C++ 继承 C++ 多态

C++ 重载运算符和重载函数

C++ 允许在同一作用域中的某个**函数**和**运算符**指定多个定义，分别称为**函数重载**和**运算符重载**。

重载声明是指一个与之前已经在该作用域内声明过的函数或方法具有相同名称的声明，但是它们的参数列表和定义（实现）不相同。

当您调用一个**重载函数**或**重载运算符**时，编译器通过把您所使用的参数类型与定义中的参数类型进行比较，决定选用最合适的定义。选择最合适的重载函数或重载运算符的过程，称为**重载决策**。

C++ 中的函数重载

在同一个作用域内，可以声明几个功能类似的同名函数，但是这些同名函数的形式参数（指参数的个数、类型或者顺序）必须不同。您不能仅通过返回类型的不同来重载函数。

下面的实例中，同名函数 **print()** 被用于输出不同的数据类型：

实例

```
#include <iostream>
using namespace std;
class printData
{
public:
void print(int i) {
cout << "整数为: " << i << endl;
}
void print(double f) {
cout << "浮点数为: " << f << endl;
}
void print(char c[]) {
cout << "字符串为: " << c << endl;
}
};
int main(void)
{
printData pd;
// 输出整数
pd.print(5);
// 输出浮点数
pd.print(500.263);
// 输出字符串
char c[] = "Hello C++";
pd.print(c);
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
整数为: 5
浮点数为: 500.263
字符串为: Hello C++
```

C++ 中的运算符重载

您可以重定义或重载大部分 C++ 内置的运算符。这样，您就能使用自定义类型的运算符。

重载的运算符是带有特殊名称的函数，函数名是由关键字 operator 和其后要重载的运算符符号构成的。与其他函数一样，重载运算符有一个返回类型和一个参数列表。

```
Box operator+(const Box&);
```

声明加法运算符用于把两个 Box 对象相加，返回最终的 Box 对象。大多数的重载运算符可被定义为普通的非成员函数或者被定义为类成员函数。如果我们定义上面的函数为类的非成员函数，那么我们需要为每次操作传递两个参数，如下所示：

```
Box operator+(const Box&, const Box&);
```

下面的实例使用成员函数演示了运算符重载的概念。在这里，对象作为参数进行传递，对象的属性使用 **this** 运算符进行访问，如下所示：

实例

```
#include <iostream>
using namespace std;
class Box
{
public:
double getVolume(void)
{
return length * breadth * height;
}
void setLength( double len )
{
length = len;
}
void setBreadth( double bre )
{
breadth = bre;
}
void setHeight( double hei )
{
height = hei;
}
// 重载 + 运算符，用于把两个 Box 对象相加
Box operator+(const Box& b)
{
Box box;
box.length = this->length + b.length;
```

```
box.breadth = this->breadth + b.breadth;
box.height = this->height + b.height;
return box;
}
private:
double length; // 长度
double breadth; // 宽度
double height; // 高度
};
// 程序的主函数
int main( )
{
Box Box1; // 声明 Box1，类型为 Box
Box Box2; // 声明 Box2，类型为 Box
Box Box3; // 声明 Box3，类型为 Box
double volume = 0.0; // 把体积存储在该变量中
// Box1 详述
Box1.setLength(6.0);
Box1.setBreadth(7.0);
Box1.setHeight(5.0);
// Box2 详述
Box2.setLength(12.0);
Box2.setBreadth(13.0);
Box2.setHeight(10.0);
// Box1 的体积
volume = Box1.getVolume();
cout << "Volume of Box1 : " << volume <<endl;
// Box2 的体积
volume = Box2.getVolume();
cout << "Volume of Box2 : " << volume <<endl;
// 把两个对象相加，得到 Box3
Box3 = Box1 + Box2;
// Box3 的体积
volume = Box3.getVolume();
cout << "Volume of Box3 : " << volume <<endl;
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Volume of Box1 : 210
Volume of Box2 : 1560
Volume of Box3 : 5400
```

可重载运算符/不可重载运算符

下面是可重载的运算符列表：

双目算术运算符	+ (加)，-(减)，*(乘)，/(除)，% (取模)
关系运算符	==(等于)，!= (不等于)，< (小于)，> (大于>，<=(小于等于)，>=(大于等于)

逻辑运算符	||(逻辑或)，&&(逻辑与)，!(逻辑非)
单目运算符	+ (正)，-(负)，*(指针)，&(取地址)
自增自减运算符	++(自增)，--(自减)
位运算符		(按位或)，& (按位与)，~(按位取反)，^(按位异或),，<< (左移)，>>(右移)
赋值运算符	=, +=, -=, *=, /= , % = , &=, |=, ^=, <<=, >>=
空间申请与释放	new, delete, new[ ] , delete[]
其他运算符	`()`(函数调用)，`->`(成员访问)，`,`(逗号)，`[]`(下标)

下面是不可重载的运算符列表：

- `.`：成员访问运算符
- `.*`, `->*`：成员指针访问运算符
- `::`：域运算符
- `sizeof`：长度运算符
- `?:`：条件运算符
- `#`： 预处理符号

运算符重载实例

下面提供了各种运算符重载的实例，帮助您更好地理解重载的概念。

序号	运算符和实例
1	一元运算符重载
2	二元运算符重载
3	关系运算符重载
4	输入/输出运算符重载
5	++ 和 -- 运算符重载
6	赋值运算符重载
7	函数调用运算符 () 重载
8	下标运算符 [] 重载
9	类成员访问运算符 -> 重载

C++ 继承 C++ 多态

1 篇笔记

写笔记

值得注意的是:

- 1、运算重载符不可以改变语法结构。
- 2、运算重载符不可以改变操作数的个数。
- 3、运算重载符不可以改变优先级。
- 4、运算重载符不可以改变结合性。

oin625 10个月前 (06-01)