

# 开始写 C++ 程序

如果你已经学会了 C 语言，学习更多的库函数后，你就可以写出任何的程序了(为什么？)。为什么要还要学习 C++ 呢？因为 C++ 提供帮助你更好地设计和扩展你的程序的规模的的语法结构 -- 它支持 面向对象程序设计 object-oriented programming (OOP) 的编程方法。

## 1. 什么是复合数据类型 compound data type

现实世界中的物体通常有多个属性，如一个学生有姓名，学号，年龄，班级等属性。因此在程序中把物体抽象地表示为数据的时候，每个物体需要使用多个基本数据类型来表示。C++ 中，你可以把同一个物体的多个基本类型的数据组合起来并定义为一个 复合数据类型。

## 2. 复合数据类型的用法

基本数据类型如 `int` 和 `double`，可以直接使用。但是，复合数据类型需要定义才能实用。以下例子中定义了一个数据类型，用来在程序中抽象地表示一个学生。这里，抽象的意思是我们用数字表示一个学生这个真实的物体，而且抽除了我们不关心的属性。

```
1 class Student
2 {
```

```

3 public:
4     char name[20];
5     int id;
6     double age; // = 18.5
7 };

```

这里，我们定义了 Student 这个复合数据类型，这个类型包含了三个基本数据类型变量，它们叫做成员变量 member variable。例子中的 class 和 public 是保留字。我们暂时不理 public 是什么。定义复合数据类型的格式是：

**class 类型名 { public: 声明多个成员变量 };**

这里，类型名可以是任意的 C 标示符。一般，类型名用大写字母开头，而变量名和函数名用小写字母开头。复合数据类型又叫作抽象数据类型 abstract data type。

定义了这个复合数据类型，我们就可以定义这个复合数据类型的变量。在 C++ 里，复合数据类型(或抽象数据类型)简称为类 class，而复合数据类型的变量叫做对象 object。此后，我们一律使用类和对象这两个叫法。看下面定义和使用对象的例子。

```

8 Student s;
9 strcpy(s.name, "Cong Liu");
10 s.id = 12340001;
11 s.age = 18.5;

```

第一行定义了 `Student` 类的一个对象 `s`。可以看到定义对象与定义基本数据类型的变量具有同样的语法:

类名 对象名; //类的 对象 的定义  
类型名 变量名; //基本数据类型 的 变量 的定义

在 2-3 行中, 我们给这个对象赋值。访问对象中的成员变量的格式是:

对象名 . 成员变量名

事实上, 在第 8 行定义 `s` 的时候, 分配给 `s` 的存储空间大小是 32 个字节, 包含 20 个 `char`、1 个 `int` 和 1 个 `double` 所需的存储空间。而 1-7 行并不是真正的程序, 它只是告诉编译器, 每个 `Student` 类的对象会拥有的成员变量。因此, 在 1-7 行并没有定义变量和分配存储空间。

1. 把一个对象赋值给另一个对象, 相当于把一个对象的所有成员变量逐个赋值 member-wise copy 给另一个对象的成员变量。在下面的例子中的 24-25 行中分别进行了两次对象的赋值。在 25 行中, `main` 函数的局部变量 `s2` 赋值给了 `print` 函数中的 `s3`

```
12 void print(Student s3) {  
13     cout << "Name is " << s3.name << endl;  
14     cout << "ID is " << s3.id << endl;  
15     cout << "Age is " << s3.age << endl;  
16 }
```

```

17
18 int main() {
19     Student s1;
20     strcpy(s1.name, "Cong Liu");
21     s1.id = 12340001;
22     s1.age = 18.5;
23     Student s2;
24     s2 = s1; // copy
25     print(s2); // copy to s3 in print
26 }

```

类除了上述的使用便利外，更重要的是它提供更好的管理数据和程序的方式。另外，在 C++ 中，除了可以声明属于某个类的成员变量，还可以定义专门用来处理这个类的成员变量的函数。

### 3. 在类中定义的函数

在类里面，除了可以声明成员变量，也可以定义函数。这种函数一般专门用来处理这个类的成员变量的函数，这种函数叫做成员函数 member function，也叫对象函数 object function。上面的例子中的 print 函数用来输出 Student 类的各个成员变量的值；在下面的例子中我们把它写到类 Student 里面。请注意比较两个程序的差别。

```

27 class Student
28 {
29 public:
30     char name[20];
31     int id;

