# C++之运算符重载

1. 运算符重载的意义就是让程序员自己定义自己的运算符。
2. 格式：**operator**运算符（参数列表）

注：这个运算符必须是C++合法的运算符，不能任意虚构C++本来没有的运算符，例如：**operator@**，因为C++没有@运算符**。**

**3**依据《C++ Primer Plus 第6版 中文版》的例子：

头文件

|  |
| --- |
| #pragma once  class Time{  private:  int hours;  int minutes;  public:  Time();  Time(int h, int m = 0);  void AddMin(int m);  void AddHr(int h);  void ReSet(int h = 0, int m = 0);  Time Sum(const Time& t) const;  Time operator \*(const Time& t) const;  void Show() const;  }; |

cpp文件

|  |
| --- |
| // Test.cpp: 定义控制台应用程序的入口点。  //  #include "stdafx.h"  #include<iostream>  #include"demotime.h"  Time::Time()  {  hours = minutes = 0;  }  Time::Time(int h, int m)  {  hours = h;  minutes = m;  }  void Time::AddMin(int m)  {  minutes += m;  hours += minutes / 60;  minutes = minutes % 60;  }  void Time::AddHr(int h)  {  hours += h;  }  void Time::ReSet(int h, int m)  {  hours = h;  minutes = m;  }  Time Time::Sum(const Time & t)const  {  Time sum;  sum.minutes += minutes + t.minutes;  sum.hours += hours + t.hours + sum.minutes / 60;  sum.minutes %= 60;  return sum;  }  Time Time::operator+(const Time & t)const  {  Time sum;  sum.minutes += minutes + t.minutes;  sum.hours += hours + t.hours + sum.minutes / 60;  sum.minutes %= 60;  return sum;  }  void Time::Show()const  {  std::cout << "hours=" << hours << " minutes=" << minutes << std::endl;  }  int main()  {  using std::cout;  using std::endl;  Time planing;  Time coding(2, 40);  Time fixing(5, 55);  Time total;  cout << "planing time:";  planing.Show();  cout << endl;  cout << "coding time:";  coding.Show();  cout << endl;  cout << "fixing time:";  fixing.Show();  cout << endl;  total = coding.Sum(fixing);  cout << "coding.Sum(fixing) time:";  total.Show();  cout << endl;  total = coding + fixing;  cout << "coding+fixing time:";  total.Show();  cout << endl;  getchar();  return 0;  } |

该例子中是对+运算符的重载，如下：

Time Time::operator+(const Time & t)const

{

Time sum;

sum.minutes += minutes + t.minutes;

sum.hours += hours + t.hours + sum.minutes / 60;

sum.minutes %= 60;

return sum;

}

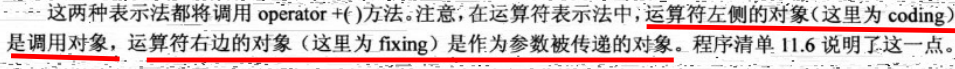
我们都知道，C++里面变量的+，比如：

Int a = 10;

Int b = 20;

Int c = a+b;

C结果是30，这是不言而喻的，但是C++不支持自定义变量的+，也就是说程序员自己定义的变量类型的+,C++是让程序员自己去实现。而类可以看做是程序员自定义的一种变量类型（类似结构体），这种类型的“变量”的+运算，需要程序员自己去实现，怎么实现？定义一个函数去实现呗！而C++已经规定好了固定的函数格式，为什么要固定格式？是因为这种格式，能在代码里的写法和普通变量一样的写法，比如：两个对象也可以写成**total = coding + fixing;**



其实上面的例子用这种方法调用：total = coding.Sum(fixing);这是函数调用写法，和total = coding + fixing;是等价的，后者也同样会调用operator+(const Time & t)const这个成员函数。

另外，有人在想operator后面的+是不是可以换成其他符号，可以是可以，但是有限制：

