərt kimi isitfadə edə bilərik. Bu zaman 0 və 0-dan kiçik ədədlər yalan, 0-dan böyük ədələrə isə doğru qiyməti kimi baxılır. Başqa sözlə 0-dan böyük ədədlərə doğru qiymət alan, 0-dan kiçik ədədlərə isə yalan qiymət alan şərt kimi baxmaq və digər şərtlərlə məntiq operatorları ilə birləşdirmək olar. Misal üçün

```
    if (5 && (x>8))
         std::cout<<"Bu setir cap olunacaq";
```

Proqram kodunda 5 ilə x>8 şərtləri VƏ operatoru ilə birləşir.

3.2 Dövr operatorları

Dövr operatorları verilmiş əməliyyatları bir neçə dəfə təkrar yerinə yetirmək üçün istifadə olunur. C++ dilində 3 dövr operatoru istifadə olunur for, while, do while

3.2.1 for operatoru

for operatoru verilmiş əməliyyatı tələb olunan sayda, misal üçün 10, 50 , 100 dəfə təkrar yerinə yetirmək üçün istifadə olunur. for operatorunda dövrlərin sayına nəzarət etmək üçün sayğacdan istifadə olunur. Sayğac olaraq tam tipli ədədlərdən istifadə olunur.

Sintakisisi aşagıdakı kimidir:

```
for ( Sayğacın_Başlanğıc_Qiyməti; Dövrün_Sona_Çatma_Şərti; Sayğacın_Dəyişməsi)
 { Proqram Kodu; }
```

Bu zaman Sayğacın_Başlanğıc_Qiyməti, Dövrün_Sona_Çatma_Şərti və Sayğacın_Dəyişməsindən asılı olaraq Proqram Kodu təkrar yerinə yetiriləcək.

Sayğacın_Başlanğıc_Qiyməti

Bu zaman biz sayğaca ilkin qiymət mənimsətməliyik. Biz sayğaca məsələnin şərtindən asılı olaraq istədiymiz ilkin qiyməti mənimsədə bilərik. Qeyd edim ki, əgər məsələdə verilmiş kodu sadəcə n dəfə təkrar yerinə yetirmək lazımdırsa onda sayğacın ilkin qiymətinin heç bir önəmi olmur və adətən ona 0 və ya 1 qiyməti mənimsədirlər. misal üçün əgər sayğac olaraq int tipli i dəyişənindən istifadə ediriksə onda ona i = 0; operatoru ilə başlanğıc 0 qiymətini mənimsətmiş olarıq.

Dövrün_Sona_Çatma_Şərti

Dövrün sona çatma şərti adından göründüyü kimi dövrlərin sayını müəyyənləşdirir. Bu zaman sayğacın qiymətinin hansısa qiymətlə müqaisəsindən istifadə olunur.

Sayğacın_Dəyişməsi

Dövrlərin təkrarolunma sayını tənzimləyən digər parametr də Sayğacın dəyişməsidir. Bu zaman biz sayğacın qiymətinin hər-dəfə necə yenilənməsini göstəririk.

for operatoruna aid proqram nümunələri ilə tanış olaq.

Çalışma 5. for operatorundan istifadə etməklə ekranda 5 dəfə "Salam Dunya" sətrini çap edən proqram tərtib edin.

Həlli. Ekranda "Salam Dunya" sətrini çap etmək üçün std::cout operatorundan istifadə edəcəyik. for operatorunun köməyi ilə bu əməliyyatı 5 dəfə təkrar icra etməliyik. Sayğac olaraq int tipli i dəyişəni elan edək, int i; . Sayğaca başlanğıc olaraq 1 qiyməti mənimsədək, i = 1; . Əgər sayğacın qiməti hər dəfə 1 vahid artsaq və onda başa çatma şərtini 6 - dan kiçikdir kimi versək dövr 5 dəfə təkrar olunar.

Sayğacın qiymətinin hər-dəfə bir vahid artırmaq üçün i = i + 1 operatorundan istifadə edə bilərik. Sayğacın 6-dan kiçikdir şərtini isə i < 6; kimi. Yekun operator aşağıdakı kimi olar:

```
for ( i = 0; i<6; i = i + 1)
  { std::cout<<"Salam Dunya"; }
```

Qeyd edək ki, dövrün əməliyyatlarının sayı 1 -dən çox olmadıqda {} mötərəzələrindən istifadə etməyə ehtiyac yoxdur, aşağıdakı kimi:

```
for ( i = 0; i<6; i = i + 1)
  std::cout<<"Salam Dunya";
```

Yekun proqram kodu belə olar:

```
// for operatoruna aid sade proqram
#include <iostream>

int main(){

  //saygac elan edek
  int i;

  for (i=0; i<6; i=i+1)
      std::cout<<"Salam Dunya";

}
```

Çalışma 6. for dövr operatorundan istifadə etməklə 1 -dən 10 -a kimi olan ədədləri ekranda çap edən proqram tərtib edin.

Həlli. for dövr operatorundan istifadə etməklə 1-dən 10-a kimi ədədləri çap etmək üçün sayğaca başlanğıc olaraq 1 qiyməti mənimsədərik, sona çatma şərtini kiçikdir bərabərdir 10 götürərik (10 -da çap etmək üçün) , sayğacın qiymətini isə hər-dəfə 1- vahid artırarıq. Hər -dəfə dövr təkrarlandıqda sayğacın qiymətini ekranda çap edərik. Proqram aşağıdakı kimi olar:

```
#include <iostream>

int main(){

 int i;

   for (i=1; i<=10; i=i+1)
        std::cout<<i;
```

```
}
```

Çalışma 7. for operatorundan istifadə etməklə 3 -dən 7 -yə qədər olan ədədlərin cəmini hesablayan proqram tərtib edin.

Həlli. Proqram kodu belə olar.

```
#include <iostream>

int main(){

   int i,S;

     S=0;

   for (i=3; i<=7; ++i)
        S=S+i;

   std::cout<<S;

 }
```

Izahı. Proqramda int tipli i və S dəyişənləri elan edirik. i-dən sayğac, S -dən isə cəmi hesablamaq üçün istifadə edəcəyik. Sayğaca ilkin qiymət 3, son qiymət 7 veririk və hər-dəfə qiymətin inkrement ilə 1 vahid artırırıq. Nəticədə sayğac 3-dən 7-yə bütün qiymətləri alır. S -ə dövrdən əvvəl 0 qiymətini mənimsədirik. Hər-dəfə dövr təkrarlandıqda S -in qiyməti i qədər artır. Nəticədə dövrün sonunda S özündə 3-dən 7-yə qədər olan ədədlərin cəmini saxlayır.

3.2.2 while opratoru

while operatorunda sayğacdan istifadə olunmur, dövrlərin sayı while operatorunun şərti ilə müəyyən olunur. Sintaksis aşağıdakı kimidir:

```
while  ( Dövrün Başa Çatma Şərti  ){
  Proqram Kodu;
}
```

Bu zaman nə qədər ki, şərt ödənir, yəni doğru qiymət alır Proqram Kodu təkrar yerinə yetiriləcək. Dövrün Başa Çatma Şərti olaraq istənilən şərt vermək olar.

Çalışma 8. while operatorundan istifadə etməklə aşağıdakı işi yerinə yetirən proqram tərtib edin. İstifadəçi klaviturадan müxtəlif sayda ədədlər daxil edir. Ən son daxil olunan ədədin 0 olduğu məlumdur. İstifadəçinin daxil etdiyi ədədlərin cəmini hesablayın.

Həlli. Bu məsələnin həllində cəmin hesablamanın sonunu müəyyənləşdirmək üçün hər-dəfə daxil olunan ədədi 0-la müqaisə edəcəyik. Nə qədər ki, istifadəçinin daxil etdiyi ədəd 0-dan fərqli olacaq, istifadəçinin daxil etdiyi ədəd əvvəlcədən daxil olunan ədədlərin cəminin üzərinə əlavə olunmalıdır. İstifadəçinin daxil etdiyi ədədi yadda saxlamaq üçün x, onların cəmini yadda saxlamaq üçün s dəyişənindən istifadə edək. Əvvəlcə s -i 0-ra mənimsətməliyik. x -ə isə 0-dan fərqli hər-hansı bir ədəd mənimsətməliyik. x = 1; Dövrün başa çatma şərtini fərqlidir operatorundan istifadə etməklə verəcəyik x != 0; .

Proqram kodu aşağıdakı kimi olar.

```
#include <iostream>
```

```
 int main(){

     int x,s;

     s = 0;
     x = 1;

       std::cout<<""Zehmet olmasa ededleri daxil edin, sonda 0  \n;

     while ( x!=0 ){
      std::cin>>x;
      s = s + x;
    }

    std::cout<<"Sizin daxil etdiyiniz ededlerin cemi = "<<s;

}
```

3.2.3 do while operatoru

do while operatoru while operatoruna analojidir, sintaksis aşağıdakı kimidir:

```
do{
  Proqram Kodu;
} while  ( Dövrün Başa Çatma Şərti  );
```

Fərq ondadır ki, while operatorunda əvvlcə şərt yoxlanırdı sonra Proqram Kodu icra olunurdu, do while operatorunda isə əvvəlcə Proqram Kodu icra olunur sonra şərt yoxlanılır.

