

Obyektyönlü programlaşdırma

C++



Dərs №4

C++ dili ilə
obyektyönlü
proqramlaşdırma

Mündəricat

İnkrement və dekrementə yükləmə	3
İndeksləşdirmə operatoruna yükləmə	10
Yekun fəsil	13
Ev tapşırığı	14

İnkrement və dekrementə yükləmə

Keçən dərsimizdə biz sizinlə operatorlara yükləməni araşdırdıq və sətirlə işi təşkil edən sinfə aid nümunəyə baxdıq. Bu sinfdə binary "=" və "+" operatorlarına yükləmədən istifadə edilirdi. Lakin, biz sizinlə unar operatorlara yükləməyə də baxsaq pis olmaz. Xüsusilə də, inkrement və dekrement operatorlarına. Unar xüsusiyyətindən başqa bu operatorlardan hər biri iki formada olur ki, bu da onların yüklənməsi zamanı böyük əhəmiyyət kəsb edir.

Qeyd etmək lazımdır ki, C++ dilinin ilkin versiyalarında ++ və -- əməliyyatlarına yükləmə zamanı postfix və prefiks formaları arasındakı fərq qoyulmurdu. Məsələn:

```
#include <iostream>
using namespace
std; class Digit{
    //Tam ədəd
    int N;
public:
    //Parametrli konstruktor
    Digit(int n)
    {
        N = n;
    }

    //ədədi ekranda göstərmək üçün funksiya
    void display()
```

```

{
    cout << "\nDigit: N = " << N << "\n";
}

//Üzv funksiya (forma fərqlənmir):
Digit& operator -- ()
{
    //Obyektin qiymətini 10 dəfə azaldırıq
    //və onu operatorun çağırıldığı yere
    //qaytarırıq
    N /= 10;
    return *this;
}
};

void main()
{
    //Z obyektini yaradıırıq, məhz onu sınaqdan
    keçirəcəyik

    Digit Z(100);

    //Obyektin ilkin formasında
    //göstərilməsi
    cout<<"\nObject Z (before):\n";
    Z.display();

    cout<<"\n-----\n";

    //Pref obyektinə prefiks formasında
    //ifadəni mənimsədirik (verilmiş halda
    //əvvəlcə Z dəyişəcək, sonra isə
    //mənimsətmə baş verəcək
    Digit Pref=--Z;

    //prefiks formasının işinin nəticəsini
    //göstəririk
    cout<<"\nPrefix\n";
    cout<<"\nObject Pref:\n";
    Pref.display();
}

```

```

cout<<"\nObject Z (after):\n";
Z.display(); cout<<"\n-----
-----\n";

//Postf obyektinə postfiks formasında
//ifadəni mənimsədirik (verilmiş halda,
//təəssüf ki, yenə əvvəlcə Z dəyişəcək,
//sonra isə mənimsətmə baş verəcək

Digit Postf=Z--;

//postfiks formasının işinin nəticəsini
//göstəririk
cout<<"\nPostfix\n";
cout<<"\nObject Postf:\n";
Postf.display();
cout<<"\nObject Z (after):\n";
Z.display();
}

```

Programın icrasının nəticəsi:

Object Z (before):
Digit: N = 100

Prefix
Object Pref:
Digit: N = 10
Object Z (after):
Digit: N = 10

Postfix
Object Postf:
Digit: N = 1
Object Z (after):
Digit: N = 1

C++ dilinin müasir versiyasında növbəti qayda qəbul edilmişdir:

- **++ və --** prefiks əməliyyatlarına yükləmə digər unar operatorlara yükləmədən heç nə ilə fərqlənmir. Başqa sözlə, konkret sinfin funksiyaları: **operator++** və **operator--**, bu sinif üçün prefiks əməliyyatlarını təyin edir.
- **++ və --** postfiks əməliyyatlarının təyin edilməsi zamanı funksiyalar daha bir int tipində əlavə parametmə malik olmalıdırlar. Proqramda postfiks ifadə istifadə edildiyi zaman, int tipində parametri olan funksiya çağırılır. Bu zaman parametmə göndərmək lazım deyil, funksiyada onun qiyməti isə sıfıra bərabər olacaq.

