C++ Programming

Chapter 10 Operator Overloading

Part 1/2

运算符重载

Zheng Guibin
（郑贵滨）

哈爾濱工業大學
Harbin Institute of Technology

10.1 The need for operator overloading

运算符重载Operator Overloading

- C++ 的内置算术运算符（+,*,/等）和关系运算符（>,<,==,!=）可用于内置数据类型（int,float等）。
- 并非所有的内置运算符都能与每一种数据类型配合使用。
- 例如，字符串不能进行乘法操作，%只适用于整型数。
- 运算符+可作用于字符串，表示字符串连接。
- 运算符+作用于数值数据类型和作用于字符串，其含义是不同的。

10.1 The need for operator overloading

- 当定义一个新的类时，可以重新定义或者重载已经存在的运算符。
- 运算符重载—— 运算符函数
  - 在类中，使用运算符定义的特殊成员函数，有特殊的用途。
  - 为对象的行为提供了方便的符号。

Program Example P10A

```
1  // Program Example P10A
2  // Demonstration of an overloaded + operator.
3  #include <iostream>
4  using namespace std ;
5
6  class time24 // A simple 24-hour time class.
7  {
8       public:
9          time24( int h = 0, int m = 0, int s = 0 ) ;
10         void set_time( int h, int m, int s ) ;
11         void get_time( int& h, int& m, int& s ) const ;
12         time24 operator+( int secs ) const ;
13      private:
14         int hours ; // 0 to 23
15         int minutes ; // 0 to 59
16         Int seconds ; // 0 to 59
17 };
```

- line 12：the prototype for the overloaded + operator (重载运算符+的函数原型)

10.2 Overloading the addition operator +

```
18  // Constructor.
19  time24::time24( int h, int m, int s ) : hours( h ), minutes( m ), seconds( s ) {}
20
21  // Mutator function.
22  void time24::set_time( int h, int m, int s )
23  {
24        hours = h ; minutes = m ; seconds = s ;
25  }
26
27  // Inspector function.
28  void time24::get_time( int& h, int& m, int& s ) const
29  {
30        h = hours ; m = minutes ; s = seconds ;
31  }
```

10.2 Overloading the addition operator +

```
32 // Overloaded + operator.
33 time24  time24::operator+( int secs ) const
34 {
35       // Add secs to class member seconds and calculate the new
time.
36        time24 temp ;
37        temp.seconds = seconds + secs ;
38        temp.minutes = minutes + temp.seconds / 60 ;
39        temp.seconds %= 60 ;
40        temp.hours = hours + temp.minutes / 60 ;
41       temp.minutes %= 60 ;
42       temp.hours %= 24 ;
43       return temp ; // Return the new time.
44 }
```

10.2 Overloading the addition operator +

```
45 main()
46 {
47        int h, m, s ;
48         time24 t1( 23, 59, 57 ) ; // t1 represents 23:59:57
49         time24 t2 ;
50         t2 = t1 + 4 ; // t2 should now be 0:0:1
51         t2.get_time ( h, m, s ) ;
52        cout << "Time t2 is " << h << ":" << m << ":" << s << endl ;
53 }
```

重载的运算符函数，可以像其他函数那样被调用， t2 = t1 + 4; 可以写成:

t2 = t1.operator+(4) ;

10.2 Overloading the addition operator +

Program Example P10A+B+C

```
1  // Program Example P10C
2  // Demonstration of an overloaded + operator.
3  #include <iostream>
4  using namespace std ;
5  class time24 // A simple 24-hour time class.
6  {
7       public:
8          time24( int h = 0, int m = 0, int s = 0 ) ;
9          void set_time( int h, int m, int s ) ;
10         void get_time( int& h, int& m, int& s ) const ;
11         time24 operator+( int secs ) const ;//重载运算符+的函数原型
12         time24 operator+( const time24& t ) const ;
13         private:
14         int hours ; // 0 to 23
15         int minutes ; // 0 to 59
16         Int seconds ; // 0 to 59
17 }
```

10.2 Overloading the addition operator +

```
19  // Constructor.
20 time24::time24(int h,int m,int s):hours(h), minutes(m), seconds(s){}
21  // Mutator function.
22  void time24::set_time( int h, int m, int s )
23  {
24        hours = h ; minutes = m ; seconds = s ;
25  }
26
27  // Inspector function.
28  void time24::get_time( int& h, int& m, int& s ) const
29  {
30        h = hours ; m = minutes ; s = seconds ;
31  }
```

