第5章　函　　数

引言

在程序设计过程中，如果程序的功能比较多，规模比较大，把所有代码都写在主函数中，会使得主函数变得庞杂、逻辑不清，阅读和维护变得困难。在程序中，如果要多次实现某一特定功能，需要多次重复编写实现此功能的程序代码，就会使得程序冗长、不精炼。解决的办法就是采用模块化编程，即函数实现。函数是程序设计中具有独立功能的一段程序，是程序的主要组成部分。C++ 程序由一个主函数和若干其他函数构成，主函数可以调用其他函数，其他函数也可以互相调用。

学习目标

- 了解：函数的作用；内存模型。
- 熟悉：函数的定义、声明与调用；函数的嵌套调用与递归调用；作用域和存储类别。
- 掌握：函数的编制方法；函数重载、内联函数和带默认值函数的特点。

课程思政

团队教育：通过函数模块化编程思想引入团队教育，讲解团队的重要性以及如何组建、分工与协作，培养学生的团队精神和协作能力。


函数的定义与声明

5.1　函数的定义与调用

5.1.1　函数概述

模块化程序设计就是把一个复杂问题分解成多个模块，每个模块独立完成一定的功能，主程序按照问题求解算法调用这些模块。模块可以通过函数实现，这样主程序更加简洁，逻辑思路更加清晰。C++ 是模块化编程语言。在面向过程程序设计中，模块就是函数；在面向对象程序设计中，模块就是类，类里面再包括成员函数。函数的好处是，用户无须知道被调用函数内部是如何实现的，直接调用函数实现相应功能，从而简化了用户的编程。

当启动一个 C++ 程序时，从 main()函数开始执行。main()函数可以调用其他函数，被调用函数执行完成后再返回 main()函数，最后由 main()函数返回以结束程序。一般情况下 main()函数不允许被其他函数调用。一个 C++ 程序可由多个源文件组成，但有且仅有一个 main()函数。

1. 系统函数和用户自定义函数

从函数定义的角度看，可分为系统函数和用户自定义函数两类。

(1) 系统函数是 C++ 编译系统、操作系统或其他系统为方便用户编程而预定义的函数。这些函数在特定的头文件中都有原型说明。例如，iostream 头文件包含了一组处理输

入输出的对象和函数；cmath 包含了一组数学计算的函数，如求绝对值的函数 abs()、求平方根的函数 sqrt()。用户只需在程序前包含这些头文件，就可以在下面的代码中直接调用这些函数。

(2) 用户自定义函数是用户根据问题求解的需要自己编写的函数。这类函数往往功能独特，使用范围比较有限，例如例 5.1 中的 SumAbs()、PrintStars()函数。自定义函数是程序设计最常见的方法，算法实现主要就是函数设计。

2. 有返回值的函数和无返回值的函数

根据函数返回的结果，可分为有返回值和无返回值两类。

(1) 有返回值的函数被调用执行完后将向调用者返回一个执行结果，称为函数返回值。数学函数即属于此类函数，如求绝对值的函数 abs()。由用户定义的有返回值的函数，必须在函数定义和函数声明中明确返回值的类型。

(2) 无返回值的函数用于完成某项特定的处理任务，执行完成后不向调用者返回函数值。这类函数类似于其他语言的过程。由于函数无须返回函数值，用户在定义此类函数时可指定它的返回为空类型，空类型的说明符为 void。例如，PrintStars()函数实现打印星号功能，无返回值。

3. 无参函数和有参函数

从主调函数和被调函数之间数据传送的角度看，又可分为无参函数和有参函数。

(1) 无参函数是指函数定义、函数声明及函数调用中均不带参数。主调函数和被调函数之间不进行参数传送。此类函数通常用来完成一组指定的功能，可以返回或不返回函数值，如 PrintStars()函数。

(2) 有参函数也称为带参函数。在函数定义及函数声明时都有参数，简称形参，称为形式参数。在函数调用时也必须给出参数，称为实际参数，简称实参。进行函数调用时，主调函数将把实参的值传送给形参，供被调函数使用，如 SumAbs()函数。