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

//Ədədlər çütlüyü ilə işləmək üçün sinif
class Pair{
    //Tam ədəd
    int N;
    //Həqiqi ədəd
    double x;
public:
    //Parametrləri olan konstruktor
    Pair(int n, double xn)
    {
        N = n;
        x = xn;
    }
    //Verilənləri ekranda göstərmək
    üçün funksiya
    void display()
    {
        cout << "\nPair: N = " << N << " x = " << x << "\n";
    }
};
```

```
//Üzv funksiya (prefiksli --):
Pair& operator -- ()
{
    //Obyektin qiymətini 10 dəfə azaldırıq və
    //onu operatorun çağırıldığı yere
    //qaytarırıq

    N /= 10;
    x /= 10;
    return *this;
}

// Üzv funksiya (postfiksli --):
Pair& operator -- (int k)
{
    //Obyektin qiymətini müvəqqəti
    //olaraq asılı olmayan Pair
    //tipində dəyişəndə saxlayırıq
    //(Burada int k əlavə parametrimin
    //qiymətinin istifadə edilməsinə
    //cəhd onun sıfıra bərabər olmasını
    //təsdiqləyir)
    Pair temp(0,0.0);
    temp.N=N+k;
    temp.x=x+k;

    //obyekti 10 dəfə azaldırıq
    N /= 10;
    x /= 10;
    //obyektin əvvəlki qiymətini
    //qaytarırıq
    //bu cür taktik gedişlə biz postfiks
    //formasının səmərəliliyini əldə
    //etmiş oluruq, yəni, A=B++ halında
    //A-ya B-nin cari qiyməti
    //mənimsənilir (B dəyişmiş olsada)
    return temp;
}

};
```

```

void main()
{
    //Z obyektini yaradırıq, məhz onu sınaqdan
    //keçirəcəyik

    Pair Z(10,20.2);

    //Obyektin ilkin formasında göstərilməsi
    cout<<"\nObject Z (before):\n";
    Z.display();

    cout<<"\n-----\n";

    //Pref obyektinə prefiks formasında
    //ifadəni mənimsədirik (verilmiş halda
    //əvvəlcə Z dəyişəcək, sonra isə
    //mənimsətmə baş verəcək
    Pair Pref=--Z;
    //prefiks formasının işinin
    //nəticəsini göstəririk
    cout<<"\nPrefix\n"; cout<<"\nObject
    Pref:\n";
    Pref.display();
    cout<<"\nObject Z
    (after):\n"; Z.display();

    cout<<"\n-----\n";

    //Postf obyektinə postfiks formasında
    //ifadəni mənimsədirik (verilmiş halda,
    //əvvəlcə mənimsətmə baş verəcək, sonra
    //isə Z dəyişəcək

    Pair Postf=Z--;

    //postfiks formasının işinin nəticəsini
    //göstəririk

```

```

    cout<<"\nPostfix\n";
    cout<<"\nObject Postf:\n";
    Postf.display();
    cout<<"\nObject Z (after):\n";
    Z.display();
}

```

Programın icrasının nəticəsi:

```

Object Z (before):
Pair: N = 10 x = 20.2
-----

```

```

Prefix
Object Pref:
Pair: N = 1 x = 2.02
Object Z (after):
Pair: N = 1 x = 2.02
-----

```

```

Postfix
Object Postf:
Pair: N = 1 x = 2.02
Object Z (after):
Pair: N = 0 x = 0.202

```

Qeyd: Yuxarıda şərh edilmiş iki misalda yaradılma zamanı bir obyektin digəri tərəfindən qiymətləndirilməsinin olmamasına baxmayaraq, biz köçürmə konstruktorundan istifadə etmirik. Bu onunla bağlıdır ki, bu vacib deyildir, belə ki, burada bitlərlə köçürmə kritik xarakter daşıyır. Buna görə də kodu artıq konstruktorla yükləməyə ehtiyac yoxdur.

İndeksləşdirmə operatoruna yükləmə

Elə indicə biz sizinlə inkrement və dekrementə yükləmə xüsusiyyətlərini araşdırdıq. İndi isə daha bir xüsusi operatorla – indeksləşdirmə operatoru (`[]` kvadrat mötərizələr) ilə tanış olaq.

Beləliklə, məntiqi olaraq təklif etmək olar ki, **A [i]** ifadəsi, burada **A** — class abstrakt tipində olub, kompilyatör tərəfindən **A.operator [] (i)** kimi təqdim edilir. Misala baxaq:

```
#include <iostream>
using namespace
std; class A{

    //int tipində 10 elementdən ibarət massiv

    int a[10];

    //massivin ölçüsü
    int size;

public:

    //parametrlərsiz konstruktor
    A(){
        size=10;
        for (int i = 0; i < 10; i++)

        //aydındır ki, burada A sinfinin konstruktorunda
        //istifadə edilmiş [] operatoru,
```

```
//standartdır, belə ki, o int tipli massivin
//adı üzərində icra edilir.
        a [i] = i + 1;
    }
    //indeksləşdirmə operatoruna yükləmə
    //OBYEKT[i]=QİYMƏT halında istinada
    //görə indeksləşdirmənin çağırıldığı
    //yerə geri qaytarma kimi obyektin özü
    //qayıdır

    int& operator[] (int j){

        //konkret obyektin qaytarılması
        return a [j];
    }
    //massivin ölçüsünü qaytaran funksiya
    int Get () const {
        return size;
    }
};

void main () {
    int i,j;
    //A tipində bir obyektə iş
    A object;
    cout<<"\nOne object:\n";
    for(i=0;i<object.Get();i++)
    {
        //array[i] object.operator [] (j) kimi
        //interpretasiya olunur
        cout<<object[i]<<" ";
    }
    cout<<"\n\n";

    //A tipində obyektlər massivi ilə iş
    A array [3];
```

```

cout<<"\nArray of objects:\n";
for(i=0;i<3;i++){
    for(j=0;j<object.Get();j++){

        //array[i][j] (array [i]).operator [] (j) kimi
        //interpretasiya olunur
        //ilk iki [] əməliyyatı standartdır, belə ki,
        //massivin adı üzərində icra edilir
        //Bu zaman onun elementinin hansı tipdə olması
        //əhəmiyyət kəsb etmir
        //ikinci [] - təyindir, belə ki, birinci
        //[] operatorunun nəticəsi A tipində obyektidir
            cout << array [i][j] << " ";
        }
        cout << "\n\n";
    }
}

```

Programın icrasının nəticəsi:

One object:

1 2 3 4 5 6 7 8 9 10

Array of objects:

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

Qeyd: Diqqətə alın!!! Verilmiş misalda, biz ikiölçülü massivdə ikiqat kvadrat mötərizələrə yükləmirik. Biz sadəcə [] operatoruna yüklənən sinfin massiv obyektlərini yaradıırıq.

Yekun fəsil

...proqramlaşdırmadan başqa digər ixtisaslara yönəlmələri üçün. İş ondadır ki, növbəti dərsdən etibarən ixtisaslara görə bölünmə olacaq. O tələbələr ki, dizayn və administrasiya ixtisaslarını seçirlər onlar bizimlə vidalaşırlar. Bununla bağlı olaraq, bugün siz bütün kəsrlərinizi verə bilmək üçün gözəl imkan əldə edirsiniz. Xüsusilə də 19-cu dərsdə altıncı tapşırıqlar üzrə C imtahanını verə bilmək üçün. Eyni zamanda verilmiş kursa görə bütün yerinə yetirilmiş ev tapşırıqlarının olmasının təsdiqlənməsi C++-n əsaslarından məqbulun (zaçot) verilməsi.

Bundan başqa, bu dərsdə siz C++ kursunun bütün öyrənilmiş mövzularına ait testlər və bizimlə qalacaq olanlar üçün ev tapşırığı əldə edəcəksiniz.

MÜVƏFFƏQİYYƏTLƏR ARZULAYIRIQ!!!

Ev tapşırığı

1. Massivin sərhədlərini aşmasını yoxlayan dinamik massiv sinfini yaradın. Operatorlara yükləyin: [], =, +, -, ++ (massivin sonuna element əlavə etmə), — (massivin sonundan elementin silinməsi).