ШАГ компьютерная академия

# ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

# Dərs №13

## İstinadlar, new və delete operatorları

## Mündəricat

# 1.İstinadlar haqqında ümumi məlumat

Bu dərsdən etibarən parametrlərin başqa cür, xüsusilə də istinadlardan istifadə etməklə ötürülməsinə baxacağıq.

Dəyişənlərə alternativ müraciət üsulu kimi göstəricilərin istifadəsi göstəricidə saxlanılan ünvan dəyişdiyi zaman təhlükəlidir, belə ki, bu göstərici lazım olan qiymətə bir daha istinad etmir.

C dili göstərici vasitəsilə dəyişənə alternativ müraciət üsulu təqdim edir. İstinad dəyişən elan etməklə göstərici kimi başqa qiymətə istinad edən obyekt yaratmaq olar, lakin göstəricidən fərqli olaraq həmişə bu qiymətə bağlı olur. Beləliklə,

***Qiymətə istinad həmişə bu qiymətə istinad edir.***

İstinadı növbəti şəkildə elan etmək olar:

```
<tipin adı>& <istinadın adı> = <ifadə>;
                və ya
<tipin adı>& <istinadın adı>(<ifadə>);
```

Belə ki, istinad artıq mövcud olan obyektin başqa bir adı olduğu üçün, qiymətləndirilən obyekt kimi yaddaşda yerləşən hər hanası bir obyektin adı kimi çıxış etməlidir. Qiymətləndirmə ilə bağlı verilən təyindən sonra istinadın qiyməti bu obyektin ünvanı olur. Bunu nümunə əsasında nümayiş etdirək:

```
#include <iostream>
using namespace std;
void main()
 {
    int  ivar  =  1234;  //  dəyişənə  qiymət  mənimsədilir.
    int *iptr = &ivar; // göstəriciyə ivar dəyişəninin
                       // ünvanı mənimsədilir
    int &iref = ivar;  // İstinad ivar ilə
                       // əlaqələndirilir.
    int *p = &iref;    // Göstəriciyə ivar dəyişəninin
                       //ünvanı mənimsədilir.

    cout << "ivar = " << ivar << "\n";
    cout << "*iptr = " << *iptr <<"\n";
    cout << "iref = " << iref << "\n";
    cout << "*p = " << *p << "\n";
 }
```

Proqramın icrasının nəticəsi:

```
ivar = 1234
*iptr = 1234
iref = 1234
*p = 1234
```

***Proramın şərhi.*** Burda dörd dəyişən elan edilir. ***ivar*** dəyişəni 1234 qiymətini alır. Tam ****iptr*** göstəricisinə ***ivar.*** Dəyişəninin ünvanı mənimsədilir. ***iref*** dəyişəni istinad kimi elan edilir. Bu dəyişən öz qiyməti kimi ***ivar*** dəyişəninin yaddaşda yerləşdiyi ünvanı qəbul edir. Növbəti operator:

```
cout << "iref = " << iref << "\n";
```

ekrana ***ivar*** dəyişəninin qiymətini verir. Bu onunla izah edilir ki, ***iref ivar*** dəyişəninin yaddaşdakı yerinə ***istinaddir***.

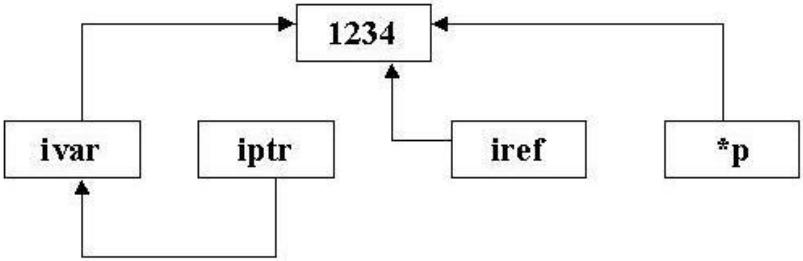
Sonuncu ***int *p = &iref;*** elanı ***iref***-də saxlanılan ünvanın mənimsədildiyi daha bir göstərici yaradır. Növbəti sətir:

```
int *iptr = &ivar;
```

və

```
int *p = &iref;
```

eyni nəticəni verir. Onların hər birində ***ivar***-a istinad edən göstəricilər yaradılır. Şəkil.1-də




verilmiş proqramdakı dəyişənlərin qarşılıqlı əlaqəsi nümayış etdirilmişdir:

İstinadların istifadəsi zamanı bir qaydanı xatırlamaq lazımdır: bir dəfə qiymətləndirilmiş istinada başqa qiymət mənimsətmək olmaz! Bütün bu ifadələr:

```
a) int iv = 3;          b) iref++;      c) iref = 4321;
   iref = iv;
```

