为什么要有函数

我们已经学习了程序设计中的三种基本控制结构：顺序、分支、循环。利用它们可以组成任何程序。但在实际应用中，还经常用到子函数的结构。

在程序设计中，我们会经常发现一些相同功能的程序片段在程序的不同地方反复出现。此时可以将这些程序片段作为相对独立的整体，用一个标识符给它起一个名字，凡是程序中出现该程序片段的地方，只要简单地写上其标识符即可。这样的程序片段，我们称之为函数。

函数的使用不仅缩短了程序，节省了内存空间及减少了程序的编译时间，而且有利于结构化程序设计。因为一个复杂的问题总可将其分解成若干个子问题来解决，如果子问题依然很复杂，还可以将它继续分解，直到每个子问题都是一个具有独立任务的模块。这样编制的程序结构清晰，逻辑关系明确，无论是编写、阅读、调试还是修改，都会带来极大的好处。

在一个程序中可以只有主函数而没有子函数，但不能没有主函数。这也就是说子函数是不能单独执行的。

我们已经使用过了 C++提供的各种标准函数，如 abs()，sqrt()等等。系统提供的函数为我们编写程序提供了很大的方便。比如：求 sin(1)+ sin(2)+ . . . +sin(100)的值。

在实际编程过程中，我们会经常需要实现自定义的函数。

第一个函数

下面的代码定义并使用了函数，同学们能够读懂这个程序吗？

```
1.  #include<iostream>
2.  using namespace std;
3.  int max2(int, int);
4.  int main()
5.  {
6.      int a = max2(3, 5);
7.      int b = max2(10, -1);
8.      cout << a << " " << b << endl;
9.      return 0;
10. }
11. int max2(int x, int y)
12. {
13.     return (x > y) ? x : y;
14. }
```

这个程序的输出结果是

```
5 10
```

第 3 行，是函数 max2 的原型声明。

第 11~14 行，是函数 max2 的具体定义实现。

函数的定义

下面是函数 max2 的定义实现代码。

```
1. int max2(int x, int y)
2. {
3.     return (x > y) ? x : y;
4. }
```

这个函数的定义遵循了如下函数定义的语法形式:

```
1. 数据类型 函数名(形式参数表)
2. {
3.     //函数体执行语句
4.     ...
5. }
```

关于函数的定义有如下说明:

函数名是标识符，一个程序中除了主函数名必须为 main 外，其余函数的名字在遵循标识符的取名规则的前提下任意选取，最好取有助于记忆的名字。

形式参数（简称形参）表可以是空的（即无参函数），也可以有多个形参。形参间用逗号隔开，不管有无参数，函数名后的圆括号都必须有。形参必须有类型说明，形参可以是变量名、数组名或指针名。

形式参数的作用是实现主调函数与被调函数之间的关系。函数在没有被调用的时候是静止的，此时的形参只是一个符号，它标志着在形参出现的位置应该有一个什么类型的数据。函数在被调用时才执行，也就是在被调用时才由主调函数将实际参数（简称实参）的值赋予形参。这与数学中的函数概念相似，如数学函数：$f(x) = x^2 + x + 1$，这样的函数只有当自变量被赋值以后，才能计算出函数的值。

函数中最外层一对花括号“{ }”括起来的若干个说明语句和执行语句组成了一个函数的函数体。由函数体内的语句决定该函数功能。如果函数内部没有任何可执行语句，就叫做空函数。

函数不允许嵌套定义，在一个函数内定义另一个函数是非法的。但是函数允许嵌套使用。

函数的返回类型是函数的返回值的类型。如果函数不需要返回值，就把返回类型定义为 void。

【函数定义示例】

```
1.  void fa()
2.  {
3.      ...
4.  }
5.  void fb(int x, char y)
6.  {
7.      ...
8.      return;
9.  }
10. int fc(double a[], char *b)
11. {
12.     ...
13.     return 1;
14. }
```

上面代码中：

- 函数 fa()是一个没有返回值的无参函数；
- 函数 fb()是一个没有返回值的有参函数；
- 函数 fc()是一个有返回值的有参函数，其中一个参数的类型是整型数组。

函数的声明

和变量要先声明，再使用一样。调用函数之前，也先要声明函数原型。

下面程序的第 3 行，就对函数 max2 的原型进行了声明。

```
1.  #include<iostream>
2.  using namespace std;
3.  int max2(int, int);
4.  int main()
5.  {
6.      int a = max2(3, 5);
7.      int b = max2(10, -1);
8.      cout << a << " " << b << endl;
9.      return 0;
10. }
11. int max2(int x, int y)
12. {
13.     return (x > y) ? x : y;
14. }
```

函数声明的形式如下：

```
1.  类型说明符 被调函数名(含类型说明的形参表);
```

下面对 max2 ()函数原型的声明是合法的：

```
1.  int max2(int x, int y);
```

在函数声明中，参数的名称并不重要，只有参数的类型是必需的，因此下面也是合法的声明：

```
2.  int factorial(int);
```

可以看到函数原型声明与函数定义时的第一行类似，只多了一个分号。

如下面程序所示，如果函数的具体功能定义，写在了函数的第一次调用之前，是可以省略函数原型声明的。

```
1.  #include<iostream>
2.  using namespace std;
3.  int max2(int x, int y)
4.  {
```

```
5.      return (x > y) ? x : y;
6.  }
7.  int main()
8.  {
9.      int a = max2(3, 5);
10.     int b = max2(10, -1);
11.     cout << a << " " << b << endl;
12.     return 0;
13. }
```

函数的调用

声明了函数原型之后，便可以按如下形式调用函数：

```
1. 函数名(实参列表)
```

在主调函数中的参数称为实参，实参一般应具有确定的值。实参列表中应给出与函数原型形参个数相同、类型相符的实参。实参可以是常量、表达式，也可以是已有确定值的变量，数组或指针名。

函数调用可以作为一条语句，这时函数可以没有返回值。函数调用也可以出现在表达式中，这时就必须有一个明确的返回值。

具体样例如下面程序的第 6 行，和第 7 行。

【参考样例】

```
1. #include<iostream>
2. using namespace std;
3. int max2(int, int);
4. int main()
5. {
6.     int b = max2(10, -1);
7.     cout << max2(3, 5) << " " << b << endl;
8.     return 0;
9. }
10. int max2(int x, int y)
11. {
12.     return (x > y) ? x : y;
13. }
```

函数的返回值

【有返回值的情况】

返回语句 return 的一般形式是：

```
return 表达式;
```

其功能是把程序流程从被调函数转向主调函数并把表达式的值带回主调函数，实现函数的返回。表达式的值实际上就是该函数的返回值。其返回值的类型即为它所在函数的函数类型。

【无返回值的情况】

当一个函数没有返回值时：

- 函数中可以没有 return 语句；
- 也可以有 return 语句，但 return 后没有表达式。

即：

```
return;
```

这时函数没有返回值，而只把流程转向主调函数。

【参考程序】

```
void fa()
{
    ...
}
void fb(int x, char y)
{
    ...
    return;
}
int fc(double a[], char *b)
{
    ...
    return 1;
}
```