ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Dərs №12

C Proqramlaşdırma dili

Mündəricat

1. Yaddaşın statik və dinamik ayrılması 3
2. Göstəricilər 4
3. Göstəricilər və massiv 8
4. Göstəricilər – funksiya arqumentləri kimi.
Arqumentlərin göstərici kimi ötürülməsi 12
5. Ev tapşırığı 14

1. Yaddaşın statik və dinamik ayrılması

Statik yaddaş – bu bütün qlobal və statik dəyişənlərin saxlanılma sahəsidir. Statik yaddaşın dəyişənləri bir dəfə elan edilir və proqramın icrasından sonra ləğv edilir.

Dinamik yaddaş və ya sərbəst saxlama yaddaşı statik yaddaşdan onunla fərqlənir ki, proqram saxlanılacaq elementlər üçün aşkar şəkildə yaddaş tələb edir və yaddaş artıq lazım olmadıqda onu boşaldır.

Dinamik yaddaşla işləyərkən proqram, məsələn, icra mərhələsində massivin elementlərinin sayını öyrənməyə imkan yarada bilər.

2. Göstəricilər

Göstərici – başqa dəyişənin ünvanını saxlayan dəyişəndir. Göstəricilər C dilində geniş istifadə olunurlar. Bu ondan irəli gəlir ki, onlar bəzən lazım olan işi ifadə etmək üçün yeganə vasitə olurlar, başqa vasitələrdən fərqli olaraq proqramın daha yığcam və səmərəli olmasına şərait yaradırlar. Belə ki, göstərici obyektin ünvanını saxladığı üçün, bu obyektə dolayı yolla müraciət etməyə imkan verir.

Hesab edək ki, x int tipində dəyişəndir, px isə qeyd edilməmiş üsulla yaradılan göstəricidir. & unar operatoru obyektin ünvanını verir, belə ki, operator:

```
px = &x;
```

x-in ünvanını px dəyişəninə mənimsədir. Bu halda deyilir ki, px x-in göstəricisidir. & operatoru yalnız dəyişənlərə və massivin elementlərinə tətbiq edilir. Növbəti şəkildə:

```
&(x-1) və &3
```

qəbuledilməzdir. Registr dəyişəninin ünvanını da əldə etmək olmaz.

Unar * operatoru öz operandına nəticə ünvanı kimi baxır və məzmunu əldə etmək üçün bu ünvana baxır. Nəticə olaraq y də int tipindədir, onda:

```
y = *px;
```

y-ə px-in göstərdiyi ünvanın məzmunu mənimsədilir. Beləliklə, növbəti ardıcılıq

```
px = &x;
y = *px;
```

y-ə aşağıda verildiyi kimi eyni qiymət mənimsədir:

```
y = x;
```

bu prosesdə istifadə edilən bütün dəyişənləri növbəti şəkildə elan etmək lazımdır:

```
int x, y;
int *px;
```

x və y dəyişənlərinin elan edilməsi formasına biz dəfələrlə rast gəlmişik. Göstəricinin elan edilməsi isə:

```
int *px;
```

yeni bir şeydir və bu onu ifadə edir ki, *px kombinasiyası int tipindədir. Bu o deməkdir ki, əgər px *px şəklində verilirsə, bu int tipində dəyişənlə ekvivalentdir. Faktik olaraq dəyişənin elan edilməsi sintaksisi bu dəyişənin rast gəlindiyi ifadənin sintaksisini təqlid edir.

Bu qeyd edilənlər mürəkkəb təsvirlər üçün bütün hallarda faydalıdır. Məsələn:

```
double atof(), *dp;
```

onu deyir ki, atof() və *dp ifadələrdə double tipində qiymətlərlə istifadə edilir. Siz həmçinin, nəzərə almalısınız ki, bu elandan belə nəticə çıxır ki, göstərici yalnız müəyyən növ obyektləri göstərə bilər.

Göstəricilər ifadələrdə də istifadə oluna bilər. Məsələn, əgər px tam qiymətli x-in göstəricisidirsə, onda *px x-in rast gəlindiyi istənilən ifadədə ola bilər. Məsələn,

```
y = *px + 1;//y-ə x+1 qiymətini mənimsədir
cout<< *px;//x-in cari qiymətini çıxarır
d = sqrt((double) *px)//d-yə x-in kvadrat kökünü mənimsədir
/*eyni zamanda sqrt funksiyasının qiyməti qaytarılmadan əvvəl x-in qiyməti double tipinə çevrilir*/
```

Növbəti ifadədəki:

```
y = *px + 1;
```

* və & unar operatorları öz operandları ilə riyazı operatorlardan fərqli olaraq daha sıx bağlıdırlar. Ona görə də, bu cür ifadə px-in göstərdiyi qiyməti götürür, üzərinə 1 əlavə edir və nəticəni y dəyişəninə mənimsədir.

