Лекція 6. Перевантаження операторів

В С++ існує можливість доозначати дію стандартних операторів (додавання, віднімання, множення і т.п. для випадку, коли операндами є об'єкти класів, створених користувачем). Така процедура називається перевантаженням операторів

Перевантаження операторів реалізується шляхом створення відповідної операторної функції.

Операторна функція може бути:

1. Зовнішньою (але, як правило, дружньою) до класу
2. Методом класу

Оголошення зовнішньої операторної функції

тип_результату operator оператор (аргументи: тип та назва)

{

//програмний код

}

Оператори:

+, -, *, /, %, =, +=, -=, *=, /=, %=, ++, --, !, [], ==, != і т.д.

Перевантаження операторів

1. При перевантаженні бінарних операторів зовнішніми операторними функціями останні мають два аргументи – перший та другий операнди.

2. При перевантаженні унарних операторів зовнішніми функціями останні мають один аргумент – операнд. Виключення – оператори інкременту та декременту (можуть мати два аргументи при перевантаженні постфіксної форми).

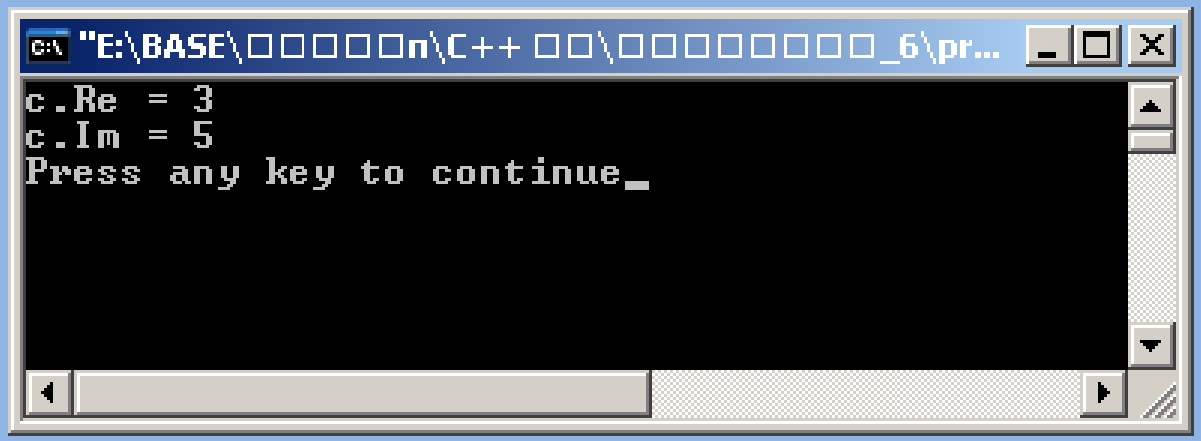
3. При перевантаженні бінарних операторів методами класу останні мають один аргумент – другий операнд. Першим операндом є об’єкт, з якого викорикається метод.

4. При перевантаженні унарних операторів методами класу останні не мають аргументів (операндом є об’єкт, з якого викликається метод). Виключення – оператори інкременту та декременту (можуть мати один аргумент при перевантаженні постфіксної форми).

Приклад 6.1. Перевантаження оператора додавання зовнішньою функцією

```
class MComp{
public:
double Re, Im;};
MComp operator+(MComp x, MComp y){
MComp z;
z.Re=x.Re+y.Re;
z.Im=x.Im+y.Im;
return z;}
int main(){
MComp a,b,c;
a.Re=1;
a.Im=2;
b.Re=2;
b.Im=3;
c=a+b;
cout<<"c.Re = "<<c.Re<<"\n";
cout<<"c.Im = "<<c.Im<<"\n";
return 0;}
```

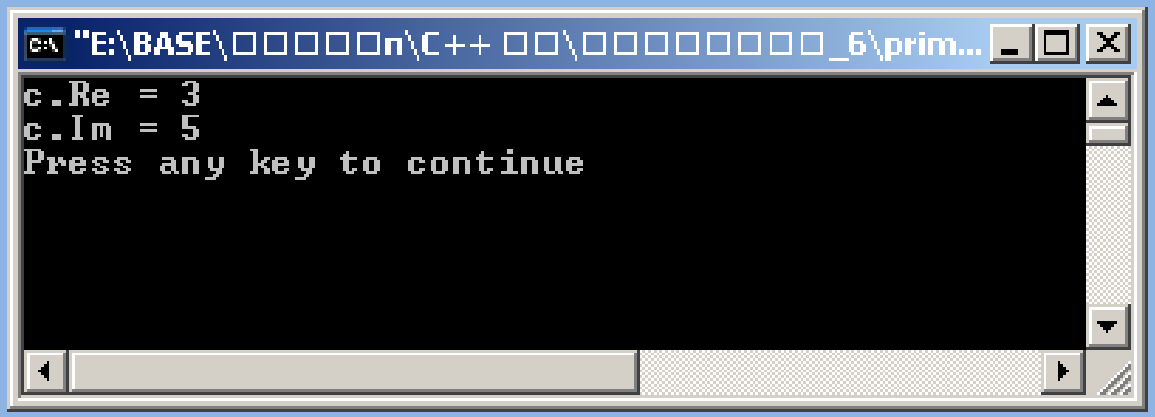
Результат



Приклад 6.2. Перевантаження оператора додавання методом класу

```
class MComp{
public:
double Re, Im;
MComp operator+(MComp y){
MComp z;
z.Re=Re+y.Re;
z.Im=Im+y.Im;
return z;}
};
int main(){
MComp a,b,c;
a.Re=1; a.Im=2;
b.Re=2; b.Im=3;
c=a+b;
cout<<"c.Re = "<<c.Re<<"\n";
cout<<"c.Im = "<<c.Im<<"\n";
return 0;}
```