Calışma 9. Aşağıdakı işi yerinə yetirən proqram tərtib edin. İstifadəçi klaviaturadan müxtəlif ədədlər daxil edir. Proqram həmin ədədləri qəbul etməlidir. İstifadəçinin daxil etdyi ədəd 5 -ə bölündükdə proqram bu barədə məlumat çap etməlidir.

Həlli . Burada proqram sonsuz olaraq istifadəçidən ədədlər qəbul edir. Əgər həmin ədədlərdən hansısa biri 5 -ə bölünürsə onda dövr dayanır və ekranda məlumat çap edir. İstifadəçinin daxil etdiyi ədədin 5-ə bölünməsini yoxlamaq üçün qalıq operatorundan istifadə edəcəyik.

Proqram kodu aşağıdakı kimi olacaq:

```
#include <iostream>

 int main(){

   int x;

 do{

   std::cin>>x;

 }while ( x%5 != 0 );

std::cout<<"Sizin daxil etdiyiniz eded 5-e bolunur \n";

}
```

3.2.4 continue və break

continue və break operatorları dövr operatorlarının daxilində istifadə olunur.
break operatoru dövrdən dərhal çıxmaq üçün istifadə olunur.
continue operatoru isə dövrün yeni tsiklə keçməsi üçün istifadə olunur.

3.3 switch operatoru

switch operatoru seçim operatoru adlanır. switch operatorunun sintaksisi aşağıdakı kimidir:

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

switch 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:

Çalışma 10. switch operatorundan istifadə etməklə istifadəçi klaviaturadan 1 qiyməti daxil etdikdə ekranda "qirmizi", 2 daxil etdikdə "yashil", 3 daxil etdikdə isə "qara" sətri çap edən proqram tərtib edin. Əgər istifadəçi bu ədədlərdən fərqli rəqəm daxil etsə, ona məlumat verin.

Həlli.

```
#include <iosteram>

int main()
{
int color = 0;
std::cout<<"Zehmet olmasa 1,2 ve ya 3 qiymetini daxil edin:\n";
std::cin>>color;

switch(color){
case 1: std::cout<<"qirmizi";
break;
case 2: std::cout<<"yashil";
```

```
break;
case 3: std::cout<<"qara";
break;
default: std::cout<<"Zehmet olmasa 1,2 ve ya 3 daxil edin";
}
return 0;
}
```

Çalışmalar

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

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

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

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.

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 Funksiyalar.

4.1 Funksiya anlayışı

Funksiya adı olan kod hissəsidir hansı ki, proqramın istənilən yerindən "çağırıla" bilər. Funksiyanı çağırdıqda o icra olunmağa başlayır və işini yekunlaşdırdıqdan sonra "geri qayıdır". Bundan əlavə funksiyanı çağırarkən ona parametr də ötürmək olar.

4.2 Funksiyanın elanı

Proqramda funksiyadan istifadə edə bilmək üçün əvvəlcə onu "elan" etmək tələb olunur. Funksiyanın elanı sətrində funksiyanın adı, qəbul etdiyi parametrlərin tipi və qaytardığı nəticənin tipi göstərilir.

Funksiyanın elanı sintaksisi aşağıdakı kimidir:

```
Funksiyanın_tipi  Funksiyanın_adı (Parametrlər) ;
```

Funksiyanın_tipi

Funksiyanın_tipi funksiyanın qaytardığı nəticənin tipini bildirir. Hər-bir funksiya icra olunduqda müəyyən nəticə qaytara bilər, misal üçün ədəd, simvol, sətir. Bu zaman funksiyanın tipi olaraq ədəd, simvol, sətir v.s. tip göstərilir. Əgər funksiya heç bir nəticə qaytarmırsa bu zaman Funksiyanın_tipinin yerinə void yazılmalıdır.

Funksiyanın_adı

Funksiyanın_adı funksiyaya müraciət etmək üçün istifadə olunur.

Parametrlər

Parametrlər funksiyaya ötürülən başlanğıc məlumatların tipini bildirir. Bu zaman parametrlərin tipi və adı vergüllə ayrılmaqla sıralanır, aşağıdakı kimi:
tip1 par1, tip2 par2, v.s.

Qeyd edək ki, parametrlər sətrində əsas tiplər önəmlidir, adlar buraxıla bilər.

tip1, tip2, v.s.

Əgər funksiya heç bir parametr qəbul etmirsə bu zaman parametrlərin yerinə void yazılmalıdır.

Çalışma 1. int tipli nəticə qaytaran, adı f olan və bir dənə char tipli parametr qəbul edən funksiya elan edin.

Həlli. Funksiyanın adı f, nəticəsinin tipi int -dir. Funksiya tipi char olan bir parametr qəbul edir. Onun elanı aşağıdakı kimi olar:

```
int f ( char );
```

Çalışma 2. Aşağıdakı funksiyaların qaytardığı nəticənin tipini, adını və arqumentlərinin tipini müəyyən edin.

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

void cap_et(void);

long en_boyuk(int , long , float );
```

Həlli.

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

Funksiyanın tipi int, adı cemle -dir. Funksiya int tipli x və y adlı 2 arqument qəbul edir.

```
void cap_et(void);
```

Funksiya heç bir nəticə qaytarmır və heç bir arqument qəbul eləmir. Funksiyanın adı cap_et -dir.

```
long en_boyuk(int x, long y, float z);
```

Funksiyanın tipi long, arqumentlərinin tipi müvafiq olaraq int, long və float -dır. Funksiyanın adı en_boyuk -dür.

Çalışma 3. char tipli nəticə qaytaran, son_simvol adlı və char və int tipli arqument qəbul edən funksiya elan edin.

Həlli. Arqumentlərin adları əhəmiyyətli olmadığından elan zamanı onlara istədiyimiz kimi ad verə bilərik. Funksiyanın elanı aşağıdakı kimi olacaq:

```
char son_simvol (char c, int x);
```

4.3 Funksiyanın Proqram kodu

Funksiyanın Proqram kodu funksiya çağırılan zaman icra olunan proqram kodudur.

Funksiyanın Proqram kodunun tərtibi

Funksiyanın proqram kodunu tərtib etmək üçün aşağıdakı sintaksisdən istifadə olunur.

```
Funksiyanın_tipi  Funksiyanın_adı (parametrlər) {

    Proqram kodu
```

```
    }
```

Funksiyanın tipi, adı və parametrlər eynilə funksiyanın elanında olduğu kimidir. { } mötərizələri arasında funksiyanın icra etməli olduğu proqram kodu yerləşdirilir.

Çalışma 4. Ekranda "Salam Dunya" sətri çap edən funksiya tərtib edin.

Həlli. Funksiya icra olunduqda ekranda "Salam Dunya" sətrini çap etməlidir. Heç bir parametr qəbul etmir və heç bir nəticə qaytarmır. Elanı sətri aşağıdakı kimi olar:

```
 void cap_et (void);
```

Proqram kodu isə müvafiq olaraq aşağıdakı kimi:

```
void cap_et ( void) {

    std::cout<<"Salam dunya ";

}
```

4.4 Funksiyanın nəticə qaytarması

Funksiyanın nəticə qaytarması üçün return əmrindən istifadə olunur. Sintaksis aşağıdakı kimidir.

```
  return nəticə;
```

return operatorundan funksiyanın proqram kodunun istənilən yerində istifadə etmək olar. Nəticədə funksiya dərhal "geri" qayıdır.

Çalışma .5 cem adlı, int tipli nəticə qaytaran və int tipli iki parametr qəbul edən funksiya tərtib edin, hansı ki nəticə olaraq verilmiş arqumentlərin cəmini qaytarır.

Həlli. Funksiyanın elanı aşağıdakı kimi olar.

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

Funksiyanın proqram kodu isə aşağıdakı kimi :

```
int cem ( int dey1, int dey2){

    // funksiyanin daxilinde int tipli x deyisheni
    // elan edek
    int x;

    // arqumentlerin cemini x -e menimsedek
    x = dey1 + dey2;

    // x -i netice olaraq qaytaraq
    return x;

}
```

4.5 Funksiyanın çağrılması

Funksiyanı elan etdikdən sonra proqramın istənilən yerindən funksiyanın adına müraciət etməklə onu çağırmaq olar. Nəticədə funksiyanın proqram kodu icra olunacaq. Əgər funksiya parametr qəbul edirsə bu zaman onlar da qeyd olunmalıdır, aşağıdakı kimi:

```
funk (par1, par2 ...);
```

4.6 Proqramda funksiyadan istifadə

Çalışma 6. Çalışma 4 -də tərtib olunan cap_et funksiyasından istifadə etməklə ekranda "Salam dunya" sətrini çap edən proqram qurun.

Həlli. Proqram aşağıdakı kimi olar:

```
#include <iostream>

// evvelce cap_et funksiyasinin elan edek

void cap_et (void);

int main(){

  // funksiyani cagirmaq ucun onun adindan
  // istifade edirik
  cap_et();

}

// cap_et funksiyasinin proqram kodu

void cap_et(void) {

   std::cout<<"Salam dunya";

}
```

Çalışma 7. Çalışma 5 -də elan olunmuş cem funksiyasından istifadə etməklə 2 ədədin cəmini hesablayan proqram tərtib edin.

Həlli. Proqram aşağıdakı kimi olar:

```
#include <iostream>

// cem funksiyasinin elani
int cem (int, int);

int main(){

  // cem funksiyasina muraciet edirik
  // ve arqument olaraq 4 ve 5 qiymetlerini otururuk
  // cem funksiyasi 4 ile 5 -in cemini hesablayib
  // netice olaraq qaytarir

  std::cout<< cem(4,5);

}


// cem funksiyasinin proqram kodu
```

```
int cem ( int dey1, int dey2){

    // funksiyanin daxilinde int tipli x deyisheni
    // elan edek
    int x;

    // arqumentlerin cemini x -e menimsedek
    x = dey1 + dey2;

    // x -i netice olaraq qaytaraq
    return x;

}
```

4.7 Lokal və Qlobal dəyişənlər

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

Çalışmalar

1. Funksiyadan istifadə etməklə 2 ədədin cəmini hesablayan proqram tərtib edin.

2. Funksiyadan istifadə etməklə 2 ədədin hasilini hesablayan proqram tərtib edin.

Əgər ədəd 1 -dən və özündən başqa heç bir ədədə bölünmürsə belə ədədə sadə ədəd deyirlər.

3. Elə funksiya tərtib edin ki, verilmiş ədədin sadə olduğunu müəyyən etsin. Həmin

funksiyadan istifadə edərək 1 -dən 100 -ə qədər olan ədədlər arasında yerləşən sadə ədədləri çap edən proqram tərtib edin.

4. Funksiyadan istifadə etməklə verilmiş düzbucaqlının sahəsini hesablayan proqram tərtib edin.

5. Funksiyadan istifadə etməklə verilmiş kubun səthinin sahəsini hesablayan proqram tərtib edin.

6. Funksiyadan istifadə etməklə verilmiş radiuslu dairənin sahəsini hesablayan proqram tərtib edin.

$5 Cərgələr

5.1 Cərgə anlayışı

Cərgə dedikdə eyni tipdən olan bir neçə dəyişənin bir ad altında "cəmi" başa düşülür. Cərgədə dəyişənlər index nömrələrinə görə sıralanır və sıralanma 0 -dan başlayır. Beləliklə cərgənin ilk elementinin index nömrəsi 0, sonuncu elementinin index nömrəsi isə elementlərin_sayı - 1 olur.

5.1.1 Cərgənin elanı

Cərgə elan eləmək üçün aşağıdakı sintaksisdən istifadə olunur.

```
tip ad [ say ] ;
```

Burada tip cərgəni təşkil edən elementlərin tipini, ad cərgənin adını, say isə cərgədəki elenemtlərin sayını bildirir.

Çalışma 1. int tipindən x adlı 5 elementdən ibarət cərgə elan edin.

Həlli. Cərgənin elanı sintaksisinə əsasən tələb olunan cərgəni aşağıdakı kimi elan edə bilərik:

```
int x[5];
```

5.1.2 Cərgənin elementlərinə müraciət

Cərgənin elementinə müraciət etmək üçün cərgənin adından və müraciət eləmək istədiyimiz elementin index nömrəsindən istifadə olunur, aşağıdakı kimi:

```
    cərgənin_adı [index];
```

Çalışma 2. Çalışma 1 -də elan olunan x cərgəsinin ilk elementini çap edin.

Həlli. İlk elementin indeksi 0 olduğundan ona müraciət etmək üçün x[0] yazmalıyıq və cout operatoru ilə onu aşağıdakı kimi çap edə bilərik:

```
std::cout<<x[0];
```

Çalışma 3. Çalışma 1 -də elan onunan cərgənin sonuncu elementinə 96 qiyməti mənimsədin.

Həlli. İndeks nömrələri 0-dan başladığından cərgədəki sonuncu elementin nömrəsini müəyyənləşdirmək üçün elementlərin sayından 1 çıxırıq. Elementlərin sayı 5 olduğundan cərgənin sonuncu elementin indeksi 4 olur. Bu elementə müraciət etmək üçün x[4] yazmalıyıq. Mənimsətmə operatoru ilə cərgənin sonuncu elementinə aşağıdakı kimi 96 qiyməti mənimsədə bilərik:

```
x[4] = 96;
```

Çalışma 4. Çalışma 1 -də elan olunan cərgənin 3 -cü elementinə istifadəçinin daxil etdiyi ədədi mənimsədən proqram tərtib edin.

Həlli. Cərgənin elementləri 0-dan başlayaraq nömrələndiyindən 3 -cü elementin indeks nömrəsi 2 olacaq, x[2]; . Bu elementə istifadəçinin daxil etdiyi ədədi mənimsətmək üçün cin operatorundan istifadə edə bilərik, aşağıdakı kimi:

```
   std::cin>>x[2];
```

Yekun proqram kodu bu şəkildə olacaq:

```
#include <iostream>

 int main(){

    // cegeni elan edek
    int x[5];

   // cergenin 3-cu elementine istifadecinin
   // daxil etdiyi qiymeti menimsedek
   std::cin>>x[2];

 }
```

Çalışma 5. Çalışma 1-də elan olunan x cərgəsinin bütün elementlərinə istifadəçinin daxil etdiyi qiymətlər mənimsədən proqram tərtib edin.

Həlli . Bu proqramı tərtib etmək üçün yenə cin operatorundan istifadə edəcəyik. for operatoru ilə dövr təşkil edəcəyik. Sayğac 0-dan 4-ə kimi dəyişəcək(4 -də daxil olmaqla). Dövr hər dəfə təkrar olunduqda istifadəçinin daxil etdiyi qiyməti cərgənin müvafiq elementinə mənimsədəcəyik. Bu zaman cərgənin müvafiq elementinin indeksi olaraq sayğacın qiymətindən istifadə edəcəyik. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main(){

  int i;

  //cerge elan edek
  int x[5];

  //istifadecinin daxil etdiyi ededleri
  // x -in elementlerine menimsedek
 for (i=0; i<=4; ++i)
  std::cin>>x[i];

}
```

Izahı: Dövrdə i sayğacı 0-dan 4-ə kimi dəyişir (4 -də daxil olmaqla). Sayğacın qiyməti hər dəfə bir vahid artır. Nəticədə dövr 5 dəfə təkrar olunur. Dövr hər dəfə təkrar olduqda cin operatoru istifadəçinin daxil etdiyi ədədi cərgənin indeksi sayğacın qiymətinə bərabər olan elementinə yerləşiririk, aşağıdakı kimi:

```
    for (i=0; i<=4; ++i)
       std::cin>>x[i];
```

Burada i -nin qiyməti 0-dan 4-ə kimi bir-bir artdıqca cin operatoru istifadəçidən növbəti ədədi daxil etməsini gözləyəcək və daxil olunan ədəd cərgənin indeksi i-yə bərabər olan elementinə mənimsədiləcə.

Çalışma 6. Çalışma 5-i elə dəyişin ki, proqram istifadəçinin daxil etdiyi ədədləri cərgəyə yerləşdirdikdən sonra onları cərgədən oxuyub təkrar ekranda çap etsin.

Həlli. Proqramın ilk hissəsi, yəni istifadəçinin daxil etdiyi ədədləri cərgəyə yerləşdirməli olan hissəsi Çalışma 5 -ə həll olundu. İndi isə yenə for operatorundan istifadə etməklə cərgəyə yerləşdirilmiş ədədləri ekranda çap edə bilərik, aşağıdakı kimi:

```
    for (i=0; i<=4; ++i)
```

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

Yekun proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main(){

  int i;

  //cerge elan edek
  int x[5];

  //istifadecini melumatlandiraq
  std::cout<<"Zehmet olmasa 5 eded daxil edin \n";

  //istifadecinin daxil etdiyi ededleri
  // x -in elementlerine menimsedek
 for (i=0; i<=4; ++i)
  std::cin>>x[i];


  //istifadecini melumatlandiraq
  std::cout<<"Siz ashagidaki ededleri daxil etdiniz \n";


   // cergenin elementlerini cap edek
   for (i=0; i<=4; ++i)
   std::cout<<x[i];

}
```

Çalışma 7. Çalışma 6 -da tərtib olunan proqramı elə dəyişin ki, istifadəçinin daxil etdiyi ədədləri əks sıralama ilə (sonuncudan əvvəlkinə doğru) çap etsin.

Həlli. Cərgənin elementlərini əks sıra ilə(axırdan əvvələ) çap etmək üçün dövrün sayğacının başlanğıc, son qiymətlərini və dəyişmə qaydasını tələb olunan qaydada yeniləməliyik. Əgər sayğacın qiymətin 4 -dən 0 -ra doğru hər -dəfə bir vahid azaltsaq onda cərgənin elementlərinin indekslərini əks sıra ilə almış oluruq. Daha sonra cout operatoru ilə onları çap edə bilərik.
Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main(){

  int i;

  //cerge elan edek
  int x[5];

  //istifadecini melumatlandiraq
  std::cout<<"Zehmet olmasa 5 eded daxil edin \n";

  //istifadecinin daxil etdiyi ededleri
  // x -in elementlerine menimsedek
 for (i=0; i<=4; ++i)
  std::cin>>x[i];


  //istifadecini melumatlandiraq
```

```
    std::cout<<"Siz ashagidaki ededleri daxil etdiniz \n";


     // cergenin elementlerini eks sira ile cap edek
     for (i=4; i=>0; --i)
     std::cout<<x[i];

  }
```

Çalışmalar.