***ivar*** dəyişəninin dəyişməsinə səbəb olacaq

*Qeyd*

1. Elan edilərkən qiymətləndirilməmiş və ya (NULL) qiyməti almış göstəricilərdən fərqli olaraq, istinadlar həmişə obyektə istinad edirlər. İstinad yaradıldığı zaman MÜTLƏQ qiymətləndirilməlidir və 0 qiyməti ala bilməz.
2. İsitinadları növbəti hallarda qiymətləndirmək olmaz:
   - funksiyanın parametri kimi istifadə edildiyi zaman.
   - funksiyanın qaytarılan qiymətinin tipi kimi.
   - siniflərin elan edilməsində.
3. İstinadlar üzərində birbaşa əməliyyat aparan operator mövcud deyil!

# 2. İstinad parametrlər. Arqumentlərin istinad kimi ötürülməsi

İstinad dəyişənlər az hallarda istifadə edilir: dəyişənin özünü istifadə etmək ona istinaddan istifadə etməkdən daha rahatdır. İstinadlar funksiya parametri kimi daha geniş istifadə edilir. İstinadlar xüsuslilə bir neçə obyekt (qiymət) qaytaran funksiyalarda daha faydalıdır. Yuxarıda deyilənləri nümayiş etdirmək üçün növbəti proqrama baxaq:

```
#include <iostream>
using namespace std;
// Göstərici istifadə etməklə yerdəyişdirmə.
void interchange_ptr (int *u,int *v)
{
    int temp=*u;
    *u = *v; *v = temp;
}

/* ------------- */
// İstinad istifadə etməklə yerdəyişdirmə.
void interchange_ref (int &u,int &v)
{
    int temp=u;
    u = v; v = temp;
}

/* -------------
*/ void main ()
{
```

```
    int x=5,y=10;
    /* ---------------------------------                    */
    cout << "Change with pointers:\n";
    cout << "x = " << x << "  y = " <<y <<"\n";
    interchange_ptr (&x,&y);
    cout << "x = " << x << "  y = " << y <<"\n";
    cout << "-----------------------------                  " <<"\n";
    cout << "Change with references:\n";
    cout << "x = " << x << "  y = " << y <<"\n";
    interchange_ref (x,y);
    cout << "x = " << x << "  y = " << y <<"\n";
}
```

***interchange_ptr()*** funksiyasında parametrlər göstərici kimi verilmişdir. Buna görə də funksiyanın gövdəsində onların yenidən adlandırılması yerinə yetirilir, bu funksiyaya müraciət zamanı faktik dəyişənlər kimi o dəyişənlərin ***(&x,&y)*** ünvanları istifadə edilir ki, onların qymətlərinin yeri dəyişdirilməlidir. ***interchange_ref()*** funksiyasında parametrlər istinadlardır. İstinadlar funksiyanın gövdəsindən proqramda təyin edilmiş adi dəyişənlər kimi istifadə edilən faktik parametrlərə müraciəti təmin edir.

İstinadlar və göstəricilər funksiyanın parametrləri kimi sıx əlaqəlidirlər. Növbəti böyük olmayan funksiyaya baxaq:

```
void f(int *ip)
{
    *ip = 12;
}
```

Bu funksiyanın daxilində ünvanı ***ip*** göstəricisində saxlanılan ötürülüən arqumentə müraciət həyata keçirilir:

```
f(&ivar); // ivar ünvanının ötürülməsi
```

Funksiyanın daxilindəki ****ip = 12***; ifadəsi ünvanı ***f()*** funksiyasına ötürülmüş ***ivar*** dəyişəninə 12 qiymətini mənimsədir. İndi də istinad parametrlər istifadə edən oxşar funksiyaya baxaq:

```
void f(int &ir)
{
   ir = 12;
}
```

***ip*** göstəricisini 12 qiyməti mənimsədilən ***ir,*** istinadı ilə əvəz edək:

```
f(ivar); // ivar-ın istinada görə ötürülməsi
```

İstinad edilən obyektə qiymət mənimsədilir: ***ivar***-ı istinada görə göndərir. Belə ki, ***ir ivar***-ra istinad etdiyi üçün, ***ivar***-a 12 qiyməti mənimsədilir.

Artıq biz istinadlarla tanış olduqdan sonra, növbəti bölməyə keçək və onların təyinatlarından birinə baxaq.

# 3. İstinadlar funksiyanın nəticəsi kimi

Burada biz istinadların funksiyanın nəticəsi kimi istifadəsinə baxırıq.