```

```

32     double age;
33
34     void print() {
35         cout << "Name is " << name << endl;
36         cout << "ID is " << id << endl;
37         cout << "Age is " << age << endl;
38     }
39 };
40
41 int main() {
42     Student s1;
43     strcpy(s1.name, "Cong Liu");
44     s1.id = 12340001;
45     s1.age = 18.5;
46     Student s2;
47     s2 = s1; // copy
48     s2.print(); // call an object function
49 }

```

首先，与 12-16 行的普通函数相比，34-38 行的对象函数没有了参数 s3，而且函数里面可以直接访问 name，id，和 age。但是，刚才讲过 30-32 行并不是定义变量，那么，35-37 行中的 name，id，和 age 是什么呢？实际上，每一个对象函数都有一个隐形的参数，这个隐形的参数是所在类的一个对象，而 35-37 行中的 name，id，和 age 就是这个默认参数的成员变量。

那么，这个隐形的参数是什么呢，它是如何传给对象函数的呢？48 行给出了对象函数调用的方式：

**对象 . 对象函数(实参表)**

对于 48 行的对象函数的调用，它的隐形的参数就是 s2 这个对象。归纳一下，对象函数必须通过函数所在的类的一个对象来调用，这个对象叫做对象函数的默认参数对象；另外，在对象函数里面访问默认参数对象的成员变量的时候，这个对象隐形了。同样地，在一个对象函数里面调用其它对象函数时，可以省略对象，格式与调用普通函数相同：

~~默认参数对象~~：默认参数对象的其它函数(实参表)

## 4. 两个特殊的函数

在 C++ 中，有数种特殊的对象函数，这里先介绍两种：构造函数 constructor 和 析构函数 destructor。请看下面的例子。

```
1 class Student
2 {
3 public:
4     char name[20];
5     int id;
6     double age;
7
8     Student() { // constructor
9         cout << "Hello world." << endl;
10        strcpy(name, "NO BODY");
11    }
12
13    void print() {
14        cout << "Name is " << name << endl;
15        cout << "ID is " << id << endl;
16        cout << "Age is " << age << endl;
17    }
```

```

18
19     ~Student() { // destructor
20         cout << "Bye world." << endl;
21     }
22 };
23
24 int main() {
25     Student s; // constructor is called here
26     s.print();
27     // destructor is called here
28 }

```

8-11 行定义了 Student 类的一个构造函数，19-21 行定义了该类的析构函数。以下是这两个函数的规定。首先，这两个函数都必须没有返回值，而且必须省略函数名前面的 void。第二，它们的名字必须与类的名字相同，而析构函数名字前面多加一个~。这两个特殊的对象函数，我们是不可以调用的。实际上，它们起特殊的作用：构造函数是用来初始化一个对象的成员变量的，会被自动调用，而且是每个对象调用的第一个函数；析构函数会在变量被释放前一刻被调用，而且是每个对象调用的最后一个函数。上面例子的输出如下

```

Hello world.
Name is NO BODY
ID is 0
Age is 0
Bye world.

```

在 25 行，对象 s 被创建的时候，它会自动执行它的构造函数，所以我们看到输出的第 1 行，同时 s 的成员变量 name 被初始化了。当 26 行的 print 被调用时，我们看到了 name

的初始值被输出。而另外的两个成员变量 `id` 和 `age` 的值是不能确定的，因为它们没有被初始化，而且它们是对象 `s` 这个局部变量的一部分。最后，在函数将要结束，局部变量 `s` 将要被释放前，`s` 自动地调用了它的析构函数。

## 5. 操作符函数

C++里面，其中一个操作数以上为对象的操作符，如 `s+1` 或 `s+s2`，实际上是函数。也就是，当编译器看到这个操作符的时候，如果其中一个操作符是对象，会把操作翻译成为函数调用。注意，两个操作数都是基本数据类型时，编译器把它翻译为简单指令，而不会翻译成函数。同时，C++允许我们定义操作符对应的函数，叫做操作符函数。

