ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Dərs №8

C

Dilində proqramlaşdırma

Mündəricat

1. Funksiyalar aləminə giriş

İstifadənin zəruriliyi. Elan edilməsi. Çağırılması.

Bəzən elə şərait yaranır ki, proqramımızda kodun bəzi hissələri bir neçə dəfə təkrarlanır. Bu halda biz kodun eyni bir fraqmentini bir neçə dəfə yazmaq məcburiyyətində qalırıq. Belə vəziyyətin bir neçə mənfi tərəfi vardir. Birincisi, proqramın yaradılma prosesi daha yorucu və uzun olur. İkincisi, nəticədə alınan faylın həcmi artır. Bəs vəziyyətdən necə cıxmalı?! Elə bir mexanizm varmı ki, proqramçının işini avtomatlaşdırsın və proqramın kodunu yığcam etsin?! Bəli - var. Belə bir mexanizm - funksiyadır. Beləliklə:

Funksiya - elə bir xüsusi strukturdur ki, onun köməyi ilə kodun iki və ya daha artıq dəfə təkrarlanan hansısa hissəsini proqramın gövdəsi xaricinə çıxarır. Çıxarılmış bu hissəyə xüsusi ad verilir və ondan gələcəkdə istifadə etmək üçün bu adı kodda qeyd etmək lazımdır.

Elanetmə sintaksisi

Funksiyanın iki cür elan edilmə üsulu var:

■ main funksiyasından əvvəl verilir.

■ Funksiya prototipin köməyi ilə verilir, main funksiyasından sonra isə funksiyanın gövdəsi təsvir olunur.

Birinci üsul: Main funksiyasından əvvəl.
Funksiyanın verilməsinin ümumi sintaksisi:

```
qaytarılan_qiymət funksiyanın_adı (parametrlər)
{
    Təkrarlanan_kod_bloku (gövdə);
}
```

1. Funksiyanın adı da dəyişənin adında olduğu kimi eyni qanunlara tabedir və təbii ki, proqramçı tərəfindən təyin edilir.
2. Parametrlər - funksiyaya kod uzərində iş üçün zəruri olan giriş verilənləridir. Parametr olaraq tipi göstərilən adi dəyişənlərdən istifadə olunur. Əgər funksiya üçün giriş verilənlərinə ehtyac yoxdursa, mötərizələri boş saxlamaq lazımdır. Parametrlərin ikinci adı - arqumentlərdir.
3. Qaytarılan qiymət - Funksiyanın işinin nəticəsi.

Qaytarılmalı olan qiymətin əvəzinə istənilən baza tipi yazılır. Bu, funksiyanın proqramda çağırılma yerinə yazılan verilənlər tipidir. Əgər funksiya icra zamanı heç nə qaytarmırsa, onda qaytarılan qiymətin əvəzinə void (boş) yazılır. Ümumiyyətlə «mənası olan» qaytarılan qiymət yalnız o halda göstərilir ki, funksiyanın işinin nəticəsi sonrakı hesablamalar üçün lazımdır.

Qeyd: Bir funksiyanın içində digərini yaratmaq olmaz.

Qeyd: Funksiya elan edilməyibsə, onu çağırmaq olmaz.

Funksiyanın çağırılma sintaksisi.

Kodun müəyyən hissəsində funksiyadan istifadə etmək üçün həmin hissədə onu çağırmaq lazımdır. Funksiya çağırılarkən onun gövdəsində yerləşən kodun hissəsinin yerinə yetirilməsi haqqında əməliyyat sisteminə təlimat verilir. Funksiyanın yerinə yetirilməsi sona çatdıqda, proqram əsas kodda funksiyanın çağırıldığı yerdən öz işini davam etdirməlidir. Funksiyanın çağırılması - funksiyanın adından, arqumentlərin (əgər varsa) ötürülməsindən və qaytarılan qiymətdən (əgər ona ehtiyac varsa) ibarətdir:

```
Dəyişənin_adı=funksiyanın_adı(parametr1, parametr2,...., parametr N);
```

1. Verilmiş qiymətlərin tipi funksiya təyin olunarkən onun arqumentlərinin tipinə uyğun olmalıdır. Ötürülən qiymətin asanlıqla lazım olan tipə çevrilə bilməsi halları istisna olur.
2. Funksiya verilərkən neçə paramentrin ötürülməsi müəyyən olunursa, funksiya çağırılanda həmin sayda da parametrlər göstərilməlidir.
3. Qaytarılan qiymətin yazıldığı dəyişənin tipi, funksiyanın qaytardığı qiymətin tipinə uyğun olmaıdır. Bu vacib deyil amma arzuolunandır.