1. İstifadəçinin daxil etdiyi ədədlərin cəmini hesablayan proqram tərtib edin.
2. İstifadəçinin daxil etdiyi ədədlər içərisində ən böyüyünü təyin edən proqram tərtib edin.
3. İstifadəçinin daxil etdiyi ədədlər içərisində 3 -ə bölünənləri təyin edən proqram tərtib edin.
4. İstifadəçinin daxil etdiyi ədədləri artan sıra ilə düzən proqram tərtib edin.
5. İstifadəçinin daxil etdiyi ədədlərdən cüt və tək ədədləri ayrı-ayrı (2 müxtəlif sırada) çap edən proqram tərtib edin.

$6 Sətirlər

Sətirlər simvol tipli cərgələrdir. Sətir elan etmək üçün aşağıdakı sintaksisdən istifadə olunur:

```
char Sətrin_Adı [ Simvolların_Sayı ];
```

Çalışma 1. 10 simvoldan ibarət s sətri elan edin.

Həlli . Sətrin elan olunma sintaksisinə əsasən tələb olunan sətri aşağıdakı kimi elan edə bilərik:

```
char s[10];
```

Sətrilərlə işləmək üçün funksiyalar

Sətirlərlə işləmək üçün bir neçə standart funksiyalar təyin olunmuşdur. Bu funksiyalardan istifadə edərkən proqrama string.h faylını əlavə etməliyik. Gəlin bu funksiyalarla tanış olaq.

strcpy(s1, s2) funksiyası

strcpy funksiyası parametr olaraq iki sətir qəbul edir və ikinci sətri birinciyə köçürür.

strcmp(s1,s2) fuksiaysı

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

strcat(s1,s2) fuksiaysı

strcat funksiyası parametr olaraq iki sətir qəbul edir və birinci sətrin sonuna ikinci sətri əlavə edir

strlen(s) funksiyası.

strlen funksiyası parametr olaraq hər-hansı sətir qəbul edir və nəticə olaraq həmin sətrin uzunluğunu qaytarır. Burada sətrin uzunluğu anlayışını elanda istifadə etdiyimiz Simvolların_Sayı ilə qarışdırmaq olmaz. Elandakı Simvolların_Sayı sətirə ən çoxu neçə simvol yerləşdirə biləcəyimizi bildirir, sətrin uzunluğu isə hal-hazırda sətrə yerləşdirilmiş simvolların sayını bildirir.

Çalışma 2. İstifadəçinin daxil etdiyi sətrin uzunluğunu ekranda çap edən proqram tərtib edin.

Həlli. Hər-hansı sətir elan edək. cin operatoru ilə həmin sətrə istifadəçinin daxil etdiyi qiyməti mənimsədək. strlen ilə sətrin uzunluğunu çap edək. Proqram aşağıdakı kimi olacaq:

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

int main(){

 int k;

 // ozunde en coxu 256 simvol saxlaya bilen
 // setir elan edirik
 char s[256];

 // istifadeciden her-hansi setir daxil etmesini isteyek
 std::cout<<"Zehmet olmasa her-hansi setir daxil edin \n";

 // istifadecinin daxil etdiyi qiymeti s setrine yerleshdirek
 std::cin>>s;

// s setrinde olan simvollarin sayini k-ya menimsedek
k = strlen(s);

// setrin uzunlugunu cap edek
```

```
  std::cout<<"setrin uzunlugu = "<<k<<"\n";

 }
```

Calışma 3. İstifadəçinin daxil etdiyi sətirdə olan 'a' simvollarının sayını tapan proqram tərtib edin.

Həlli . Proqramda hər-hansı sətir elan edib istifadəçinin daxil etdiyi ifadəni həmin sətirə yerləşdirəcəyik. Sətrin uzunluğunu strlen funksiyası ilə hesablayacayıq. Daha sonra dövr operatoru ilə sətrin bütün simvollarını yoxlayıb, a-ya bərabər olanların sayını tapa bilərik. Proqram kodu aşağıdakı kimi olar:

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

  int main(){

   int i,k,say;

   // ozunde en coxu 256 simvol saxlaya bilen
   // setir elan edirik
   char s[256];

   // istifadeciden her-hansi setir daxil etmesini isteyek
   std::cout<<"Zehmet olmasa her-hansi setir daxil edin \n";

   // istifadecinin daxil etdiyi qiymeti s setrine yerleshdirek
   std::cin>>s;

  // s setrinde olan simvollarin sayini k-ya menimsedek
  k = strlen(s);

  // evvelce say deyishenine 0 qiymeti menimsedek
  say = 0;

  //dovr operatoru ile setirde olan simvollari bir-bir yoxlayaq
  // eger simvol 'a' -dirsa onda say -i bir vahid artiraq
  for (i=0; i<k; ++i)
   {
     if (s[i] == 'a') say++;
   }

  // setirde olan 'a' simvollarinin sayini cap edek
  std::cout<<"setirde olan a simvollarinin sayi = "<<say<<"\n";

 }
```

Çalışma 4. İstifadəçinin daxil etdiyi iki sətrin eyni olub olmadığını müəyyən edən proqram tərtib edin.

Həlli. İki sətrin elan edirik və istifadəçinin daxil etdiyi ifadələri həmin sətirlərə yerləşdiririk. Daha sonra strcmp funksiyası ilə bu sətirləri müqaisə edirik. Proqram kodu aşağıdakı kimi olar:

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

   int main(){
```

```
    // iki setir elan edek
    char s1[256], s2[256];

    // istifadeciye birinci setri daxil etmesini bildirek
    std::cout<<"Zehmet olmasa birinci setri daxil edin \n";

    // Istifadecinin daxil etdiyi birinci setri s1 -e yazaq
    std::cin>>s1;

    // eyni qayda ile ikinci setri s2 -ye yazaq
    std::cout<<"Zehmet olmasa ikinci setri daxil edin \n";
    std::cin>>s2;

    // strcmp ile s1 ve s2 -ni muqaise edek
    if (strcmp(s1,s2) == 0)
          std::cout<<"Setirler eynidir \n";
    else
          std::cout<<"Setirler ferqlidir \n";

}
```

Çalışma 5. İstifadəçinin daxil etdiyi iki sətri birləşdirib çap edən proqram tərtib edin.

Həlli. İki sətir elan edək, istifadəçinin daxil etdiyi ifadələri həmin sətrilərə yerləşdirək. daha sonra strcat ilə ikinci sətri birinicinin sonuna əlavə edək. Proqram kodu aşağıdakı kimi olar:

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

int main(){

 // iki setir elan edek
 char s1[100], s2[100];

  std::cout<<"Zehmet olmasa birinci setri daxil edin \n";
 std::cin>>s1;

 std::cout<<"Zehmet olmasa ikinci setri daxil edin \n";
 std::cin>>s2;

 // strcat ile s1 -in sonuna s2 -ni elave edek
 strcat(s1,s1);

 // s1 - cap edek
 std::cout<<s1<<"\n";

}
```

Çalışmalar.

1. İstifadəçini daxil etdiyi sətrin uzunluğunu çap edən proqram tərtib edin.

2. İstifadəçinin daxil etdiyi sətrin son 5 simvolunu ekranda çap edən proqram tərtib edin.

3. İstifadəçinin daxil etdiyi sətrin ilk 3 simvolu ilə son 5 simvolunu birləşdirib çap edən proqram tərtib edin.

4. Elə proqram qurun ki, istifadəçinin daxil etdiyi sətrin 5-ci simvolu ilə 15-ci simvolu arasında qalan hissəsini çap etsin.

5. Elə proqram tərtib edin ki, istifadəçidən 3 sətir qəbul etsin və bu sətirləri ardıcıl birləşdirərək tam sətir kimi çap etsin.
6.* Elə proqram tərtib edin ki, istifadəçidən 4 sətir qəbul etsin və bu sətirləri daxil olma sırasının əksi ardıcıllığında birləşdirərək tam sətir kimi çap etsin.

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

$7 Struct tiplər.

