## Создание функций преобразования типов

Иногда надо создавать класс таким образом, чтобы иметь возможность свободно использовать его в смешанных выражениях с данными других типов. Хотя перегруженные функции-операторы могут обеспечить смешанное использование типов, иногда достаточно простого преобразования типов. В таких случаях можно использовать функцию преобразования типа, преобразующую тип класса в другой тип, совместимый с остальными членами выражения. Общая форма функции преобразования типа имеет вид:

operator (тип)() {return значение;}

Здесь тип является целевым типом, к которому преобразуется класс, а значение представляет со­бой значение класса после преобразования. Функция преобразования должна быть членом клас­са, для которого она определяется.

Чтобы проиллюстрировать создание функции преобразования типа, используем класс three\_d еще раз. Предположим, что необходимо преобразовать объект типа three\_d в целое число таким  
образом, чтобы оно могло использоваться в целочисленном выражении. Далее надо иметь преоб­разование, в котором имеет место произведение трех размерностей. Для выполнения такого пре­образования можно определить функцию преобразования следующим образом:

operator int() { return х \* у \* z; }

Ниже приведена программа, иллюстрирующая работу этой функции преобразования:  
  
#include <iostream.h>

class three\_d {

int x, у, z; // трехмерные координаты

public:

three\_d(int a, int b, int c) {x=a; y=b, z=c; }

three\_d operator+(three\_d op2);

friend ostream &amp;operator<<(ostream &amp;stream, three\_d &amp;obj);

operator int() {return x\*y\*z;}

};

// вывод координат X, Y, Z (оператор вставки для three\_d)

ostream &amp;operator<< (ostream &amp;stream, three\_d &amp;obj)

{

stream << obj.x << ", ";

stream << obj.у << ", ";

stream << obj.z << "\n";

return stream; // возврат потока

}

three\_d three\_d::operator\* (three\_d op2)

{

three\_d temp(0, 0, 0);

temp.x = x+op2.x; // сложение целых

temp.у = y+op2.y; // + сохранил применительно к ним

temp.z = z+op2.z; // исходный смысл

return temp;

}

int main()

{

three\_d a(1, 2, 3), b(2, 3, 4);

cout << a << b;

cout << b+100; // вывод 124 из-за преобразования к int

cout << "\n";

cout << a+b; // вывод 3, 5, 7 - нет преобразования

return 0;

}

Эта программа выдаст следующий результат:  
  
1, 2, 3  
2, 3, 4  
124  
3, 5, 7

Как иллюстрирует данная программа, когда объект three\_d используется в целочисленном выра­жении, таком как cout<< b+100, используется функция преобразования объекта. В этом частном случае функция преобразования возвращает значение 24, которое затем прибавляется к 100. Од­нако в случае, когда преобразование не требуется, например, в выражении cout << a+b, функ­ция преобразования не вызывается.  
Надо иметь в виду, что можно создать различные функции преобразования, отвечающие раз­личным требованиям. Каждая из них будет применяться автоматически.