return - açar sözü

Funksiyanın qiymətinin yenidən proqrama, onun çağrıldığı yerə qaytarılması üçün return əmrindən istifadə edilir. Qaytarılma sintaksisi belədir:

```
return qiyməti;
```

Əgər funksiya heç bir qiymət qaytarmırsa, onda return əmrindən funksiyanı dayandırmaq üçün istifadə olunur. Bunun üçün yazırıq:

```
return; // Bu halda return, dövrdəki break əmrinə oxşar vəzifəni yerinə yetirir.
```

return əmrindən istifadənin bir neçə vacib məqamları:

1. Bir neçə qaytarma əmri ola bilər (şəraitdən asılı olaraq), amma onlardan yalnız biri işləyəcək.
2. Əgər return əmri işə düşdüsə (formasından asılı olmayaraq), bu halda funksiyada ondan aşağıda yerləşən heç nə işləməyəcək.
3. Əgər funksiyanın qaytardığı tip void deyilsə, onda həmişə belə bir formadan istifadə olunur: return qiymət;

Biz artıq nəzəri hissə ilə tanış olduq. İndi isə funksiyanın yaradılmasına aid nümunələr bölməsinə keçək.

2. Funksiyanın yaradılmasına aid nümunələr və onun çağırılması

Heç bir parametr qəbul etməyən və heç bir qiymət qaytarmayan funksiya.

```
#include <iostream>
using namespace std;

// funksiyanın yaradılması
void Hello(){
	// ekrana sətrin verilməsi
	cout<<"Hello, World!!!\n\n";
}
void main(){


	Hello(); // çağırılma
	Hello(); // çağırılma
	Hello(); // çağırılma


}
```

Nəticə olaraq Hello World!!! sətri üç dəfə ekrana verilir.

Bir parametr qəbul edən, lakin heç bir qiymət qaytarmayan funksiya.

```
#include <iostream>
using namespace std;

// ulduzlardan ibarət count uzunluqlu sətir verir.
void Star(int count){
	for(int i=0;i<count;i++)
		cout<<'*';
	cout<<"\n\n";
}


void main(){

	Star(3); // üç ulduzdan ibarət sətrin göstərilməsi
	Star(5); // beş ulduzdan ibarət sətrin göstərilməsi

}
```

Iki paramentr qəbul edən lakin heç bir qiymət qaytarmayan funksiya.

```
#include <iostream>
using namespace std;
// symb simvolundan ibarət, count uzunluqlu sətir verir
 void AnyLine(char symb, int count){
          for(int i=0;i<count;i++)
                    cout<<symb;
          cout<<"\n\n";
 }
 void main(){


              AnyLine('+',3); // üç toplama işarəsindən ibarət sətrin göstərilməsi
              AnyLine('=',5); // beş bərabərlik işarəsindən ibarət sətrin göstərilməsi


  }
```

İki paramentr qəbul edən və qiymət qaytaran funksiya.

```
#include <iostream>
using namespace std;
// (Digit) ədədin (Pow) dərəcəsini hesablayır
int MyPow(int Digit, int Pow){
          int key=1;
          for(int i=0;i<Pow;i++)
                    key*=Digit;
          return key;
}
void main(){
           // res dəyişəninə qaytarılan nəticənin yazılması
          int res=MyPow(5,3);
          cout<<"Res = "<<res<<"\n\n";



}
```

3. Arqumentlərin ötürülməsi. Funksiyaların prototipləri.

Arqumentlərin qiymətə görə ötürülməsi.

Gəlin əməli yaddaşda baş verənlər haqda danışaq.

Funksiya təyin olunarkən göstərilən arqumentlər - formal arqumentlər adlanır. Bu onunla əlaqədardır ki, onlar əməli yaddaşda funksiyanın çağrılması zamanı yaranırlar. Funksiyadan çıxış zamanı, bu cür parametrlər ləğv olunurlar. Buna görə də, əgər proqramın başqa bir funksiyasında eyni adlı parametrlər istifadə olunarsa, onda heç bir münaqişə olmaz.

