类和对象

C++在 C 语言的基础上增加了面向对象编程，C++支持面向对象程序设计。

类是 C++的核心特性，通常被称为用户定义的类型。

本章主要内容:

- 什么是类和对象
- 例：类的成员变量
- 类的成员函数
- 例：类的成员函数
- 类的私有成员
- 构造函数和析构函数

什么是类和对象

下面的代码定义一个类 Box，以及类 Box 的对象。

```
class Box
{
    public:
        double length; //矩形的长度
        double width; //矩形的宽度
        double getArea(); //计算矩形的面积
};
Box box1;
Box box2[10];
```

类（class）是抽象的数据类型，它包含了数据的表示和用于处理数据的方法。

对象是类的具体实例。

类中的数据和方法（函数）称为类的成员。

数据定义了类的对象包括了什么。

函数定义了可以在这个对象上执行哪些操作。

声明类的对象，和声明基本类型的变量一样。

下面代码声明了类 Box 的一个对象 box1:

```
Box box1;
```

下面代码声明了包含 10 个类 Box 对象的数组 box2:

```
Box box2[10];
```

每个对象都有各自的数据成员。

关键字 public 确定了类成员的访问属性。public 成员在类的外部是可访问的。

也可以指定类的成员为 private 或 protected。

例：类的成员变量

类的对象的公共（public）数据成员可以使用直接成员访问运算符“.”来访问。

【参考程序】

```
#include<iostream>
using namespace std;
class Box
{
    public:
        double length;
        double width;
};
int main()
{
    Box box;
    cin >> box.length >> box.width;
    cout << box.length * box.width << endl;
    return 0;
}
```

【输入样例】

```
5
6
```

【输出样例】

```
30
```

类的成员函数

类的成员函数是指把定义写在类定义内部的函数，就像类定义中的其他变量一样。

类成员函数是类的一个成员，类的任意对象都可以使用类成员函数，类成员函数可以访问对象中的所有成员。

```
1. class Box
2. {
3.     public:
4.         double length; //矩形的长度
5.         double width; //矩形的宽度
6.         double getArea(); //计算矩形的面积
7.         double getPerimeter(); //计算矩形的周长
8. };
```

现在我们要使用成员函数来访问类的成员，而不是直接访问这些类的成员。

可以在类定义内部定义成员函数，比如 getPerimeter()。

也可以使用范围解析运算符::，在类定义外部定义成员函数，比如 getArea()。

```
1. class Box
2. {
3.     public:
4.         double length;
5.         double width;
6.         double getArea();
7.         double getPerimeter()
8.         {
9.             return 2*(length+width);
10.        }
11. };
12. double Box::getArea ()
13. {
14.     return length * width;
15. }
```

例：类的成员函数

调用成员函数：在对象上使用点运算符“.”，这样它就能操作与该对象相关的数据和函数。

【参考程序】

```
#include<iostream>
using namespace std;
class Box
{
    public:
        //成员变量声明
        double length;
        double width;
        //成员函数声明
        double getArea();
        double getPerimeter()
        {
            return 2*(length+width);
        }
};
double Box::getArea()
{
    return length * width;
}
int main()
{
    Box box;
    cin >> box.length >> box.width;
    cout << box.getArea() << endl;
    cout << box.getPerimeter() << endl;
    return 0;
}
```

【输入样例】

```
5
6
```

【输出样例】

```
30
22
```

类的私有成员

声明为 public 的成员叫做公有成员：

- 公有成员在程序中类的外部是可访问的。
- 可以不使用成员函数来设置和获取公有变量的值。

声明为 private 的成员叫做私有成员：

- 私有成员变量或函数在类的外部是不可访问的。
- 默认情况下，类的所有成员都是私有的。

实际操作中，一般会在私有区域定义数据，在公有区域定义相关的函数，以便在类的外部也可以调用这些函数。

在 C++ 中，每一个对象都能通过 **this** 指针来访问自己。

【参考程序】

```
#include<iostream>
using namespace std;
class Box
{
    private:
        double length;
        double width;
    public:
        void setLength(double length);
        void setWidth(double width);
        double getLength();
        double getWidth();
        double getArea();
};
void Box::setLength(double length)
{
    this->length = length;
}
void Box::setWidth(double width)
{
    this->width = width;
}
```

```
23. double Box::getArea()
24. {
25.     return length * width;
26. }
27. double Box::getLength()
28. {
29.     return length;
30. }
31. double Box::getWidth()
32. {
33.     return width;
34. }
35. int main()
36. {
37.     Box box;
38.     box.setLength(6.0);
39.     box.setWidth(7.0);
40.     cout << box.getArea() << endl;
41.     return 0;
42. }
```

构造函数和析构函数

类的构造函数

- 是类的一种特殊的成员函数，它在每次创建类的新对象时执行。
- 构造函数的名称与类的名称是完全相同的，并且不会返回任何类型，也不会返回 void。
- 构造函数可用于为某些成员变量设置初始值。
- 默认的构造函数没有任何参数，但如果需要，构造函数也可以带有参数，这样在创建对象时就会给对象赋初始值。

类的析构函数

- 类的一种特殊的成员函数，它会在每次删除所创建的对象时执行。
- 析构函数的名称与类的名称是完全相同的，只是在前面加了个波浪号（~）作为前缀，它不会返回任何值，也不能带有任何参数。
- 析构函数有助于在跳出程序（比如关闭文件、释放内存等）前释放资源。

【参考程序】

```
#include<iostream>
using namespace std;
class Box
{
    private:
        double length;
        double width;
    public:
        Box();
        Box(double length, double width);
        ~Box();
        double getArea();
};
Box::Box()
{
    cout << "created 1" << endl;
    length = 1;
    width = 1;
```

```
}
Box::Box(double length, double width)
{
    cout << "created 2" << endl;
    this->length = length;
    this->width = width;
}
Box::~Box()
{
    cout << "deleted " << length << endl;
}
double Box::getArea()
{
    return length * width;
}
int main()
{
    Box box1;
    cout << box1.getArea() << endl;
    Box box2(4, 5);
    cout << box2.getArea() << endl;
    return 0;
}
```

【运行结果】

```
created 1
1
created 2
20
deleted 4
deleted 1
```