7.1 Struct tipinin yaradılması

Struct tiplərindən müxtəlif tiplərdən olan dəyişənlərdən ibarət yeni tip yaratmaq üçün istifadə olunur. Struct tipi elanı sintaksisi aşağıdakı kimidir:

```
   struct Ad {

      tip_1 dəyişən_1;
      tip_2 dəyişən_2;
      .
      .
      .
      tip_n dəyişən_n
  };
```

Burada Ad yeni yaratdığımız struct tipinin adını bildirir. dəyişən_1, ... dəyişən_n isə yeni tipin həddləri adlanır.

Çalışma 1. int tipli x və char tipli c həddlərindən ibarət s adlı struct tipi elan edin.

Həlli. Tələb olunan struct tipinin adı s -dir və 2 həddi var: int tipli x və char tipli c. Struct tipinin elanı sintaksisinə asasən onu aşağıdakı kimi elan edə bilərik:

```
   struct s {
```

```
    int  x;
    char c;

};
```

7.2 Struct tipindən olan dəyişənlər

Struct tipi elan edərkən biz yeni dəyişən tipi yaratmış oluruq. Yeni yaratdığımız tipdən digər standart tiplərdən olduğu kimi dəyişən elan edə bilərik.

Çalışma 2. Çalışma 1 -də tərtib olunan s struct tipindən q adlı dəyişən elan edin.

Həlli. Tipin adı s, dəyişənin adı q olduğuna görə adi standart tiplərin elan olunması qaydasına əsasən q dəyişənin aşağıdakı kimi elan edə bilərik:

```
   // s adli yeni tip yaradiriq
   struct s {

    int  x;
    char c;

};


  // s tipindən q dəyisheni elan edirik
   s q;
```

7.3 Struct tipinin həddlərinə müraciət

Struct tipindən dəyişən elan etdikdən sonra biz artıq onun həddlərinə müraciət edə bilərik. Bunun üçün Dəyişənin_Adı.Həddin_adı sintaksisindən istifadə edirik (dəyişənin_adı nöqtə həddin_adı).

Çalışma 3. Çalışma 2 -də elan olunan q dəyişənin x həddinə 10, c həddinə isə 'A' qiyməti mənimsədin.

Həlli. Dəyişənin adı q, olduğundan onun x həddinə 10, c həddinə 'A' qiyməti mənimsətmək üçün

```
  q.x = 10;
  q.c = 'A';
```

yazmalıyıq. Struct tipinin elanı və q dəyişəninin elanı sətrilərini də nəzərə alsaq yekun kod aşağıdakı kimi olar:

```
   // s adli yeni tip yaradiriq
   struct s {

    int  x;
    char c;

};


  // s tipindən q dəyisheni elan edirik
  s q;
```

```
// q -nün x həddinə 10 qiyməti mənimsədək
q.x = 10;

// q -nün c həddinə 'A' qiyməti mənimsədək
q.c = 'A';
```

$8 Siniflər.

8.1 Sinfin elanı

Siniflər struct tiplərdən fərqli olaraq özlərində dəyişənlərlə yanaşı funksiya həddləri də saxlaya bilər. Sinfin elan olunma qaydası aşağıdakı kimidir:

```
class sinfin_adı {

  tip1 dəyişən1;
  tip2 dəyişən2
  .
  .
  .
  tip_n dəyişən_n;

  tip_1 funksiya_1 ();
  tip_2 funksiya_2 ();
  .
  .
  .
  tip_k funksiya_k ();

};
```

Dəyişənlər sinfin dəyişən həddləri, funksiyalar isə funksiya həddləri adlanır.

Çalışma 1. int tipli en və int tipli uz dəyişən həddləri, int sahe(int,int) funksiya həddindən ibarət duzbucaqli adlı sinif elan edin. sahe funksiyası nəticə olaraq qəbul etdiyi parametrlərin hasilini qaytarır.

Həlli . Sinfin elanı sintaksisini nəzərə alsaq tələb olunan sinfi aşağıdakı kimi elan edə bilərik:

```
class duzbucaqli {

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

};
```

8.2 Sinif tipindən dəyişən elan etmək

Sinif tipini yaratdıqdan sonra adi standart tiplərdə olduğu kimi dəyişən elan edə bilərik. Sinif tipindən elan olunan dəyişənlər bəzən obyekt də adlandırılır.

Çalışma 2. Çalışma 1 -də tərtib olunmuş duzbucaqlı sinfindən duzb adlı dəyişən elan edin.

Həlli. Sinif tipindən dəyişən elan etmə sintaksisinə əsasən tələb olunan dəyişəni aşağıdakı kimi elan edə bilərik.

duzbucaqli duzb;

8.3 Sinfin həddlərinə müraciət

Sinfin həddlərinə müraciət edən zaman struct tipinin həddlərinə müraciət qaydasından istifadə edəcəyik. Başqa sözlə dəyişənin_Adı.həddin_adı kimi.

Çalışma 3. Çalışma 2 -də elan olunan duzb dəyişəninin en həddinə 10, uz həddinə 20 qiymətləri mənimsədin.

Həlli. Sinif tipinin həddlərinə müraciət qaydasından istifadə edib sinfin dəyişən həddlərinə tələb olunan qiymətləri aşağıdakı kimi mənimsədə bilərik:

```
duzb.en = 10;
duzb.uz = 20;
```

8.4 Sinfin funksiya həddinin tərtibi

Sinfin funksiya həddinin proqram kodunu tərtib etmək üçün aşağıdakı sintaksisdən istifadə olunur:

```
nəticə_tipi  sinfin_adı::funksiyanın_adı (parametrlər) {

     proqram kodu;
}
```

Çalışma 4. Çalışma 1 -də elan olunmuş duzbucaqli sinfinin sahe funksiyasının proqram

kodunu tərtib edin.

Həlli. sinfin_adı duzbucaqli, funskiya həddinin adı sahe, funksiyanın qaytardığı nəticənin tipi int, qəbul etdiyi hər iki parametrin tipi int-dir. sahe funksiyası nəticə olaraq qəbul etdiyi parametrlərin hasilini qaytarır. Bunları və sinfin funksiya həddinin tərtib olunma qaydasını nəzərə alıb sahə funksiyasının proqram kodunu aşağıdakı kimi tərtib edə bilərik:

```
int  duzbucaqli::sahe (int x, int y){

   return x*y;
}
```

8.5 Açıq və gizli həddlər

Struct tipindən fərqli olaraq siniflər öz həddlərinə müraciəti açıq və gizli xassələri ilə tənzimləyə bilir. Sinfin açıq həddlərinə müraciət sərbəstdir. Gizli həddlərə isə yalnız sinfin öz həddləri müraciət edə bilər.

8.5.1 Açıq həddlər

Sinfin hər-hansı dəyişən və ya funksiya həddini açıq elan etmək üçün "public" xassəsi ilə elan etmək lazımdır. Sintaksis aşağıdakı kimidir:

```
class sinfin_adı {

   public:

     tip_1 dey_1;
     tip_2 dey_2;
     .
     .
     .
     tip_n dey_n;
};
```

8.5.2 Gizli həddlər

Sinfin hər-hansı dəyişən və ya funksiya həddini gizli elan etmək üçün "private" xassəsi ilə elan etmək lazımdır. Sintaksis aşağıdakı kimidir:

```
class sinfin_adı {

   private:

     tip_1 dey_1;
     tip_2 dey_2;
     .
     .
     .
     tip_n dey_n;
};
```

8.6 Yaradıcı

8.6.1 Sinif Yaradıcısı

Yaradıcı (eng. consturctor) sinfin dəyişən həddlərinə başlanğıc qiymətlər mənimsətmək üçün istifadə olunur. Sinin tipindən dəyişən elan edərkən yaradıcı funksiya avtomatik çağırılır və ilkin qiymətləndirməni yerinə yetirir. Yaradıcı elan etmək üçün sinfin daxilində sinfin tipinin adı ilə eyni adlı funksiya elan etmək lazımdır.

Çalışma 5. Çalışma 1-də elan olunmuş duzbucaqli sinfinin yaradıcısını tərtib edin.

Həlli. Sinfin tipinin adı duzbucaqli olduğundan yaradıcı elan etmək üçün sinfin daxilində duzbucali funksiya həddi elan etməliyik, aşağıdakı kimi:

```
class duzbucaqli {

  public:

    duzbucaqli();
    int en;
    int uz;
    int sahe (int, int);

};
```

8.6.2 Yaradıcının proqram kodu

Yaradıcının proqram kodunu sinfin digər funksiya həddləri kimi tərtib edə bilərik. Yeganə fərq odur ki, yaradıcını tərtib edərkən nəticə tipi göstərilmir.

Çalışma 6. Çalışma 1-də elan olunmuş duzbucaqli sinfinin yaradıcısının proqram kodunu tərtib edin. Yaradıcı sinfin en və uz həddlərinə uyğun olaraq 20 və 30 qiymətləri mənimsətməlidir.

