## 6. Предефиниране на операции

В **C++** всяка съществуваща унарна или бинарна операция, в която участва поне един обект от някакъв клас, може да бъде предефинирана от програмиста. Това дава възможност класовете да бъдат интерпретирани, като нови типове данни, които могат да се използват по начин аналогичен на базовите типове. Така например, ако бъде дефиниран клас вектор и **v1** и **v2** са обекти от този клас, то е възможно използването на изрази като: **v1+v2**, **v1-v2**, **v1*v2**, **v1/v2...**
Предефинирането на операции се осъществява чрез *операторни функции*. Операторната функция е член-функция или приятелска функция на класа, за който е дефинирана.

**I. Предефиниране чрез член-функции.**
Общата форма на една член-функция оператор е следната:

```
тип-резултат име-на-клас::operator#(списък аргументи)
{
 //изпълнявана операция
}
```

Типът на връщания резултат най-често е същият като типа на класа, за който е дефиниран операторът. Операторът, който се предефинира, замества # в конструкцията.

### 1.1. Предефиниране на бинарни оператори.

Когато една операторна член-функция предефинира бинарен оператор, функцията ще приема само един параметър. Този параметър ще получава обекта, който е от дясната страна на оператора. Обектът от лявата страна е този, който генерира обръщението към операторната функция и точно той се предава неявно на функцията посредством указателя **this**.

**Задача 1.** Следната програма предефинира операторите **+**, **-** и **=** за класа **coord**. Този клас поддържа **X**, **Y** координатна система.

```
#include<iostream.h>

class coord {
       int x, y;  //кординатни стойности
    public:
        coord() {x=0; y=0;}
        coord(int i, int j) { x=i; y=j; }
        void get_xy(int &i, int &j) { i=x; j=y; }
        coord operator+(coord ob2);
        coord operator-(coord ob2);
        coord operator=(coord ob2);
 };
coord coord::operator+(coord ob2) //Предефинира + за класа coord.
{
  coord temp;

  temp.x = x+ob2.x;
  temp.y = y+ob2.y;

  return temp;
}
coord coord::operator-(coord ob2) //Предефинира - за класа coord.
{
  coord temp;

  temp.x = x-ob2.x;
  temp.y = y-ob2.y;

  return temp;
}
coord coord::operator=(coord ob2) //Предефинира = за класа coord.
```

```
{
  x = ob2.x;
  y = ob2.y;

  return *this;
}
int main()
{ coord o1(10,10), o2(5,3), o3;
  int x, y;

  o3 = o1 + o2; //добавя два обекта – извиква се operator+()
  o3.get_xy(x, y);
  cout << "(o1+o2) X:" << x << ", Y:" << y << "\n";

  o3 = o1 - o2; //изважда два обекта – извиква се operator-()
  o3.get_xy(x, y);
  cout << "(o1-o2) X:" << x << ", Y:" << y << "\n";

  o3 = o1; //присвоява обект - извиква се operator=()
  o3.get_xy(x, y);
  cout << "(o3=o1) X:" << x << ", Y:" << y << "\n";
  return 0;
}

Резултати:
(o1+o2) x:15, y:13
(o1-o2) x:5, y:7
(o3=o1) x:10, y:10
```

Функцията **operator+()** връща обект от тип **coord**, който съдържа сумата от **X** координатите в **x** и **Y** координатите в **y** за двата операнда. Временният обект **temp** в тялото на **operator+()** съдържа резултата и се връща като резултат. Следната конструкция

```
o3 = o1 + o2;
```

е валидна, защото резултатът от **o1+o2** е обект от тип **coord** и той може да бъде присвоен на **o3**.
При функцията **operator-()**, за разлика от **operator+()**, е важно да се извади операнда отдясно от операнда отляво. Тъй като левият операнд генерира обръщението към **operator-()**, то изваждането трябва да се извърши в следният ред:

```
x - ob2.x;
```

Операторната функция **operator=()** връща ***this**, т.е. тя връща обекта, на който се присвоява стойност. Връщайки като резултат ***this**, предефинираният оператор за присвояване позволява на обекти от тип **coord** да бъдат използвани в поредица от присвоявания:

```
o3 = o2 = o1;
```

В горния пример предаването на параметрите на операторните член-функции се прави по стойност. Може да се използва псевдоним като параметър към една операторна член-функция. Напр. ето един коректен начин да се предефинира оператора **+** за класа **coord**:

```
coord coord::operator+(coord &ob2)      //предефиниране на + за класа coord чрез
псевдоним
{
  coord temp;
  temp.x = x + ob2.x;
  temp.y = y + ob2.y;
  return temp;
}
```

**1.2. Предефиниране на унарни оператори.**
Когато се предефинира даден унарен оператор посредством операторна член-функция, функцията не приема параметри. Тъй като операндът е само един, то той е този, който генерира обръщението към операторната функция.