Funksiyalar obyektə istinadları o zaman qaytara bilər ki, bu obyektlər ***funksiya aktiv olmadığı zaman mövcud olsun.*** Beləliklə, funksiyalar lokal avtomatik dəyişənlərə istinad qaytara bilməz. Məsələn, növbəti şəkildə təyin olunmuş funksiya üçün:

```
double &rf(int p);
```

tam tipli arqument lazımdır və o təxminən hardasa başqa bir yerdə elan edilmiş ***double*** tipli obyektə istinadı qaytarır. Deyilənləri konkret misalla nümayiş etdirək.

*Nümunə 1. İkiölçülü massivin eyni ədədlərlə doldurulması*

```
#include  <iostream>
using      namespace
std; int a[10][2];
void main ()
{
    int & rf(int index); // funksiyanın prototipi
    int b;
    cout << "Fill array.\n";
    for (int i=0;i<10;i++)
    {
```

```
        cout << i+1 << " element: ";
        cin >> b;
        a[i][0] = b;
        rf(i) = b;
    }
    cout << "Show array.\n";
    cout << "1-st column 2-nd column" << "\n";
    for (int i=0;i<10;i++)
        cout << a[i][0] << "\t\t" << rf(i) << "\n";
}

int &rf(int index)
{
    return a[index][1]; // Massivin elementinə
                        // istinadı qaytarmaq

}
```

Burada tam ədədlər ehtiva edən qlobal ***a*** massivi elan edilir. ***main()*** funksiyasının əvvəlində birmənalı şəkildə ***index*** parametri ilə eyni olan ***a*** massivinin ikinci sütununun tam qiymətinə istinadı qaytaran ***rf()*** istinad funksiyasının prototipi yerləşir.

Belə ki, ***rf()*** funksiyası tam qiymətə isitinadı qaytardığı üçün funksiyanın adı mənimsətmə operatorunun sol tərəfində yerləşə bilər ki, bu da növbəti sətirdə nümayış etdirilmişdir:

```
rf(i) = b;
```

*Nümunə 2. Massivin ən böyük qiymətinin tapılması və onun sıfırla dəyişdirilməsi*

```
#include <iostream>

using namespace std;

// Funksiya massivin ən böyük qiymətli
// elementinə istinadı təyin edir.

int &rmax(int n, int d[])
{
    int im=0;
    for (int i=1; i<n; i++)
       im = d[im]>d[i]?im:i;
    return d[im];
}

void main ()
{
    int x[]={10, 20, 30,
    14}; int n=4;
    cout << "\nrmax(n,x) = " << rmax(n,x) <<
    "\n"; rmax(n,x) = 0;
    for (int i=0;i<n;i++)
    cout << "x[" << i << "]=" << x[i] << " ";
    cout << "\n";
}
```

Proqramın icrasının nəticəsi:

```
rmax (n,x) = 30
x[0]=10  x[1]=20  x[2]=0  x[3]=14
```

Növbəti sətrin icrası zamanı:

```
cout << "\nrmax(n,x) = " << rmax(n,x) << "\n";
```

birinci arqumenti massivin elementlərinin sayı, ikinci isə - massivin özü olan ***rmax()*** funksiyasına ilk müraciət baş verir. Nəticədə massivin ən böyük qiymətinə istinad qaytarılır ki, bu qiymət də ekrana verilir. Növbəti sətrin icrası zamanı:

```
rmax(n,x) = 0;
```

***rmax()*** funksiyasına yenidən müraciət edilir. İndi isə tapılmış istinada əsasən ən böyük qiymət 0-la əvəzlənir.

# 4. Yaddaşın ayrılması operatorları: new və delete

## Yaddaşın ayrılması əməliyyatı - new

Yuxarıda qeyd edilən əməliyyat vasitəsilə proqramın icrası mərhələsində yaddaşın dinamik ayrılmasını təmin edə bilərik.

Adətən new operatu ehtiva edən ifadə növbəti şəkildə olur:

```
tip_göstəricisi = new tipin_adı (qiymətləndirici)
```

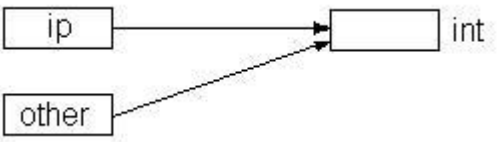
Qiymətləndirici – massivdən başqa bütün tiplər üçün istifadə oluna bilən vacib olmayan qiymətləndirmə ifadəsidir.