【例 5.1】 函数实例。

```
#include<iostream>
using namespace std;
#include<cmath>                                  //包含若干数学函数的头文件
int SumAbs(int num1, int num2)                   //用户自定义函数,求两数绝对值之和
{
    int n1, n2, sum;
    n1 = abs(num1);                              //系统定义的函数,获取绝对值
    n2 = abs(num2);
    sum = n1 + n2;
    return sum;
}
void PrintStars()                                //用户自定义函数,打印星号
{
    for (int i = 0; i < 10; i++)
    {
        cout << '*';
    }
    cout << endl;
```

```
}
int main()                                          //主函数
{
    int num1, num2;
    PrintStars();                                   //函数调用,打印星号
    cout << "请输入任意两个数:";
    cin >> num1 >> num2;
    cout << "两数的绝对值之和为:" << SumAbs(num1, num2) << endl;
                                                    //函数调用,求两数绝对值之和
    PrintStars();                                   //函数调用,打印星号
}
```

程序执行结果如下：

```
**********
请输入任意两个数:-10 20
两数的绝对值之和为:30
**********
```

在本例中，abs 是系统定义的绝对值函数，可以直接调用，但是需要使用语句 #include <cmath> 将相应的头文件包含进来。用户自定义了有参数和返回值的 SumAbs 函数，实现求两个数的绝对值之和，定义了无参数和返回值的 PrintStars 函数，打印一行星号。

5.1.2 函数的定义

在 C++ 程序中用到的函数必须先定义后使用。函数定义的语法格式如下：

```
函数类型  函数名(形参表)
{
    函数体
}
```

一个函数定义由两部分组成：函数头和函数体。函数头包括函数类型（即返回值类型）、函数名以及形参表，确定了该函数的语义功能。函数体为函数提供一种实现方式，用一对花括号括起来，由一组语句组成，确定了该函数执行时的具体操作。

【例 5.2】 用户自定义函数 Max 和 Welcome，分别实现求两个数中的较大数和显示欢迎信息的功能。

```
#include<iostream>
using namespace std;
int Max(int num1, int num2)             //有参数和返回值的函数,求两个整数中的较大数
{
    return num1 > num2 ? num1 : num2;
}
void Welcome()                          //无参数和返回值的函数,显示欢迎信息
{
    string name;
    cout << "请输入您的名字:";
    cin >> name;
    cout << name << ",欢迎您!";
```

```
    return;
}
int main()                                  //主函数
{
    int num1, num2;
    cout << "请输入任意两个数:";
    cin >> num1 >> num2;
    cout << "两个数中的较大数为:" << Max(num1, num2) << endl;   //函数调用,求较大数
    Welcome();                              //函数调用,显示欢迎信息
    return 1;
}
```

程序执行结果如下：

```
请输入任意两个数:10 20
两个数中的较大数为:20
请输入您的名字:阿强
阿强,欢迎您!
```

说明：

(1) 函数类型指该函数结束执行时要返回结果的数据类型。若不需要返回值可以写为void。

(2) 函数名是用户自定义的名称,要符合C++标识符的命名规则。

(3) 形参表指调用该函数时向它传递的数据,分别对形式参数进行定义,参数之间用逗号进行间隔。例5.2中的Max()函数定义了num1和num2两个形式参数。

(4) 函数体指函数所要完成的功能。函数体包括变量声明部分和执行语句部分。

(5) 函数定义中的一对花括号不能省略,它用于指明函数体的开始和结束。

(6) 当函数有返回值时,如Max()函数,在函数体中至少应有一条return语句,函数返回值就是return后面表达式的值。当函数没有返回值时,如Welcome()函数,可以写为return;或者省略return语句。

5.1.3 函数的调用

一个函数被定义后,通过被其他函数调用实现该函数的功能。对于一个函数,只有被调用时,其函数体才能执行。一个函数调用就是确定一个函数名并提供相应的实参,然后开始执行其函数体中的语句。当函数体执行结束时,函数调用就得到了一个值,返回给调用方。

