C 语言教程讲义

Potato

2024 年 7 月 13 日

第六章

本文件编译于 2024 年 7 月 13 日

6 函数

6.1 函数的定义

想象一下，我们有一个需求，需要输入三个整型 a、b 和 $c(c \neq 0)$，计算这样一个数字

$$x = \frac{a^2 + b^2}{c^2} + \frac{a(a^2 - c^2)}{b^2 + c}$$

我们的程序会写成这样

```c
#include <stdio.h>

int main() {
    int a;
    int b;
    int c;
    scanf("%d %d %d", &a, &b, &c);

    if (c == 0) {
        printf("c不能为0!\n");
    } else {
        double x = (double)(a * a + b * b) / (c * c) + (double)(a * (a * a - c * c)) / (b * b + c);
        printf("%lf\n", x);
    }

    return 0;
}
```

这里算 x 的式子看着就让人头大，也许我们可以提前把三个数的平方算出来，把程序改成这样

```c
#include <stdio.h>

int main() {
    int a;
    int b;
    int c;
    scanf("%d %d %d", &a, &b, &c);

    int a2 = a * a;
    int b2 = b * b;
    int c2 = c * c;

    if (c == 0) {
        printf("c不能为0!\n");
    } else {
        double x = (double)(a2 + b2) / c2 + (double)(a * (a2 - c2)) / (b2 + c);
        printf("%lf\n", x);
    }

    return 0;
}
```

这样代码确实简洁了很多，但是我们发现，对三个数进行平方运算的代码是一样的，我们如果能把这段代码提取出来，使得代码可以被复用，我们的程序就会简单很多。这就像是数学中的函数一样，我们给定一个输入，它给出一个输出。c 语言也提供了函数，便于我们复用代码。我们在定义变量的时候，有变量的声明语法和赋值语法，函数也是类似的，有函数原型和函数的实现。

我们在数学中学习函数时，知道函数有名字（比如正弦函数），函数传入一个值，并返回一个值（比如我们向正弦函数传入 $\pi/4$，返回的就是 $\sqrt{2}/2$）。有些函数可能传入两个或多个值（比如 $f(a,b,c)=a+b+c$），但是总是只返回一个值（比如 $f(1,2,3)$ 返回 6）。

c 语言中的函数也是类似的，一个函数，有函数名，参数列表和返回值。所谓函数名，就是函数的名字，对函数的取名要尽可能简明扼要地显示出函数的作用，比如起名 Square 就要优于起名为 S，函数使用大驼峰命名，就像命名常量一样；参数列表，就是传入函数的变量，每个参数之间用逗号连接，如我们要传入两个整型，就可以写成 int a, int b；至于返回值，就是函数计算后得到的值。由此，我们得到了定义函数原型的语法：返回值 函数名 (参数列表);

比如，我们想要实现这个函数

$$\text{Square}(x)=x^2,\ x\in\mathbb{Z}$$

就可以这样写它的函数原型

```c
int Square(int x);
```

这行代码意味着我们的函数名为 Square，一看就知道是用于计算平方的。它的参数列表只有一个整型 x，返回值也是整型。

注意，函数原型要写在 main 函数之前。有了函数原型，就要写函数的具体实现了，其语法是将函数原型照抄一遍，但是将最后的分号改成花括号，将程序主体在写花括号内，比如

```
int Square(int x) {
    return x * x;
}
```

函数实现写在哪都可以，只要写在函数原型之后即可。这里的 return 就是用于返回返回值的。函数执行到 return 就结束了，因此一切写到 return 后的语句都是无效的。当我们调用函数 Square 时，里面的程序就会像 main 函数一样从上到下依次执行。

接下来就要调用函数了，调用函数的语法是函数名 (参数)。也就是说，我们可以这样调用 Square。

```
Square(a);
```

那么，我们怎么获得函数的返回值呢？我们定义一个变量，用它来接收即可，比如

```
#include <stdio.h>

int Square(int x);

int Square(int x) {
    return x * x;
}

int main() {
    int a = 2;
    int b = Square(a);

    printf("a的平方是%d\n", b);

    return 0;
}
```

运行程序，输出结果是 4。我们明明是对 a 进行计算，为什么函数里使用的是 x 呢？观察 main 函数内的代码，我们发现，我们将 a 作为参数传给了 Square，因此 Square 内的 x 就是 a。

