Перегружаемые операторы

Перегрузка операторов в программировании – это реализация в одной области видимости нескольких различных ***вариантов*** применения оператора, имеющих одно и то же имя, но различающихся типами параметров, к которым они применяются.

Ключевое `operator` слово объявляет функцию, указывающую, что означает `<символ_оператора>` при применении его к объекту. Это дает оператору более одного значения – «перегружает» его. Компилятор различает разные значения оператора, проверяя типы его операндов.

Синтаксис:

```
<тип> operator <символ_оператора> (список_параметров)
```

Перегрузить можно только те операторы, которые уже определены в C++. Создать новые операторы нельзя. Также нельзя изменить количество операндов, их ассоциативность, приоритет.

Перегруженные операторы реализуются в виде функции, например:

```
#include "stdafx.h"
#include <locale>
#include <iostream>


enum SERVER{STARTING, SHOOTDOWN, WORK, STOP};
enum CLIENT{CONNECT, SEND, RECV, DISCONNECT};
enum EVENT {WAIT,DO, ERROR};

EVENT operator>>(CLIENT c, SERVER s)
{
    EVENT rc = ERROR;
    if      ((c == CONNECT ||c == SEND ||c == RECV)&&(s == STARTING || s == STOP)) rc = WAIT;
    else if ((c == CONNECT ||c == SEND ||c == RECV)&&(s == WORK)) rc = DO;
    return rc;
};

int _tmain(int argc, _TCHAR* argv[])
{

   EVENT e1 = CONNECT>>STARTING;

   EVENT e2 = CONNECT>>WORK;

   EVENT e3 = SEND>>SHOOTDOWN;

   system("pause");
   return 0;
```

e1	WAIT (0)
e2	DO (1)
e3	ERROR (2)

```
#include <locale>
#include <iostream>

struct POINT  { int x;  int y; };
struct VECTOR { int x;  int y; };

VECTOR operator-(POINT p1, POINT p2)
{
  VECTOR rc = {p1.x-p2.x, p1.y-p2.y} ;
  return rc;
};
VECTOR operator+(VECTOR v1, VECTOR v2)
{
  VECTOR rc = {v1.x+v2.x, v1.y+v2.y} ;
  return rc;
};

int _tmain(int argc, _TCHAR* argv[])
{
    POINT p1 = {10, 20}, p2 = {15, -10};
    VECTOR v1, v2, v3;

    v1 = p1 - p2;      v1    {x=-5 y=30 }

    v2 = p2 - p1;      v2    {x=5 y=-30 }

    v3 = v1 + v2;      v3    {x=0 y=0 }

    system("pause");
    return 0;
}
```