Operatorun icrası zamanı

```
int *ip = new int;
```

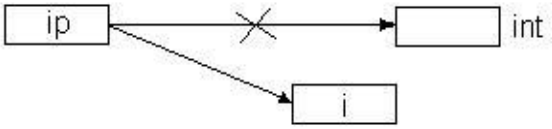
2 obyekt yaradılır: dinamik adsız obyekt və qiyməti dinamik obyektin ünvanı olan onun ip adlı göstəricisi. Eyni dinamik obyektin başqa göstəricisini də yaratmaq olar:

```
int *other=ip;
```



Əgər ***ip*** göstəricisinə başqa qiymət mənimsədilərsə, onda dinamik obyektə müraciətdən məhrum ola bilərik:

```
int *ip=new
(int); int i=0;
ip=&i;
```



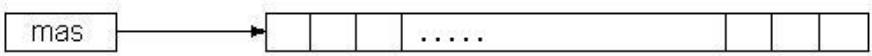

Nəticədə dinamik obyekt əvvəlki kimi mövcud olacaq, lakin ona müraciət artıq mümkün olmayacaq. Bu cür obyektlər "zibil" adlanırlar.

Yaddaşın ayrılması zamanı obyektə qiymət vermək olar:

```
int *ip = new int(3);
```

Yaddaşın massiv kimi dinamik ayrılması da mümükündür:

```
double *mas = new double [50];
```




Sonra bu cür dinamik ayrılmış yaddaşla adi massiv kimi işləmək olar:

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

#include <time.h>
```

```
using namespace
std; void main(){
    srand(time(NULL));
    int size;
    int * dar;
    /* massivin ölçüsünün klaviaturadan
    daxil edilməsi*/
    cout<<"Enter size:\n";
    cin>>size;
    /*yaddaşın elementlərin sayı size ilə massiv üçün
    ayrılması*/
    dar=new int [size];

    if(!dar){
        cout<<"Sorry, error!!!";
        exit(0);//funksiya proqramdan çıxışı təşkil edir
    }

    //massivin doldurulması və ekrana verilməsi
    for(int i=0;i<size;i++){
        dar[i]=rand()%100;
        cout<<dar[i]<<"\t";
    }
    cout<<"\n\n";
}
```

new əməliyyatı müvəffəqiyyətlə icra edildikdən sonra qiyməti sıfırdan fərqli olan göstərici qaytarır.

0-a, yəni, ***NULL*** sıfır göstəricisinə bərabər olan əməliyyatın nəticəsi onu bildirir ki, lazım olan boş ardıcıl yaddaş sahəsi tapılmamışdır.

## Yaddaşın azad edilməsi əməliyyatı - delete

***delete*** əməliyyatı ***new*** əməliyyatı ilə əvvəlcədən təyin olunmuş yaddaşı gələcək istifadə üçün azad edir.

```
delete ip; /* int tipində dinamik obyekti ləğv
           edir
delete [ ] mas; // double *mas =new double[50];
                // olarsa, uzunluğu 50 olan dinamik
                // massivi ləğv edir
```

***NULL*** göstəricisinə əməliyyat tətbiq etmək tamamilə təhlükəsizdir. Nəticədə eyni göstəriciyə delete əməliyyatının təkrar tətbiq edilməsi təyin edilməmişdir. Adətən dövrdən çıxmama səhvi baş verir.

Bu cür səhvlərdən qurtulmaq üçün növbəti strukturu tətbiq etmək olar:

```
int *ip=new int[500];
. . .
if (ip){
    delete ip; ip=NULL;
}
else
{
    cout <<" yaddaş artıq azad edilmişdir. \n";
}
```

Yuxarıda şərh edilən nümunəyə artıq yaddaşın azad edilməsini də əlavə edə bilərik.

```
#include <iostream>
#include <stdlib.h>
#include <time.h>

using namespace std;

void main(){

    srand(time(NULL));

    int size;
```

```
    int * dar;
    /*massivin   ölçüsünün   klaviaturadan
    daxil edilməsi*/
    cout<<"Enter size:\n";
    cin>>size;
    /*yaddaşın   elementlərinin   sayı   size   olan
    massiv kimi ayrılması*/
    dar=new int
    [size]; if(!dar){
    cout<<"Sorry, error!!!";
    exit(0);/*funksiya proqramdan çıxışı
    təşkil edir
    }
    /*massivin   doldurulması   və   ekrana
    verilməsi*/
        for(int i=0;i<size;i++){
        dar[i]=rand()%100;
        cout<<dar[i]<<"\t";
    }
    cout<<"\n\n";
    //yaddaşın azad edilməsi
    delete[]dar;
}
```

# 5. Ev tapşırığı

1. Göstəricinin göstəricisi vasitəsilə iki ədədin cəmini hesablayıb üçüncüsünə mənimsədin.
2. Yalnız göstəricilərdən istifadə etməklə sadə kalkulyator yazın.
3. Yalnız göstəricilərdən istifadə etməklə ədədin faktorialını tapın.
4. Yalnız göstəricilərdən istifadə etməklə ədədin verilmiş qüvvətini tapın.
5. Sıfra bölmə əlamətini göstəricinin göstəricisi vasitəsilə təyin etməli.