10.2 Overloading the addition operator +

```
32 // Overloaded + operator.
33 time24 time24::operator+( int secs ) const
34 {//重载+运算符：将time24对象和一个整型的秒数相加并返回结果
35       // Add secs to class member seconds and calculate the new time.
36        time24 temp ;
37        temp.seconds = seconds + secs ;
38        temp.minutes = minutes + temp.seconds / 60 ;
39        temp.seconds %= 60 ;
40        temp.hours = hours + temp.minutes / 60 ;
41       temp.minutes %= 60 ;
42       temp.hours %= 24 ;
43       return temp ; // Return the new time.
44 }
```

```
45 // Overloaded + operator.
/*重载运算符+,将一个time24对象的时间值和time24对象相加,
并返回结果*/
46 time24 time24::operator+( const time24& t ) const
47 {
48        // Add total seconds in t to seconds and calculate the new time.
49        time24 temp ;
50        int secs = t.hours * 3600 + t.minutes * 60 + t.seconds ;
51       temp.seconds = seconds + secs ;
52       temp.minutes = minutes + temp.seconds / 60 ;
53       temp.seconds %= 60 ;
54       temp.hours = hours + temp.minutes / 60 ;
55       temp.minutes %= 60 ;
56       temp.hours %= 24 ;
57       return temp ; // Return the new time.
58 }
```

10.2 Overloading the addition operator +

```
// Non-member overloaded + operator.
59 time24 operator+( int secs, const time24& t )
60 {
61       // Add secs to t to calculate the new time.
62        time24 temp ;
63       temp = t + secs ; // Uses the member function operator+( int ).
64       return temp ; // Return the new time.
65 }
```

10.2 Overloading the addition operator +

//Program Example P10B

```
66 main()
67 {
68        int h, m, s ;
69        time24 start_time( 23, 0, 0 ) ;
70         time24 elapsed_time( 1, 2, 3 ) ;
71         time24 finish_time ;
72        finish_time = start_time + elapsed_time ;
73         finish_time.get_time( h, m, s ) ;
74        cout << "Finish Time is "
75         << h << ":" << m << ":" << s << endl ;
```

finish_time = start_time.operator+(elapsed_time) ;

10.2 Overloading the addition operator +

Program Example P10C

```
76        time24 t1( 23, 59, 57 ) ; // t1 represents 23:59:57
77        time24 t2 ;
78        t2 = t1 + 4 ;
79        t2.get_time ( h, m, s ) ;
80        cout << "Time t2 is " << h << ":" << m << ":" << s << endl ;
81        t2 = 4 + t1 ;
82        t2.get_time ( h, m, s ) ;
83        cout << "Time t2 is " << h << ":" << m << ":" << s << endl ;
84 }
```

t2 = t1.operator+(4) ;

t2 = operator+(4, t1)

t2 = operator+(t1, 4) ;//?

10.3 Rules of operator overloading
运算符重载的规则

- 除了下面列举的五个运算符以外，附录B中的所有C++运算符都可以重载

  . .* :: ?: sizeof

- 运算符重载的规则
  - 不能发明新的运算符。例如，不可以重载<>来表示“不等于”或者重载**来表示“幂”
  - 运算符重载后的操作数个数与原运算符的操作数个数必须相同。例如，重载运算符==必须有两个操作数，重载运算符++应只有一个操作数。
  - 运算符重载后仍保持其原有的优先级。例如，无论是否对*进行重载，它的优先级都将高于+和-
  - 重载运算符时不能使用默认实参。
  - 用于内置数据类型（如int、float等）时，运算符的含义不能被重新定义。

```
1 // Program Example P10D
2 // Program to demonstrate overloading the prefix ++ operator.
3 #include <iostream>
4 using namespace std ;
5
6 class time24 // A simple 24-hour time class.
7 {
8 public:
9     time24( int h = 0, int m = 0, int s = 0 ) ;
10    void set_time( int h, int m, int s ) ;
11    void get_time( int& h, int& m, int& s ) const ;
12    time24 operator+( int secs ) const ;
13    time24 operator+( const time24& t ) const ;
14    time24 operator++() ;
15 private:
16    int hours ; // 0 to 23
17    int minutes ; // 0 to 59
18    int seconds ; // 0 to 59
19 } ;
```

重载前缀自增运算符++的原型

10.4 Overloading ++

内置指针：this
在类的所有成员函数中都可以使用this指针，this指针指向调用成员函数的对象。

```
66 //Overloaded prefix ++ operator.
67 time24 time24::operator++()
68 {
69   // Add 1 second to the calling object's seconds.
70   *this = *this + 1 ; // Uses member function operator+( int ).
71   return *this ; // Return updated time of calling object.
```

```
83 main()
84 {
85   int h, m, s ;
86   time24 t1( 23, 59, 57 ) ;
87
88   ++t1 ;
89   t1.get_time ( h, m, s ) ;
90   cout << "Time t1 is " << h << ":" << m << ":" << s << endl ;
91 }
```