Həlli. Sinfin funksiya həddlərinin proqram kodunun tərtibi sintaksisinə nəzər salaq:

```
nəticə_tipi  sinfin_adı::funksiyanın_adı (parametrlər) {

     proqram kodu;
}
```

sinfin_adı duzbucaqli, funksiyanın adı həmçinin duzbucaqli (bir daha yada salaq ki, yaradıcının adı sinfin tipinin adı ilə adlandırılır), nəticənin_tipi göstərilmədiyindən duzbucali sinfinin yaradıcısının proqram kodunu aşağıdakı kimi tərtib edə bilərik:

```
duzbucaqli::duzbucali(){

   en = 20;
   uz = 30;

}
```

Çalışma 7. Çalışma 1 -də elan olunan duzbcaqli sinfindən istifadə edərək proqram kodu tərtib edin.

Həlli. Nümunə proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

 class duzbucaqli {
   public:
     duzbucaqli();
     int sahe (int, int);
     int en;
     int uz;
  };

duzbucaqli::duzbucali(){
   en = 20;
   uz = 30;
}

int  duzbucaqli::sahe (int x, int y){
  return x*y;
}

 int main(){

    duzbucaqli duzb;

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

 }
```

Izahı. Proqramda əvvəl duzbucaqli sinfi elan olunur, daha sonra onun yaradıcısı və sahe funksiyalarının proqram kodları tərtib olunur. Proqramın əsas funksiyasında duzbucali sinfindən duzb adlı dəyişən elan olunur. Bu zaman yaradıcı funksiya avtomatik çağırılır və duzb obyektinin en və uz həddlərinə müvafiq olaraq 20 və 30 qiymətləri mənimsədir.

8.7 Nəsilvermə

Nəsilvermə hər-hansı mövcud sinif tipindən istifadə edərək bir qədər fərqli yeni sinif tipinin yaradılmasına deyilir. Bu zaman yeni yaradılan tip varis, başlanğıc tip isə əcdad sinif adlanır. Yeni sinif tipi yaradarkən istifadə olunan başlanğıc tipə əlavə dəyişən və ya funksiya həddləri artırıla bilər, həmçinin mövcud funksiya həddləri dəyişdirilə bilər.

8.7.1 Nəsilvermə ilə yeni sinfin yaradılması

Hər-hansı mövcud A sinfindən nəsilvermə ilə yeni B sinfini yaratmaq üçün aşağıdakı sintaksisdən istifadə olunur:

```
class B : puclic class A {

};
```

Bu zaman B sinfi öz əcdadı sayılan A sinfinin bütün funksiya və dəyişən həddlərinə sahib olur.

Çalışma 8. Nəsilvermə yolu ilə **Çalışma 1** -də elan olunmuş duzbucaqli sinfindən yeni paralelpiped sinfi yaradın.

Həlli. Nəsilvermə sintaksisinə əsasən tələb olunan sinfi aşağıdakı kimi yarada bilərik:

```
class paralelpiped : public class duzbucaqli {

};
```

8.7.2 Varis tipə yeni həddlərin artırılması

Nəsilvermə yolu ilə yaradılan tip öz əcdadının bütün həddlərinə sahib olur. Əgər hər-hansı yeni hədd artırma tələb olunsa onu sinfi elan edərkən { və } mötərizələri arasında qeyd etmək lazımdır, aşağıdakı kimi:

```
    class B : puclic class A {

      yeni həddlər

   };
```

Çalışma 9. Çalışma 8 -də elan olunmuş paralelpiped sinfinə yeni int tipli hündürlük dəyişən həddi və int (void) tipli hecm funksiya həddi artırın.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
class paralelpiped : public class duzbucaqli {

          public:

             int hundurluk
             int hecm(void);
};

int paralelpiped::hecm(){

   return en*uz*hundurluk;

}
```

8.7.3 Əcdad sinfin funksiyalarının dəyişdirilməsi

Əcdad sinfin hər-hansı funksiya həddini dəyişdirmək üçün həmin funksiyanı yenidən proqram kodunu yenidən tərtib etməliyik və bu zaman sinif_adı yerinə varis sinfin adını yazmalıyıq .

Çalışma 10. Çalışma 8 -də elan olunmuş paralelpiped sinfinin nəsilvermə ilə əldə etdiyi sahə funksiyasının proqra kodunu elə dəyişin ki, nəticə olaraq paralelpipedin sahəsini qaytarsın. Paralelpipedin sahəsi 2*(en*uzunluq+en*hündürlük+uzunluq*hündürlük) düsturu ilə hesablanır.

Həlli. Proqram kodu aşağıdakı kimi olar.

```
int paralelpiped::sahe (){

   return 2*(en*uz + en*hudurluk + uz*hundurluk);

}
```

Çalışma 11. Çalışma 8-də elan olunmuş paralelpiped sinfindən parp adlı obyekt elan edin. parp -ın həcmini və sahəsini çap edən proqram tərtib edin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

//duzbucaqli sinfi elan edirik
 class duzbucaqli {

public:

  duzbucaqli();
  int sahe (int, int);
  int en;
  int uz;

 };


duzbucaqli::duzbucali(){

   en = 20;
   uz = 30;

}


int  duzbucaqli::sahe (int x, int y){

   return x*y;
}


//nesilverme ile duzbucaqli sinfinden
//paralelpiped sinfi yaradiriq

  class paralelpiped : public class duzbucaqli {

          public:

             paralelpiped();
             int hundurluk
             int hecm(void);
```

```
};

// paralelpiped sinfinin yaradicisini yenileyirik
// ki, hundurluk heddine 25 qiymeti menimsetsin

paralelpiped::paralelpiped(){

   hundurluk = 25;

}

int paralelpiped::hecm(){

   return en*uz*hundurluk;

}


int paralelpiped::sahe (){

   return 2*(en*uz + en*hudurluk + uz*hundurluk);

}


int main(){

   // paralelpiped sinfinden parp adli obyekt
   // yaradaq

   paralelpiped parp;

   std::cout<<"parp obyektinin sahesi = "<<parp.sahe()<<"\n"
            <<"parp obyektinin hecmi  = "<<parp.hecm()<<"\n";

}
```

$9 Göstəricilər.

Bu paraqrafda biz proqramlaşdırmanın ən vacib və çətin hissələrindən biri sayılan göstəricilərlə tanış olacayıq. Göstəricilərlə iş təcrübə tələb edir və bu paraqrafda biz göstəricilərlə bağlı ən əsas məsələləri izah etməyə çalışmışıq.

Bu paraqrafda göstəricilərlə bağlı aşağıdakı məsələlərə toxunulur : dəyişənin ünvanı, göstəricilərin elanı, unvana görə ötürülmə, göstərici ilə cərgələrin əlaqəsi, göstəricilər üzərində hesab əməlləri, göstərici tipli obyektlər, dinamik yaradılma və silinmə.

9.1 Ünvan

9.1.1 Ünvan anlayışı

Sistem proqramlaşdırmada bəlkə də ən mühüm məsələ ünvan məsələsidir. Hər-hansı məlumata müraciət etmək üçün prosessor onun yaddaşdakı ünvanını mütləq bilməlidir. Bizim proqramda dəyişənlərə verdiyimiz adlar kompilyasiya zamanı müvafiq yaddaş ünvanları ilə əvəzlənir.

Ünvan məlumatın və ya proqram kodunun yaddaşdakı yerini göstərir.

Bütün bunlar sistem proqramlaşdırmanın mövzusu olsa da, istifadəçi proqramlaşdırma da unvanlardan geniş istifadə olunur.

9.1.2 Dəyişənin ünvanı

Dəyişənin ünvanının örgənmək üçün ünvan operatorundan istifadə olunur. Ünvan operatoru '&' kimi işarə olunur. Hər-hansı dəyişənin unvanın əldə etmək üçün onun adının əvvəlinə ünvan operatoru artırılır, aşağıdakı kimi:

```
  &dəyişən;
```

Çalışma 1. int tipli x adlı dəyişən elan edin. Onun ünvanın ekranda çap edin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
 #include <iostream>

 int main(){

    int x;

    std::cout<< "x deyisheninin unvani = "
```

```
            << &x
            << "\n";

}
```

9.2 Göstərici

Sətirlərlə, cərgələrlə, funksiyaya ötürülən parametrlə, məlumat struktları ilə (növbə, stek, siyahı, ağac), dinamik obyektlərlə işləyərkən ünvan əməliyyatları zamanı göstəricilərdən istifadə olunur.

Göstərici özündə qiymət olaraq **ÜNVAN** yadda saxlayan dəyişəndir.

Göstəriciyə istənilən dəyişənin ünvanın mənimsədə bilərik, nəticədə göstərici həmin dəyişənin yaddaşdakı yerini bildirəcək, istinad edəcək , başqa sözlə həmin dəyişəni GÖSTƏRƏCƏK. Əgər göstəricinin qimətini dəyişib ona başqa ÜNVAN mənimsətsək, onda göstərici müvafiq ünvandakı məlumata istinad edəcək (göstərəcək).

9.2.1 Göstəricinin elanı

Göstərici elan etmək üçün aşağıdakı sintaksisdən istifadə olunur:

```
tip  *göstərici_dəyişənin_adı;
```

Sintaksisdən göründüyü kimi göstərici elan edərkən adi dəyişənlərin elanı qaydasından istifadə olunur (tip ad;). Fərq yalnız göstəricinin adından əvvəl ulduz - '*' işarəsinin olmasıdır. Eyni sətirdə həm adi, həm də göstərici dəyişənlər elan edə bilərik.