**Задача 2.** Следната програма предефинира оператора **--** за класа **coord** чрез префикс и постфикс форми.

```
#include<iostream.h>

class coord {
       int x, y;  //кординатни стойности
    public:
        coord() { x=0; y=0; }
        coord(int i, int j) { x=i; y=j; }
        void get_xy(int &i, int &j) { i=x; j=y; }
        coord operator--(); //prefix
        coord operator--(int notused);  //postfix
 };
coord coord::operator--()   //Предефиниране на префиксен -- за класа coord.
{
  x--;
  y--;
  return *this;
}
coord coord::operator--(int notused)     //Предефиниране на постфиксен -- за класа
coord.
{
  x--;
  y--;
   return *this;
 }
int main()
{
  coord o1(10, 10);
  int x, y;

  --o1;   //декрементиране на обект
  o1.get_xy(x, y);
  cout << "(--o1) X:" << x << ", Y:" << y << "\n";

  o1--;   //декрементиране на обект
  o1.get_xy(x, y);
  cout << "(o1--) X:" << x << ", Y:" << y << "\n";
  return 0;
}
```

В примера се създават два варианта на функцията **operator--()**. Ако **--** се намира пред операнда, то се извиква функцията **operator--()**. Обаче ако **--** е след операнда, то се извиква функцията **operator--(int notused)**. В този случай на **notused** винаги ще се предава стойност 0.

**II. Предефиниране чрез приятелски функции.**

Тъй като една приятелска функция не получава **this** указател, в случай на бинарен оператор това означава, че се предават директно и двата операнда. При унарните оператори се предава единствен операнд.

*Не може да се използва приятелска функция, за да се предефинира оператора за присвояване. Оператора за присвояване може да бъде предефиниран само от операторна член-функция.*

**Задача 3.** Следната програма предефинира оператора **++** чрез приятелска функция. Ако се използва приятелска операторна функция за предефиниране на унарните оператори **++** и **-**, трябва да се предава операнда на функцията като **псевдоним**. Това е необходимо, защото приятелските функции нямат **this** указател.

Ако се предаде операнда към приятелската функция като псевдоним, то промените, извършени в тялото на приятелската функция, ще изменят обекта, генерирал извикването.

```
#include<iostream.h>

class coord {
       int x, y;  //кординатни стойности
    public:
        coord() { x=0; y=0; }
        coord(int i, int j) { x=i; y=j; }
        void get_xy(int &i, int &j) { i=x; j=y; }
        friend coord operator++(coord &ob);
 };
coord operator++(coord &ob) //Предефинира ++ с помощта на приятелска функция
{
  ob.x++;
  ob.y++;
  return ob;    //връща обекта, който е извършил извикването
}
int main()
{
  coord o1(10, 10);
  int x, y;

  ++o1;   //o1 се подава с псевдоним
  o1.get_xy(x, y);
  cout << "(++o1) X:" << x << ", Y:" << y << "\n";

  return 0;
}
```

**Задачи за самостоятелна работа:**

1. Изтеглете и разгледайте примера Ex8_06.cpp.
2. Реализирайте по подобен начин предефиниране на операторите:
   - <
   - = =
   - -
3. Демонстрирайте функционалността чрез създаване на тестова програма.

**Допълнителни задачи:**

1. Дефинирайте един клас RINT, който се държи подобно на int, с изключение на това, че са разрешени единствено операциите +(унарна и бинарна), -(унарна и бинарна), *, /.
2. Дефинирайте клас Vec4 като вектор от четири float. Дефинирайте operator[] за Vec4. Дефинирайте операторите +, -, *, / за комбинация от вектори и float.