当程序88行调用operator++()时，this指针中包含的是对象t1的地址。

10.4.1 Overloading prefix and postfix forms of ++
重载前缀和后缀形式的++

- 对于自增运算符++，需要对其前缀和后缀形式分别进行重载。
  - t1,t2是time24的对象，下面两种表述将产生不同的结果:

```
t2 = ++t1 ; // Use of prefix ++.
t2 = t1++ ; // Use of postfix ++.
```

  - 第一种表述t1先增加然后将值赋给t2。
  - 第二种表述先将t1的值赋给t2，然后t1再增加。
- 自减运算符--的情况类似

10.4.1 Overloading prefix and postfix forms of ++

- C++编译器如何区分重载的后置++ 前置++
  - 若d为自定义类 MyClass的对象

自增表达式	编译器自动生成的成员函数调用	对应的函数原型
++d	d.operator++()	MyClass &operator++();
d++	d.operator++(0)	MyClass operator++(int);

自增表达式	编译器自动生成的非成员函数调用	对应的函数原型
++d	operator++(d)	friend MyClass &operator++(MyClass &);
d++	operator++(d,0)	friend MyClass operator++(MyClass &, int);

```
1 // Program Example P10E
2 // Program to demonstrate overloading prefix and postfix ++.
6 class time24 // A simple 24-hour time class.
7 {
8 public:
9     time24( int h = 0, int m = 0, int s = 0 ) ;
10    void set_time( int h, int m, int s ) ;
11    void get_time( int& h, int& m, int& s ) const ;
12    time24 operator+( int secs ) const ;
13    time24 operator+( const time24& t ) const ;
14    time24 operator++() ; // prefix.
15    time24 operator++( int ) ; // postfix.
16 private:
17    int hours ; // 0 to 23
18    int minutes ; // 0 to 59
19    int seconds ; // 0 to 59
20 } ;
```

后缀版本使用一个“虚拟”的整型参数

10.4.1 Overloading prefix and postfix forms of ++

```
76 time24 time24::operator++( int )
77 {
78   // Save calling object before incrementing seconds.
79   time24 temp ;
80   temp = *this ;
81   *this = *this + 1 ; // Uses operator+( int ),
82                       // could also use ++(*this).
83   return temp ; // Return the saved calling object.
84 }
```

10.4.1 Overloading prefix and postfix forms of ++

```
95 main()
96 {
97   int h, m, s ;
98   time24 t1( 23, 59, 57 ) ;
99   time24 t2 ;
100
101  t2 = t1++ ; // Test postfix ++
102  t1.get_time ( h, m, s ) ;
103  cout << "Using postfix ++: " << "time t1 is: "
104       << h << ":" << m << ":" << s ;
105  t2.get_time( h, m, s ) ;
106  cout << ", time t2 is: "
107    << h << ":" << m << ":" << s << endl ;
```

```
108
109  t1.set_time (23, 59, 57 ) ; // Reset the time.
110
111  t2 = ++t1 ; // Test prefix ++
112  t1.get_time ( h, m, s ) ;
113  cout << "Using prefix ++: " << "time t1 is: "
114       << h << ":" << m << ":" << s ;
115  t2.get_time ( h, m, s ) ;
116  cout << ", time t2 is: "
117       << h << ":" << m << ":" << s << endl ;
118 }
```

10.4.2 Improving the prefix ++ operator member function
改进前缀++运算符成员函数

```
68 time24 time24::operator++()
69 {
70    // Add 1 second to the calling object
71    *this = *this + 1 ; // Uses member function operator+( int ).
72    return *this ;
73 }
```

- 程序的第72行的return语句返回了对象的副本。
- 如果返回的是该对象的引用，则效率会更高（尤其对于大的对象）。
- 为了返回一个对象的引用，68行的**函数头**改写为:
  time24**&** time24::operator++()
- **函数原型**改为:
  time24**&** operator++() ;

10.4.2 Improving the prefix ++ operator member function

```
76 time24 time24::operator++( int )
77 {
78   // Save calling object before incrementing seconds.
79   time24 temp ;
80   temp = *this ;
81   *this = *this + 1 ; // Uses operator+( int ),
82   // could also use ++(*this).
83   return temp ; // Return the saved calling object.
84 }
```

- 对于重载后缀运算符++成员函数，不能做这样的改写。
- 因为:在第83行中返回的对象是一个局部对象temp，当其所在函数调用结束后，由于超出了该局部对象的作用域而成为未定义的对象。
- 返回一个未定义的对象的引用将产生错误，因此只能返回temp对象的副本。