定义操作符函数与定义普通函数具有同样的格式，不同的是，操作符函数的参数个数是规定的：二元操作符必须有两个参数，一元操作符必须有一个参数。

```
29 Student combine(Student s1, Student s2) {
30     return (s1.age < s2.age ? s1 : s2);
31 }
32
33 Student operator + (Student s1, Student s2) {
34     return (s1.age < s2.age ? s1 : s2);
35 }
36
37 int main() {
38     Student c1;
39     Student c2;
40     strcpy(c1.name, "name1");
41     strcpy(c2.name, "name2");
```



```

42     c1.id = 1;
43     c2.id = 2;
44     c1.age = 18.1;
45     c2.age = 18.2;
46     Student c3 = combine(c1, c2);
47     Student c4 = c1 + c2; // calls 'operator +'
48 }

```

上面的例子中的 `combine` 函数与 `+` 号操作符定义的唯一区别是函数名 `combine` 换成了 `operator +`，其中 `operator` 是保留字。在 47 行中，编译器会把 `c1+c2` 匹配为 33 行的函数，因为它的操作符及左右操作数的类型都匹配。

操作符函数也可以定义为对象函数，这时它会少一个参数，因为 C++ 规定：对于二元操作符，对象函数左操作数必须是默认参数；对于一元操作符，对象函数唯一操作数必须是默认参数。下面的例子把 33-35 行的操作符函数改为对象函数。

```

49 class Student
50 {
51 public:
52     char name[20];
53     int id;
54     double age;
55
56     Student operator + (Student s2) {
57         Student s1 = (*this);
58         return (s1.age < s2.age ? s1 : s2);
59     }
60 };

```

在 57 行，`this` 是一个保留字，它代表对象函数的默认参

数的地址，它的类型是所在类的地址。所以 `s1` 获得了默认参数的值。调用这个操作符函数的方式与 47 行相同。当然，如果同时定义了 56-59 行和 33-35 行的操作符函数，在 47 行会有歧义函数调用 `ambiguous function call` 的编译错误。

我们以后会看到其它的操作符。其实，`=`号也是可以定义的操作符。还有，`cout << "Hello"` 就是一个操作符函数，它的左操作数是 `cout` 这个代表控制台输出对象，而右操作数是一个字符数组。

## 6. C++ 的类库

C++ 在原来 C 的函数库的基础上加入了很多的类，包括一些你已经使用过的，如 `fstream`。我们会以 C++ 的类库中的 `string` 和 `vector` 为例子讲解 C++ 的语法。其实 `string` 是用来代替 C 语言里面的字符数组(`char *`)的，`string` 类里的成员变量就是一个字符数组。`string` 类的作用就是简化字符数组的使用，即我们从下一章开始将介绍的封装这个概念。同样，`vector` 是用来封装 C 语言里面的数组的。

## 7. 接下来学些什么

我们揭开了 C++ 的神秘面纱，接下来我们将介绍更多的 C++ 中引入的语言结构。学习如何利用 OOP 提供的数据封装和函数多态性这两个机制来设计大规模的程序。

## 习题

### 1. 用自己的话解释下面的概念

复合数据类型 / 抽象数据类型 / 类

对象

成员变量

成员函数 / 对象函数

对象函数的默认参数

构造函数

释构函数

操作符函数

### 2. 用自己的话回答下面的问题

类是怎样定义的？

对象怎样定义的？

怎样通过对象访问它的成员变量和成员函数？

一个对象函数的默认参数是什么，如何访问默认参数的成员变量，如何访问默认参数本身？

构造函数和释构函数有那些特点，它们什么时候会被调用？

如何定义和调用操作符函数？

### 3. 以面向对象的方式写一个同学录软件

要求我们可以在同学录中添加同学的信息，同学的信息至少包含同学。可以根据学号或姓名查找和删除同学的信息。最后可以显示同学录中所有同学的信息。

程序中设计两种事物：同学和同学录，我们可以定义两个类。同学类包含最少两个属性：姓名和学号。同学录包含一个同学类的数组。注意，类的成员变量除了可以是基本数据类型，也可以是其它的复合数据类型的。同学录类里因该定义一些对象函数，已完成如插入、查找、删除、等功能。同学类里面也需要定义一些对象函数。设计的原则应该是：把专门处理一个类的成员变量的函数设计为这个类的对象函数。

输入输出的格式请自己定义，但是提交作业时请说明清楚。