Çalışma 2. int tipindən olan y adlı göstərici elan edin.

Həlli. Göstəricilərin elanı qaydasından istifadə edərək int tipindən olan y adlı göstəricini aşağıdakı kimi elan edə bilərik:

```
    int *y;
```

9.2.2 Göstəriciyə qiymət mənimsədilməsi

Assembler proqramlaşdırma kursunda fiziki yaddaşın strukturu, prosessorun yaddaşa müraciət imkanları mövzuları zamanı fiziki ünvanın 0-dan böyük - bərabər ədəd olduğu qeyd olunur. Yüksək səviyyəli dillərdə unvanların ədədi qiymətlərindən elə də geniş istifadə olunmur, sadəcə hesab əməlləri zamanı istifadə olunur. Bununla biz irəlidəki mövzularda məşğul olacayıq. Hələlik isə göstəricilərə qiymət mənimsədərkən unvan operatorundan istifadə etməklə hansısa dəyişənin ünvanın və ya başqa bir göstəricinin qiymətin mənimsədə bilərik.

Çalışa 3. int tipli adi x və göstərici y dəyişəni elan edin. Ünvan operatorundan istifadə etməklə x dəyişəninin ünvanını y -ə mənimsədin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
int x, *y;

y = &x;
```

Çalışma 4. int tipli adi x dəyişəni, y və z göstəriciləri elan edin. Ünvan operatorundan istifadə etməklə x dəyişəninin ünvanını y -ə mənimsədin. y-in qiymətini (x -in ünvanını) z

-tə mənimsədin.

Həlli.

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

   y = &x;

   z = y;
```

9.2.3 Göstəricinin istinad etdiyi dəyişənə müraciət

Göstəriciyə hər-hansı ünvanı mənimsətdikdən sonra həmin unvanda yerləşən məlumatı asanlıqla oxuyub, dəyişdirmək olar. Bunun üçün göstəricinin adının əvvəlinə ulduz - '*' simvolunu artırmaq tələb olunur.

Çalışma 5. int tipli x dəyişəni və y göstəricisi elan edin. x -in ünvanın y-ə mənimsədin. Göstəricinin istinad etdiyi sahədəki məlumatı çap edin.

Həll. int ipli x dəyişəni və y göstəricisi elanı üçün int x, *y ; kodundan istifadə edək. x -ə 25 qiyməti mənimsədək, x = 25; . y = &x; kodu ilə x -in ünvanın y -ə mənimsədək. y-in istinad etdiyi məlumatı çap etmək üçün cout<< *y ; kodundan istifadə edə bilərik. Yekun proqram kodu aşağıdakı kimi olar:

```
 #include <iostream>

 int main(){

int x, *y;

 x = 25;


  y = &x;

  std::cout<< *y ;

}
```

Çalışma 6. int tipli x dəyişəni və y göstəricisi elan edin. x -in ünvanın y-ə mənimsədin. Göstəricinin istinad etdiyi sahədəki məlumatı dəyişdirin. Dəyişikliyi yoxlayın.

Həlli. Yenı Çalıma 5 -dəki kimi y -ə x -in ünvanın mənimsədək. Əvvəlcə x-ə 25 qiyməti mənimsədək. Daha sonra * simvolundan istifadə etməklə y-in istinad etdiyi sahəyə hər-hansı qimyət yazaq, misal üçün 40, aşağıdakı kimi *y = 40;. y göstəricisi x -ə istinad etdiyindən bu zaman x -in 40 olar. Bunu yoxlamaq üçün *y = 40; əməliyyatından əvvəl və sonra x -in qiymətini çap edək. Proqram kodu aşağıdakı kimi olar.

```
 #include <iostream>

 int main(){

int x, *y;

 x = 25;

 y = &x;

 std::cout<< " *y = 40; -dan  evvel x = " << x ;
```

```
 *y = 40;

 std::cout<< " *y = 40; -dan sonra x = " << x ;

}
```

9.3 Göstəricilər ilə Cərgələrin əlaqəsi

Cərgələr göstəricilərin xüsusi bir formasıdır. Cərgələr də göstəricilər kimi yaddaşda müəyyən bir sahəyə istinad edir. Fərq yalnız ondadır ki, göstəricini istənilən ünvana yönləndirmək olar, cərgələr isə proqramın icrası boyu yalnız bir unvana - elementlər yerləşdiyi sahənin başlanğıcına (cərgənin ilk elementinə) istniad edirlər. Başqa sözlə cərgənin adı cərgənin ilk elementinə istinad edən və qiyməti dəyişdirilə bilməyən (constant) göstəricidir.

9.3.1 Göstəricinin cərgəyə mənimsədilməsi

Cərgə elan edən zaman biz aşağıdakı sintaksisdən istifadə edirik:

```
   tip  cərgənin_adı [elementərin_sayı];
```

Qeyd etdik ki, burada cərgənin_adı cərgədəki ilk elementə istinad edən göstəricidir. Buna görə başqa göstərici elan edib bu cərgəyə mənimsədə bilərik. Misal üçün aşağıdakı kimi:

```
    tip  cərgənin_adı [elementərin_sayı], *göstərici;

     göstərici = cərgənin_adı;
```

Bu zaman göstərici cərgənin ilk elementinə istinad etmiş olacaq. Bu bizə * operatorundan istifadə etməklə həmin elementin qiymətini örgənməyə və dəyişməyə imkan verir. Misal üçün *göstərici = 45; əməliyyatı cərgənin ilk elementinə ()cərgənin_adı[0]) 45 qiymətni mənimsətmiş olacaq.

Çalışma 7. int tipli 5 elementdən ibarət x cərgəsi və y göstəricisi elan edin. y göstəricisindən istifadə etməklə x-in ilk elementinə 120 qiymətini mənimsədin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
 int main(){

  int x[5], *y;

  y = &x;

  *y = 120;

}
```

9.3.2 Göstərici üzərində hesab əməlləri

Göstərici qiymət olaraq özündə yaddaş ünvanı saxlayır. Qeyd etdik ki, fiziki yaddaş ünvanları 0-dan böyük - bərabər tam ədədlərlə ifadə olunur. Bu isə ünvan üzərində hesab əməlləri aparmağa imkan verir. İstifadəçi proqramlaşdırmada ünvanlarla bağlı əsasən toplama və çıxma əməliyyatları aparılır.

Göstəricinin qiymətini artırıb-azaltdıqda o yaddaş sahəsi boyu müvafiq olaraq "yuxarı" və "aşağı" sürüşür. Bundan istifadə edib göstəricinin qiymətini artırıb-azaltmaqla onu cərgə boyu yuxarı və aşağı sürüşdürərək cərgənin istənilən elementi üzərinə yerləşdirmək olar.

Tutaq ki, aşağıdakı kimi cərgə və göstərici elan etmişik:

```
tip  cərgənin_adı [elementərin_sayı], *göstərici;
```

göstəricini cərgəyə mənimsədək,

```
göstərici = cərgənin_adı;
```

Hal-hazırda göstərici cərgənin ilk elementi üzərindədir (cərgənin_adı[0]). Əgər göstəricinin qiymətini 1 vahid artırsaq o növbəti elementin üzərinə sürüşər,

```
göstərici = göstərici + 1;
```

Hal-hazırda göstərici cərgənin ikinci elementi üzərindədir (cərgənin_adı[1]).

Eyni qayda ilə əgər göstəricinin qiymətini 1 vahid azaltsaq onda o cərgənin əvvəlki elementi üzərinə sürüşər.

```
göstərici = göstərici - 1;
```

Bu zaman göstərici cərgənin ilk elementi (cərgənin_adı[0]) üzərinə sürüşər.

Çalışma 8. int tipli 10 elementdən ibarət x cərgəsi və y göstəricisi elan edin. Dövr operatorundan və y göstəricindən istifadə etməklə x -in elementlərinə 50-dən 59 -a kimi qiymətlər mənimsədin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main (){

 int x[10], *y;

 int i;

 // y-e x -i menimsedek
 y = x;

 // y hal-hazirda x -in ilk elementine (x[0])
 // muraciet edir

 for (i=0; i<10; ++i){

   // y -in istinad etdiyi elemente qiymet menimsedek
   *y = 50 + i;

    // y -i cergenin novbeti elementi uzerin surushdurek
    y = y + 1;
 }

}
```

9.3.3 Göstəricilərlə sətirlərin əlaqəsi

Sətirlər cərgələrin xüsusi - elementlərinin tipi char olan halıdır. Ona görə göstəricilər və cərgələrlə bağlı bütün qaydalar analoji olaraq sətirlərə tətbiq olunur.

Çalışma 9. Tutaq ki, 16 elementi olan s sətrinə "abcdefghijklmnop" qiyməti mənimsətmişik. Əgər char tipi g göstəricisi elan etsək və ona s qiyməti mənimsətsək o hansı simvola istinda edər?