1. 函数调用的形式

函数调用的形式如下：

函数名(实参表)

函数名是已定义的一个函数的名字;实参表由零个、一个或多个实参构成。如果是调用无参函数,实参表为空,但圆括号不能省。在实参表中,每个实参都是一个表达式。实参表中的实参与被调用函数的形参表中的形参要求在个数、类型、顺序上一致。在函数调用过程中,先计算各实参的值,再赋给对应的形参,最后再启动执行函数体。

2. 函数调用的方式

按函数调用在语句中的作用划分,有3种函数调用方式。

（1）语句调用。是指函数调用作为一条语句单独出现，通过加上一个分号构成一条语句。其调用格式为

函数名(实参表);

例如，在例 5.2 中：

```
Welcome();
Sum(10, 20);
```

都是函数调用语句。这种调用方式忽略了函数的返回值。对于无返回值的函数调用，就只能用这种调用方式；对于有返回值的函数调用，也能如此调用，只是忽略了返回值。但是，如果返回值比较重要，如此调用就失去了意义。

（2）表达式调用。是指函数调用作为一个表达式出现。其调用格式为

变量名=函数名(实参表);

或者

cout<<函数名(实参表);

函数调用本身就是一个表达式，也具有特定的返回值类型，因此函数调用可出现在表达式中，以函数返回值参与表达式的计算。这种方式要求函数有返回值，而不能是 void。例如：

```
max = Max(num1, num2);                    //把 Max 函数的返回值赋予变量 max
cout<<Max(num1, num2);                    //输出 Max 函数的返回值
```

（3）参数调用。是指一个函数调用作为另一个函数调用的实参。这种情况是把前者的返回值作为后者的实参，因此要求前者必须是有返回值的。例如：

```
max = Max(Max(num1, num2), num3);
```

把 Max 函数调用的返回值又作为 Max 函数的实参，求出 num1、num2、num3 的最大值。此时将返回值作为实参的内层 Max 函数调用要先执行，返回的结果作为外层 Max 函数调用的实参值，再次调用 Max 函数执行。

5.1.4 函数的声明

在编写较大规模的模块化程序时，因为程序中的函数定义较多，占用很多代码行，对整个程序的可读性会产生一定的影响。为了避免这种情况，可对函数先声明，后给出定义。函数声明又称函数原型说明，用来告诉编译器函数的名称、返回值类型以及函数要接收的参数个数、类型和顺序，编译器用函数原型验证函数调用。

一个完整的函数定义包含一个函数头和一个函数体。其中函数头就是函数原型，包括函数的名称、形参及返回值。函数原型说明的语法格式为

函数类型　函数名(参数表);

函数原型说明由函数类型、函数名和参数列表组成。参数表与函数定义中的参数表在参数个数、顺序和类型上应一致。在函数原型说明中可以不给出参数名，只给出类型。例 5.2 的程序可改写如下：

```
#include<iostream>
using namespace std;
int Max(int, int);                                          //函数原型说明
void Welcome();                                             //函数原型说明
int main()
{
    int num1, num2;
    cout << "请输入任意两个数:";
    cin >> num1 >> num2;
    cout << "两个数中的较大数为:" << Max(num1, num2) << endl;   //函数调用
    Welcome();                                              //函数调用
    return 1;
}
int Max(int num1, int num2)                                 //函数的定义
{
    return num1 > num2 ? num1 : num2;
}
void Welcome()                                              //函数的定义
{
    string name;
    cout << "请输入您的名字:";
    cin >> name;
    cout << name << ",欢迎您!";
    return;
}
```

函数的定义是平行的,不能嵌套定义,但可以嵌套调用。主函数 main 和最大值函数 Max 地位是平行的。主函数要调用 Max 函数,因此需要在主函数调用 Max 函数前,对 Max 函数进行函数原型说明,原型说明用于对调用的函数与函数的定义进行对照检查。