就像是变量的声明和赋值可以放在一起一样，函数原型和函数实现也可以写在一起，像是上面的代码，我们可以改写成

```
#include <stdio.h>

int Square(int x) {
    return x * x;
}

int main() {
    int a = 2;
    int b = Square(a);
```

```
10
11          printf("a的平方是%d\n", b);
12
13          return 0;
14      }
```

既然如此，为什么还要写一个函数原型呢？我们前面说，函数实现可以写在其它任何地方，也就是说，上面的代码可以写成

```
1   #include <stdio.h>
2
3   int Square(int x);
4
5   int main() {
6       int a = 2;
7       int b = Square(a);
8
9       printf("a的平方是%d\n", b);
10
11      return 0;
12  }
13
14  int Square(int x) {
15      return x * x;
16  }
```

如果我们有一堆函数，我们就可以只在 main 前写函数原型，将所有的函数实现写在后面，这样代码更加简洁。后面学习头文件时，我们会对这一点有更深刻的了解。

当然了，我们也可以在函数里写除了求值外的其它功能，比如将 Square 改成

```
1   int Square(int x) {
2       printf("计算%d的平方!\n", x);
3       return x * x;
4   }
```

如果我们的函数不需要返回值什么办？比如我们写一个函数用来打印 n 次 Hello，函数体内肯定是这样的

```
1   for (int i = 0; i < n; ++i) {
2       printf("Hello\n");
3   }
```

这个函数不需要返回值，因此使用 void 写在返回值的地方。因此这个函数就变成了

```
1   void SayHello(int n) {
2       for (int i = 0; i < n; ++i) {
3           printf("Hello\n");
```

```
    }
}
```

在调用它的时候，也就不需要一个变量要接收它的返回值了，我们可以直接这样写

```
#include <stdio.h>

void SayHello(int n) {
    for (int i = 0; i < n; ++i) {
        printf("Hello\n");
    }
}

int main() {
    SayHello(2);

    return 0;
}
```

实际上，有返回值的函数也不必须要接收它的返回值，但是这样就浪费了它的返回值。

我们在写函数原型时写的参数列表叫做函数的形式参数，简称形参。在调用函数时传入的参数叫做函数的实际参数，简称实参。

一个函数可以没有返回值，也可以没有形参。比如

```
#include <stdio.h>

void SayHi() {
    printf("Hi\n");
}

int main() {
    SayHi();

    return 0;
}
```

在一些比较古老的编译器中，我们需要使用 void 来显式地说明函数没有形参，比如将上面的函数改成

```
void SayHi(void) {
    printf("Hi\n");
}
```

我们写一个函数，可能需要让它完成很多工作，因此我们推荐在函数原型前写多行注释要详细阐释函数的作用以及每个参数、返回值的作用。比如

```
/*
计算数字的平方
```

```
3     x 需要计算的底数
4     return 传入数的平方值
5     */
6     int Square(int x);
```

另外值得注意的是，函数就像变量一样，是不能重名的。比如这样是不允许的

```
1     int Square(int x);
2     int Square(int x);  // 重复定义!
```

观察这种情况

```
1     int Square(int x);
2     int Square(int x, int y);
```

像这样，两个函数只是名称相同，但参数列表或返回值不同的情况叫做函数的重载。c 语言不支持函数重载，因此上面的代码也是错误的。但是很多计算机语言都是允许这样的，如 C++和 Java。

6.2 main 函数

我们一直在 main 函数内写程序，它是程序入口，main 函数中的程序会有上到下执行，我们的一切程序都是由 main 函数组织起来的。main 函数写成

```
1     int main() {}
```

这说明它是一个返回值为整型的函数，我们之前一直在代码最后写

```
1     int main() {
2         return 0;
3     }
```

这就是 main 函数的返回值了。我们默认返回 0，因为这代表着程序正常执行。那么这个返回值在哪里能看到呢? 我们执行完程序后，会发现终端最后有一行 “退出代码 0”，这就是 main 函数的返回值。如果我们把 main 的返回值改成别的数字，那么这里的 “退出代码” 也会改变。

我们看到，main 函数就是一个没有形参，返回值为整型的函数。实际上 main 函数还可以写成这样

```
1     int main(int argc, char *argv[]) {}
```

这里的 main 接收两个参数，后面这个奇形怪状的 argv 暂且不需要知道是干什么的，我们在学习数组时会介绍它的作用。