Həlli. g göstəricisini s sətrinə mənimsətdikdə (g = s;) g göstəricisi s sətrinin (cərgəsinin) ilk elementinə istinad edər, başqa sözlə 'a' simvoluna.

Çalışma 10. Tutaq ki, 16 elementi olan s sətrinə "abcdefghijklmnop" qiyməti mənimsətmişik. Əgər char tipi g göstəricisi elan etsək və ona s qiyməti mənimsətdikdən sonra göstəricinin qiymətini 1 vahid artırsaq o hansı simvola istinda edər?

Həlli. g göstəricisini s sətrinə mənimsətdikdə (g = s;) g göstəricisi s sətrinin (cərgəsinin) ilk elementinə istinad edər, yəni 'a' simvoluna. Daha sonra g -nin qiymətini 1 vahid artırsaq (g = g + 1;) onda o növbəti elementin üzərinə sürüşər, başqa sözlə 'b' simvolunun.

Çalışma 11. Tutaq ki, 16 elementi olan s sətrinə "abcdefghijklmnop" qiyməti mənimsətmişik. Əgər char tipi g göstəricisi elan etsək və ona s qiyməti mənimsətdikdən sonra göstəricinin qiymətini 5 vahid artırsaq o hansı simvola istinda edər?

Həlli. g göstəricisini s sətrinə mənimsətdikdə (g = s;) g göstəricisi s sətrinin (cərgəsinin) ilk elementinə istinad edər, yəni 'a' simvoluna. Daha sonra g -nin qiymətini 5 vahid artırsaq (g = g + 5;) onda o 5 vahid sonrakı elementin , başqa sözlə 'f' simvolunun üzərinə sürüşər.

Çalışma 12. Tutaq ki, 20 elementli char tipli s sətri və 10 elementli q sətri elan edilib və s sətrinə "abcdefghijklmnop" qiyməti mənimsədilib. Göstəricilərdən və strncpy funksiyasından istifadə etməklə s sətrinin 3-cü simvolu ilə 10 -cu simvolu arasında qalan hissəsini q sətrinə köçürün .

Həlli. Əvvəlcə g göstəricisi elan edək və onu s sətrinin 3 -cü elementi üzərinə sürüşdürək. Bunun üçün əvvəlcə g -ni s -ə mənimsədək, daha sonra onun qiymətini 2 vahid artıraq, aşağıdakı kimi:

```
g = s;
g = g + 2;
```

Bu zaman g göstəricisi s sətrinin 3-cü elementi üzərinə sürüşmüş olar. Daha sonra növbəti 7 simvolu (10 -cu simvola qədər) q -yə köçürmək üçün strncpy funksiyasından aşağıdakı kimi istifadə edə bilərik:

```
strncpy(q,g,7);
```

Yekun proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

int main(){

  char s[20], q[10], *g;

  // s setrine  "abcdefghijklmnop" qiymeti yerleshdirek
  strcpy(s, "abcdefghijklmnop");
```

```
    // g -ni s -in 3-cu simvolu uzerine surushdurek
    g = s;
    g = g + 2;
    // birbasha g = s + 2; ve ya g = &s[2]; yaza da  bilerdik

    // s -in 3 -cu simvolu ile 10 -cu simvolu arasinda
    // qalan hissesini q -ye menimsedek
    strncpy(q,g,7);


}
```

9.4 Dinamik Yaradılma və Silinmə

Proqramın icrası zamanı yaddaş sahəsinin ayrılmasına və silinməsinə dinamik yaradılma və silinmə deyilir. Gösətricilərə proqramın icrası boyu yaddaşda yer ayrıla bilər və həmin yerə ehtiyac qalmadıqda həmin yer azad oluna bilər.

9.4.1 Dinamik yaradılma

Göstəriciyə yaddaşda yer ayırmaq üçün new operatorundan istifadə olunur. new operatorundan istifadə etməklə göstəriciyə yer ayırmaq üçün aşağıdakı sintaksisdən istifadə edilir:

```
    göstərici = new tip;
```

Çalışma 13. int tipli x göstəricisi elan edin və ona dinamik yer ayırın.

Həlli. Kod aşağıdakı kimi olar:

```
    int *x;

    x = new int;
```

9.4.2 Dinamik ayrılan yaddaşdan istifadə

! Tipik proqramlaşdırma səhvi: Göstəriciyə yer ayırmada ona muraciət etmək.

Göstəriciyə new operatoru ilə yer ayırdıqdan sonra ulduz * operatorundan istifadə etməklə həmin yerdə olan məlumata müraciət etmək olar. Ulduz operatoru ilə göstəricinin istinad etdiyi sahəyə müraciət qaydası ilə biz artıq tanışıq.

Çalışma 14. int tipli x göstəricisi elan edin, new operatoru ilə x -ə yer ayırın və həmin yerə 5 qiymətini yazın.

Həlli. Kod aşağıdakı kimi olacaq:

```
    int *x;

    x = new int;

    *x = 5;
```

9.4.3 Dinamik silinmə

new operatoru ilə ayrılan yeri silmək üçün delete operatorundan istifadə olunur. delete operatorunun sintaksisi aşağıdakı kimidir:

```
   delete göstərici;
```

Çalışma 15. int tipli x göstəricisi elan edin, new operatoru ilə x -ə yer ayırın və həmin yerə 5 qiymətini yazın. x -ə ayrılmış yeri yaddaşda silin.

Həlli. Kod aşağıdakı kimi olacaq:

```
int *x;

x = new int;

*x = 5;

delete x;
```

9.4.4 Dinamik obyektlər

Strukt və ya sinif tipindən olan dəyişənlə obyekt adlanır. Göstərici obyektlərin elanı və onlara dinamik yer ayrılma və silinmə adi dəyişənlərdə olduğu kimidir. Həddlərinə müraciət isə bir qədər fərqlidir. Adi halda obyektin həddinə müraciət etmək üçün nöqtə - '.' -dən istifadə edirdik. Dinamik obyektlərin həddlərinə müraciət etmək üçün isə '->' işarələməsindən istifadə olunur, aşağıdakı kimi:

```
obyekt->hədd
```

Çalışma 16. int tipli x və char tipli c həddlərindən ibarət s adlı struct tipi elan edin. s tipində g adlı göstərici obyekt elan edin. g göstəricisinə dinamik yer ayırın. g -nin x həddinə 5, c həddinə 'B' qiyməti mənimsədin.

Həlli. Kod aşağıdakı kimi olar:

```
// evvelce   int tipli x və char tipli c həddlərindən ibarət
// s adlı struct tipi elan edek

 struct s{
    int  x;
    char c;
 };

// s tipinden g gostericisi elan edek
s *g;

// g -ye yer ayıraq
g = new s;

// g -nin heddlerine qiymetler menimsedek
g->x = 5;
g->c = 'B';
```

Çalışma 17. int tipli en və uz dəyişən həddlərindən, int sahe() funksiya həddindən ibarət kvadrat sinfi elan edin. kavadrat sinfindən kv göstəricisi elan edin. kv -yə dinamik yer ayırın və sahəsini çap edin.

Həlli. Proqram kodu aşağıdakı kimi olar:

```
#include <iostream>

class kvadrat{

   public:
    kvadrat();
    int sahe();
   private:
```

```
    int en;
    int uz;
};

int main(){

   kvadrat *kv;

   kv = new kvadrat;

   std::cout<< kv->sahe();

   delete kv;
}
```

$10 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

```
// prg_11_1.cpp
/*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";
}
```

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 edək.
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.

Char	**Dec**						
\0	0	(sp)	32	@	64	`	96
(soh)	1	!	33	A	65	a	97
(stx)	2	"	34	B	66	b	98
(etx)	3	#	35	C	67	c	99
(eot)	4	$	36	D	68	d	100
(enq)	5	%	37	E	69	e	101
(ack)	6	&	38	F	70	f	102
(bel)	7	'	39	G	71	g	103
(bs)	8	(	40	H	72	h	104
\t	9	)	41	I	73	i	105
\n	10	*	42	J	74	j	106
(vt)	11	+	43	K	75	k	107
(np)	12	,	44	L	76	l	108
(cr)	13	-	45	M	77	m	109
(so)	14	.	46	N	78	n	110
(si)	15	/	47	O	79	o	111

(dle)	16	0	48	P	80	p	112
(dc1)	17	1	49	Q	81	q	113
(dc2)	18	2	50	R	82	r	114
(dc3)	19	3	51	S	83	s	115
(dc4)	20	4	52	T	84	t	116
(nak)	21	5	53	U	85	u	117
(syn)	22	6	54	V	86	v	118
(etb)	23	7	55	W	87	w	119
(can)	24	8	56	X	88	x	120
(em)	25	9	57	Y	89	y	121
(sub)	26	:	58	Z	90	z	122
(esc)	27	;	59	[	91	{	123
(fs)	28	<	60	\	92	|	124
(gs)	29	=	61	]	93	}	125
(rs)	30	>	62	^	94	~	126
(us)	31	?	63	_	95	(del	) 127