Arqumentin ötürülmə üsullarından birinə baxaq:

Verilənlərin qiymətə görə ötürülməsi zamanı formal parametrlərin işinə aid nümunə.

```
#include <iostream>
using namespace std;

// dəyişənlərin qiymətlərinin yerini dəyişməlidir.
void Change(int One, int Two){
	cout<<One<<" "<<Two<<"\n\n";// 1 2
	int temp=One;
	One=Two;
	Two=temp;
	cout<<One<<" "<<Two<<"\n\n";// 2 1
}
void main(){


	int a=1,b=2;
	cout<<a<<" "<<b<<"\n\n"; // 1 2
	// qiymətə görə ötürülmə
	Change(a,b);
	cout<<a<<" "<<b<<"\n\n"; // 1 2

}
```

1. Funksiyaya a və b deyil onların dəqiq surəti ötürülür.
2. Bütün dəyişikliklər surətlər (one, two) üzərində olur, a və b dəyişilməz qalırlar.
3. Funksiyadan çıxış zamanı bu müvəqqəti surətlər ləğv olunurlar.

Yuxarıda deyilənləri nəzərə alaraq funksiyanın daxilində qiymətlər emal olunarkən diqqətli olun. Nəticə olaraq bu problemi həll etməyi öyrənəcəyik.

Qeyd: Yeri gəlmişkən, massivlərlə bu baş vermir.

Funksiyadan çıxışda massivdə baş verən bütün dəyişikliklər saxlanılır.

Massivlər haqqında bəzi məqamlar...

Massivlər arqumentlər kimi istifadə olunarkən bəzi xüsusiyyətlərə malik olurlar. Bu xüsusiyyət özünü onda göstərir ki, massivin adı birinci elementin göstəricisinə çevrilir, yəni massivin ötürülməsi zamanı göstəricinin ötürülməsi baş verir. Bu səbəbdən çağrılan funksiya ayırd edə bilmir ki, onun ötürdüyü göstərici massivin əvvəlinə aiddir yoxsa ki, yeganə olan obyektə. Bir ölçülü massivin ötürülməsi zamanı boş kvadrat mötərizə göstərmək kifayətdir:

```
int summa (int array[ ], int size){
        int res=0;
        for (int i = 0; i < size; i++)
            res += array[i];
        return res;
}
```

Əgər funksiyaya ikiölçülü massiv ötürülərsə, onda funksiyanın uyğun arqumentinin təsviri sütunların sayını göstərməlidir; sətirlərin sayı əhəmmiyətli deyil.

Belə ki, yuxarıda deyildiyi kimi funksiyaya faktiki olaraq gösterici ötürülür.

```
int summa (int array[ ][5], int size_row, int size_col){
        int res=0;
        for (int i = 0; i < size_row; i++)
              for (int j = 0; j < size_col; j++)
                        res += array[i][j];
        return res;
}
```

Funksiyanın prototipləri və ya funksiyanın verilməsinin ikinci üsulu.

Funksiyanın verilməsinin ikinci üsulu zamanı kompilyatora funksiyanın varlığı haqqında məlumat vermək lazımdır. Bunun üçün main strukturuna qədər funksiyanın adı, onun arqumentləri və qaytarılan qiymətin tipi göstərilir. Belə struktura funksiyanın prototipi deyilir. Kompilyator funksiyanın prototipinə rast gəldikdə, o dəqiq bilir ki, funksiya proqramda main strukturundan sonra gəlir və orada da yerləşməlidir.

```
Kitabxana
Qaytarılan_qiymət funksiyanın_adı(arqumentlər);
void main(){
          main gövdəsi;
}
Qaytarılan_qiymət funksiyanın_adı(arqumentlər){
          Funksiyanın gövdəsi;
}
```

```
#include <iostream>
using namespace std;
// prototiplər
void MyFunc();
void MyFuncNext();


void main(){
	MyFunc();//MyFunc
	MyFuncNext(); //MyFuncNext
}
//kodun təsviri
void MyFunc(){
	cout<<"MyFunc\n";
}
void MyFuncNext(){
	cout<<"MyFuncNext\n";
}
```

Funksiyanın belə verilməsi daha düzgün sayılır.

4. Görünmə oblastı. Qlobal və lokal dəyişənlər.

Görünmə oblastı.

Proqram kodunda istənilən fiqurlu mötərizə görünmə oblastını yaradır. Bu o deməkdir ki, bu mötərizələrin içində verilən dəyişənlər ancaq bu mötərizələr daxilində görünəcək. Başqa sözlə funksiyanın, dövrun, if əmrinin və s. funksiyanın daxilində yaranan dəyişənə proqramın başqa yerindən müraciət olunsa, xəta baş verər. Belə ki, fiqurlu mötərizələrin daxilindən çıxdıqdan sonra bu dəyişən məhv olunur.

```
int a=5;
if(a==5){
        int b=3;
}
cout<<b; // xəta! b movcud deyil
```

Qlobal və lokal dəyişənlər.

Görünmə oblastının qaydalarına əsasən dəyişənlər iki növ olur - lokal və qlobal.