一个函数只能定义一次,但函数原型说明可以在程序中多次出现。先说明函数原型,使下面的代码能调用该函数,检查函数调用是否符合语法规则。函数的完整定义可以放在本源程序的后面、另一个源程序中或者编译后的库文件中。在 C++ 中,一般将 main 函数放在最前面,因为它通常提供了程序的整体结构;而将函数的完整定义放在后面。这样,程序的结构就更加清晰。

对于一组通用的函数,如果希望它能被更多的用户在更多的程序中调用,但又不能公开源代码而被人篡改,也不想反复编译而浪费时间,可将这些函数原型说明放在一个头文件(扩展名为.h)中,相应的函数定义放在其他源文件中,再将这些源文件编译为库文件,最后将头文件和库文件公开给他人使用。就像使用 cmath 头文件中的 abs 函数一样,先用 #include<cmath>说明函数原型,然后就能调用其中的 abs 函数了。

5.2 函数参数与函数返回

5.2.1 函数参数

在调用函数时,大多数情况下,主调函数和被调函数之间有数据传递关系,这就是前面提到的有参数的函数形式。函数参数的作用是传递数据给函数,使函数利用接收的数据进

行相应的操作。

1. 形式参数与实际参数

在定义函数时，把函数括号中的变量名称为形式参数，简称形参。在函数调用时，传递给函数的值将被复制到这些形参中。

在调用函数时，把函数括号中的参数称为实际参数，简称实参。实参是指在调用函数时函数的调用者提供给函数形参的实际值。实参可以为常量、变量和表达式。

函数的形参和实参具有以下特点：

（1）形参只有在被调用时才分配存储单元，在调用结束时即刻释放其占用的存储单元。因此，形参只在函数内部有效，函数调用结束返回主调函数后则不能再使用该形参变量。

（2）实参可以是常量、变量、表达式、函数等。无论实参是何种类型的量，在进行函数调用时，它们都必须具有确定的值，以便把这些值传送给形参。因此应预先用赋值、输入等办法使实参获得确定值。

（3）实参和形参在个数、类型和顺序上应严格一致，否则会发生类型不匹配的错误。

（4）函数调用中发生的数据传递是单向的。即只能把实参的值传递给形参，而不能把形参的值反向地传递给实参。因此，在函数调用过程中，形参的值发生改变，而实参的值不会改变。

【例 5.3】 求两个整数的和。

```
#include<iostream>
using namespace std;
int Sum(int, int);                              //函数原型说明
int main()
{
    int n1, n2;
    cout << "请输入任意两个整数:";
    cin >> n1 >> n2;
    cout << Sum(n1, n2);                        //函数调用,n1 和 n2 为实参
    cout << Sum(10, 2 * 10);                    //函数调用,10 和 2 * 10 为实参
}
int Sum(int num1, int num2)                     //函数的定义,num1 和 num2 为形参
{
    return num1 + num2;
}
```

程序执行结果如下：

```
请输入任意两个数:10 20
两个数中的较大数为:20
请输入您的名字:阿强
阿强,欢迎您!
```

在定义函数时，num1 和 num2 为形参。在调用函数时，变量 n1 和 n2、常数 10 和表达式 2 * 10 是实参。函数调用时，实参的值被复制给形参，在函数体内进行运算。

在 C++ 中，函数调用中要求形参和实参要一一对应。在函数调用时，系统为形参分配存储空间，并将实参的值传递给形参。在函数调用过程中，参数的传递方式有 3 种：值传递、地址传递和引用传递。下面介绍值传递和地址传递，引用传递将在 6.5.2 节详细讲解。

2. 值传递

值传递是参数传递数据最常用的方式。值传递的特点是单向传递，即主调函数调用被调函数时给形参分配存储单元，把实参的值传递给形参；在调用结束后，形参的存储单元被释放，而形参值的任何变化都不会影响到实参的值，实参的存储单元仍保留并维持其值不变。