Biz tezliklə bu ifadənin nə demək olduğuna qayıdacağıq:

```
y = *(px + 1);
```

Göstəriciyə istinad mənimsətmənin istənilən yerində ola bilər. Əgər px x-in göstəricisidirsə, onda:

```
px = 0;
```

x in 0-a bərabər olduğunu bildirir.

```
*px += 1;
```

isə növbəti ifadə kimi x-in qiymətini 1 vahid artırır:

```
(*px) + 1;
```

Sonuncu ifadədəki dairəvi mötərizələr vacibdir, əgər onlar yazılmazsa, * və + unar əməliyyatlar prioritetə görə soldan saga doğru icra olunduğu üçün bu ifadə px-in göstərdiyi qiyməti deyil, onun öz qiymətini artırmış olur.

Və nəhayət, göstəricilər dəyişənlər olduqları üçün onlarla digər dəyişənlər kimi də rəfdar etmək olar. Əgər py int tipində dəyişənin başqa bir göstəricisidirsə, onda:

```
py = px;
```

px-in saxladığı qiyməti py-ə köçürür, nəticədə py elə px kimi eyni qiymətin göstəricisi olur.

3. Göstəricilər və massivlər

C dilində göstəricilər və massivlər arasında güclü bağlılıq var, bu o qədər güclüdür ki, göstəricilərə və massivlərə eyni zamanda baxmaq lazımdır.

Massivin indeksi ilə icra olunan istənilən əməliyyatı göstəricilər vasitəsi ilə də icra etmək olar.

Göstərici ilə olan üsul daha sürətlidir, lakin anlama baxımından, xüsusilə də yeni başlayanlar üçün nisbətən çətindir.

Təyin:

```
int a[10];
```

10 ölçüdə, a[0], a[1], ..., a[9] adlanan 10 ardıcıl obyektdən ibarət yığımı təyin edir. a[i] massivin başdan i-ci mövqeyinə işarə edir. Əgər pa tam tipli göstərici kimi elan edilmişdirsə:

```
int *pa;
```

onda növbəti mənimsətmə nəticəsində:

```
pa = &a[0]
```

pa a massivinin 0-ci elementinin göstəricisi olur. Bu o deməkdir ki, pa a[0] elementinin ünvanını saxlayır. Növbəti mənimsətmə:

```
x = *pa
```

a[0]-ın qiymətini x-ə köçürəcək. Əgər pa a massivinin hər hansı bir elementinin göstəricisi olarsa, pa+1 növbəti elementin göstəricisi olacaq, pa-i pa-nın göstəricisi olduğu elementdən i mövqeyindən əvvəldə yerləşən elementi göstərir, pa+i isə sonrakı i mövqeyində yerləşən elementi göstərir.

Beləliklə, əgər pa a[0] elementinin göstəricisidirsə:

```
*(pa+1)
```

a[1]-in qiymətinə, pa+i - a[i] ünvanına, *(pa+i) isə a[i]-nin qiymətinə istinad edir.

Bu qeydlər a massivinin elementlərinin tipindən asılı olmayaraq qanunauyğundur. «göstəriciyə 1 əlavə etmək», həmçinin onun üzərində hesabi əməliyyat aparmaq o deməkdir ki, göstəricinin göstəricisi olduğu obyektin yerləşdiyi yaddaşın ölçüsü qədər keçid baş verir. Beləliklə pa+i ifadəsindəki i əlavə edilməzdən əvvəl pa-nin göstəricisi olduğu obyektin yaddaşdakı ölçüsünə vurulur.

Aydın olduğu kimi, indeksləşdirmə və göstərici hesabları arasında bağlılıq var.

Həqiqətdə, kompilyator massivə istinadı massivin əvvəlinin göstəricisinə çevirir. Bunun nəticəsində massivin adı göstərici ifadəsi olur. Buradan bir neçə faydalı nəticə çıxır. Belə ki, massivin adı onun sıfırıncı elementinin mövqeyinin sinonimidir:

```
pa = &a[0]
```

mənimsətməsini pa = a kimi də yazmaq olar.

Daha təəccüblüsü odur ki, ilk baxışda heç olmazsa, a[i] elementinə istinadı *(a+i) şəklində də yazmaq olar.

a[i] ifadəsini analiz edən zaman o dərhal *(a+i)-yə çevrilir. Bu iki forma tamamilə ekvivalentdirlər. Əgər bu cür ekvivalentlik əlaqəsinin hər iki tərəfinə & operatorunu tətbiq edəriksə, onda &a[i] və a+i ifadələrinin də eyni olduğunu alarıq: a+i a-nın başlanğıcından i-ci ünvandır. Digər tərəfdən, pa göstəricidirsə, onda ifadələrdə onu pa[i] və ya *(pa+i) kimi istifadə edə bilərik.