Lokal dəyişənlər kodun hər hansı bir hissəsində yaranır. Biz artıq bilirik ki, proqram üçün bu nə deməkdir.

Qlobal dəyişənlər hər hansı görünmə oblastından kənarda yaranır: Əsasən main() funksiyasından əvvəl. Belə dəyişənlər proqramın istənilən hissəsində görünür. Susmaya görə qlobal dəyişənlər lokal dəyişənlərdən fərqli olaraq başlanğıc qiymət olaraq 0 qiymətini alırlar.

Və ən əsası, funksiya daxilində qlobal dəyişənlə baş verən dəyişikliklər sonuncudan çıxarkən yadda saxlanılır.

Qeyd: yadda saxlayın - məsələn əgər a adlı qlobal dəyişən varsa və həm də funksiya daxilində eyni adlı dəyişən elan edilirsə, onda funksiya daxilində məhz burada verilən dəyişən istifadə olunacaq. Ona görə də çalışın görünmə oblastında istifadə olunan dəyişən adlarından istifadə etməyəsiz. Proqramda eyni adlı dəyişənlərdən istifadə etməməklə bundan qaçmaq olar.

```
int a=23; // qlobal a
void main(){
        int a=7;// lokal a
        cout<<a; // 7, lokal dəyişən istifadə olunur
}
```

5. Susmaya görə arqumentlər (parametrlər)

Funksiyanın formal parametrinə susmaya görə arqument verilə bilər. Bu o deməkdir ki, bu arqumentə çağırış zamanı qiymət ötürülməyə bilər. Bu halda susmaya görə qiymət istifadə olunacaq.

Belə yanaşmanın reallaşdırılması üçün ümumi sintaksis belə olacaq:

```
Qaytarılan_qiymətin_tipi funksiyanın_adı (arqumentin_tipi arqumentin_adı=susmaya_görə_qiymət)
```

Burada həm susmaya_görə_qiymət və həm də arqumentə verilən qiymət vardır (əgər o, çağırılma zamanı buraxılmışdırsa). Aydındır ki, bir neçə susmaya görət arqument ola bilər:

```
Qaytarılan_qiymətin_tipi funksiyanın_adı(arq1=qiymət, arq2=qiymət)
```

Susmaya görə arqumentlər funksiyanın parametrlərinin siyahsının sağ kənarından başlayıb, daha sonra isə soldan sağa kəsilməz ardıcıllıqda ola bilər.

Məsələn:

```
void foot (int i, int j = 7) ;                          //yol verilən

void foot (int i, int j = 2, int k) ;                   //yol verilməyən

void foot (int i, int j = 3, int k = 7) ;               // yol verilən

void foot (int i = 1, int j = 2, int k = 3); // yol verilən

void foot (int i=- 3, int j);                           // yol verilməyən
```

Susmaya görə parametrlərlə işə aid nümunələrə baxaq.

```
#include <iostream>
using namespace std;

// ulduzlardan ibarət count uzunluqlu sətir verir
void Star(int count=20){
	for(int i=0;i<count;i++)
		cout<<'*';
	cout<<"\n\n";
}


void main(){

	Star(); // 20 ulduzdan ibarət sətrin göstərilməsi
	Star(5); // beş ulduzdan ibarət sətrin göstərilməsi

}
```

Bugünlük bu qədər. Test olmayacaq ancaq ev tapşırığını etməlisiniz. Uğurlar!!!

6. Ev tapşırığı

1. Arqumentlər olaraq, tam müsbət ədəd və bu ədədin çevrilməli olduğu say sisteminin əsasını arqument kimi qəbul edən funksiya yazın (say sistemi 2-dən 36-a qədərdir). Məsələn, 27 ədədini 16-lıq say sisteminə çevirərkən 1B; 13 ədədini 5-lik say sisteminə çevirərkən 23; 35-i 18-liyə çevirərkən isə 1H alınmalıdır.

2. «Zərlər» oyunu. Şərt: Üzərində 1-dən 6-a qədər qiymətlər olan 2 oyun zəri vardır. Oyun kompüterlə oynanılır. Zərlər növbə ilə atılır. 5 atışdan sonra zərlərin aldığı qiymətlərin cəmi daha böyük olan oyunçu qalib gəlir. Birinci gedişin kompüter və ya insan tərəfindən yerinə yetirilməsi halını əvvəldən nəzərə alın. Zərlər simvollar vasitəsi ilə təsvir olunur. Oyunun axırında hər iki iştirakçı üçün atışların orta qiymətini vermək lazımdır.