Результат



Приклад 6.3. Перевантаження операторної функції

```
class MComp{
public:
double Re, Im;};
```

визначення класу

```
MComp operator+(MComp x, MComp y){
MComp z;
z.Re=x.Re+y.Re; z.Im=x.Im+y.Im;
return z;}
```

об’єкт + об’єкт

```
MComp operator+(double x, MComp y){
MComp z;
z.Re=x+y.Re; z.Im=y.Im;
return z;}
```

число + об’єкт

```
MComp operator+(MComp y, double x){
MComp z;
z.Re=x+y.Re; z.Im=y.Im;
return z;}
```

об’єкт + число

Приклад 6.4. Перевантаження операторів інкременту та декременту

```
class MComp{
public:
double Re, Im;
MComp operator--(){
Re--;
return *this;}
MComp operator--(int unused){
Im--;
return *this;}
};
```

```
MComp operator++(MComp &x){
x.Re++;
return x;}
MComp operator++(MComp &x, int unused){
x.Im++;
return x;}
```

визначення класу

Оператор декременту:
префіксна форма
постфіксна форма

МЕТОД КЛАСУ
ЗОВНІШНЯ ФУНКЦІЯ

Оператор інкременту:
префіксна форма
постфіксна форма

Приклад 6.5. Перевантаження оператора присвоєння

```
class MComp{
public:
double Re;
double Im;
MComp operator=(MComp x){
Re=x.Re+1;
Im=x.Im-1;
return *this;}
};
int main(){
MComp a,b;
a.Re=1;
a.Im=2;
b=a;
cout<<"b.Re = "<<b.Re<<"\n";
cout<<"b.Im = "<<b.Im<<"\n";
return 0;}
```

Результат

b=a >>> b≠a

Приклад 6.6. Індексування об’єктів

```
const int n=10;

class RealNums{
public:
int p[n];
```

```
RealNums(){
int k;
for(k=0;k<n;k++)
p[k]=rand()%n;
}
```

```
int &operator[](int i){
return p[i%n];
}
```

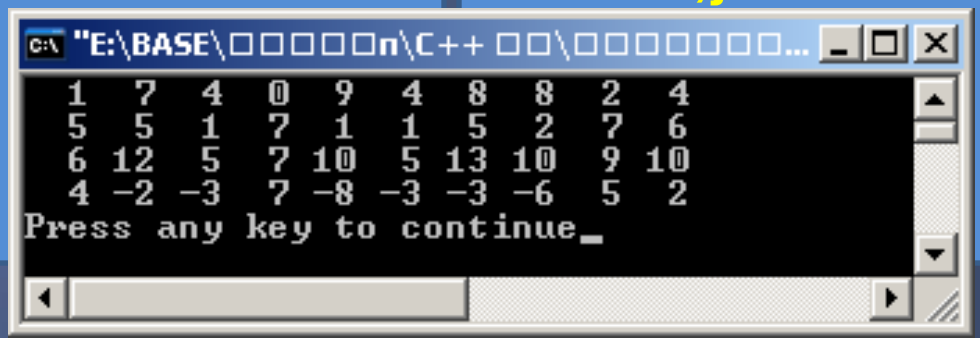
```
RealNums operator+(RealNums obj){
int i;
RealNums tmp;
for(i=0;i<n;i++)
tmp[i]=p[i]+obj[i];
return tmp;}
```

```
void show(){
int i;
for(i=0;i<n;i++)
printf("%3d",p[i]);
cout<<endl;}
};
```

```
int main(){
RealNums a,b;
a.show();
b.show();
(a+b).show();
for(int i=0;i<n;i++)
a[i]=b[i]-a[i];
a.show();
return 0;}
```

Зверніть увагу!!!

Зверніть увагу!!!



РЕЗЮМЕ

1. Операторна функція – функція, формальною назвою якої є перевантажуваний оператор. Операторна функція можу бути як зовнішньою, так і методом класу.

2. Аргументами операторної функції є операнди відповідного виразу. У випадку зовнішньої операторної функції для бінарних операторів аргументи два, для унарних операторів - один. Якщо використовується метод класу, бінарним операторам відповідає один аргумент, унарним – жодного. Унарні оператори інкременту та декременту в цьому відношенні мають особливості.

3. За умовчанням для операторів інкременту та декременту перевантажується префіксна форма. Для перевантаження постфіксної форми використовують додатковий числовий аргумент.

4. Операторні функції можуть перевантажуватись.