Qısaca olaraq, massiv və indekslər ehtiva edən istənilən ifadə göstəricilər, onların qarışığı və əksinə istifadə oluna bilər. Bu hətda eyni ifadədə də ola bilər.

Massivin adı ilə göstərici arasındakı bir fərqi nəzərə almaq lazımdır. Göstərici dəyişəndir, ona görə də, pa=a və pa++ əmrləri məna kəsb edir. Lakin massivin adı sabitdir, dəyişən deyil:

a=pa və ya a++ və ya p=&a yazılışları yolverilməzdir.

Massivin adı funksiyaya ötürüldüyü zaman əslində funksiyaya massivin başlanğıc ünvanı ötürülmüş olur. Çağırılan funksiya daxilində bu cür arqument digərləri kimi dəyişən sayılır. Belə ki, massivin adı arqument kimi həqiqətən də göstəricidir, yəni unvan saxlayan dəyişəndir.

```
/* ekranda m-i göstərir*/
   void ShowElements(int *m, int size)
   {
        int n;
        for (n = 0; n < size; m++,n++)
                cout<<*m<<"\t";


    }
```

m-in artirilması əməliyyatı tamamilə qanunauyğundur, belə ki, bu dəyişən göstərici olduğu üçün m++ ShowElements funksiyasında emal edilən massivə təsir etmir, yalnız ShowElements funksiyası üçün ünvanın lokal nüsxəsini artırır.

Funksiyanın təyinində formal parametrlərin növbəti şəkildə təsviri:

```
int m[];
```

və

```
int *m;
```

tamamilə ekvivalentdirlər; funksiyanın yazılışı zamanı hansı ifadələrin istifadə ediləcəyindən asılı olaraq bu təyinlərdən hansına üstünlük verəcəyinizə qərar verə bilərsiz. Əgər funksiyaya massivin adı ötürülərsə, hansının daha rahat olduğundan asılı olaraq, ehtimal etmək olar ki, funksiya ya massiv, ya da göstərici üzərində əməliyyat aparır və bundan sonra uyğun şəkildə fəaliyyət göstərmək lazımdır. Əgər uyğun görülərsə, hər iki əməliyyatdan istifadə etmək olar.

4. Göstəricilər – funksiyanın arqumentləri kimi. Arqumentlərin göstəriciyə görə ötürülməsi

Belə ki, C-də arqumentlərin funksiyaya ötürülməsi qiymətə görə həyata keçirildiyi üçün, çağırılan prosedur çağıran proqramdakı dəyişənin qiymətini birbaşa dəyişdirə bilmir. Bu zaman əgər siz, həqiqətən də dəyişənin qiymətini dəyişdirmək istəyərsinizsə, nə etməlisiniz? Məsələn, çeşidləmə proqramı swap adlı funksiya vasitəsilə elementin yerini dəyişdirmək istəyir. Bunun üçün:

```
swap(a, b);
```

yazmaq lazımdır və swap funksiyasını növbəti şəkildə təyin etməlisiz:

```
void swap(x, y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

Qiymətə görə çağırmaqdan dolayı swap funksiyası çağırılan funksiyanın a və b arqumentlərinə təsir edə bilmir. Xoşbəxtlikdən, lazım olan səmərəliliyi əldə etmək üçün imkan vardır. Çağıran proqram qiyməti dəyişdirilə bilən göstəricilər göndərə bilir:

```
swap(&a, &b);
```

Belə ki, & operatoru dəyişənin ünvanını verdiyi üçün, &a a-nın göstəricisidir. swap funksiyasında isə arqumentlər göstərici kimi təyin edilir və faktiki operandlara müraciət onlar vasitəsilə həyata keçirilir:

```
void swap(px, py)
  {
        int tmp;


         temp = *px;
         *px = *py;
         *py = temp;
   }
```

5. Ev tapşırığı

1. Tam ədədlər massivi verilir. Göstəricilərdən isitfadə etməklə massivin cüt və tək indeksli elementlərinin yerlərini dəyişdirin (yəni, massivin cüt nömrəli mövqedə yerləşən elementini tək nömrəli mövqedə yerləşən elementlə dəyişdirin.

2. Artma ardıcıllığında çeşidlənmiş iki massiv verilir: A[n] və B[m]. A və B massivlərinin elementlərini ehtiva edən artma ardıcıllığında çeşidlənmiş C[n+m] massivini qurun.

3. A[n] və B[m] massivləri verilir. Üçüncü massivi növbəti şəkildə qurmaq lazımdır:

- Hər iki massivin elementlərindən;
- İki massivin ortaq elementlərindən;
- A massivinin B massivində olmayan elementlərindən;
- B massivinin A massivində olmayan elementlərindən;
- A və B massivlərinin ortaq olmayan elementlərindən (yəni, bu massivlərin birləşməsindən).