【例 5.4】 编写 Swap 函数，实现两个整数的互换。

```
#include<iostream>
using namespace std;
void Swap(int, int);
int main()
{
    int n1 = 10;
    int n2 = 20;
    cout << "交换前,main 函数中:n1=" << n1 << " n2=" << n2 << endl;
    Swap(n1, n2);                                    //值传递,实参复制一个副本给形参
    //形参值的改变不会影响对应实参的值
    cout << "交换后,main 函数中:n1=" << n1 << " n2=" << n2 << endl;
    return 1;
}
void Swap(int num1, int num2)
{
    cout << "Swap 函数中,交换前:num1=" << num1 << " num2=" << num2 << endl;
    int temp = num1;
    num1 = num2;
    num2 = temp;
    //形参值发生了改变
    cout << "Swap 函数中,交换后:num1=" << num1 << " num2=" << num2 << endl;
}
```

程序执行结果如下：

```
交换前,main 函数中:n1=10 n2=20
Swap 函数中,交换前:num1=10 num2=20
Swap 函数中,交换后:num1=20 num2=10
交换后,main 函数中:n1=10 n2=20
```

main 函数调用 Swap 函数时，实参 n1、n2 的值传递给形参 num1、num2。在 Swap 函数中，形参 num1 和 num2 的值进行交换，但 main 函数中实参 n1 和 n2 的值并没有改变。

在本例中，Swap 函数没有真正实现两数的互换。如果要实现这个功能，就需要使用下面的地址传递这种方式。

3. 地址传递

地址传递是指形参接收到的是实参的地址，即指向实参的存储单元，形参和实参占用相同的存储单元，因此形参和实参是相同的。在以数组名或指针作为函数参数时进行的传递就是地址传递。形参在取得该地址之后，与实参共同拥有一段内存空间，形参的变化也就是实参的变化。将例 5.4 的值传递改为地址传递，就可以真正地实现两数的互换。

【例 5.5】 编写 Swap 函数，通过地址传递实现两个整数的互换。

```
#include<iostream>
using namespace std;
```

```
void Swap(int *, int *);
int main()
{
    int n1 = 10;
    int n2 = 20;
    cout << "交换前,main 函数中:n1=" << n1 << " n2=" << n2 << endl;
    Swap(&n1, &n2);                          //地址传递,实参将自己的地址传递给形参
    cout << "交换后,main 函数中:n1=" << n1 << " n2=" << n2 << endl;
    return 1;
}
void Swap(int * num1, int * num2)            //形参定义为指针,接收实参传递的地址
{
    int temp = * num1;
    * num1 = * num2;
    * num2 = temp;
}
```

程序执行结果如下：

```
交换前,main 函数中:n1=10 n2=20
交换后,main 函数中:n1=20 n2=10
```

num1、num2 被定义为指针变量（指针在第 6 章详细讲解），main 函数调用 Swap 函数时，将实参 n1、n2 的地址传递给形参。这样，形参 num1 和 num2 分别指向实参 n1 和 n2，从而就可以改变 n1 和 n2 的值。

5.2.2 函数返回

主调函数和被调函数具有严格的控制与被控制的关系。当主调函数调用被调函数后，主调函数处于等待状态，转而执行被调函数；当被调函行结束后，必须返回主调函数，主调函数才能从等待处继续执行。

被调函数通过 return 语句返回主调函数，其功能包括：①结束被调函数的执行，返回主调函数；②在结束被调函数时，可以将返回值传递给主调函数。

1. 有返回值的函数

如果函数定义时确定了返回值的类型，那么被调函数执行完后要向主调函数返回一个结果，就是函数返回值。返回值的类型就是该函数的类型。函数的返回值只能通过 return 语句返回主调函数。return 语句的一般形式为

return 表达式;

或者

return (表达式);

【例 5.6】 编写判断一个整数是否为素数的函数，实现输出 100～200 的素数，每行输出 8 个数，最后输出这些素数的和。

```
#include<iostream>
using namespace std;
int IsPrime(int n);
int main()
```

```
{
    int count = 0;                                  //控制每行素数的输出数量
    int sum = 0;                                    //100~200 的素数和
    cout << "100~200 的素数:" << endl;
    for (int i = 100; i <= 200; i++)
        if (IsPrime(i))
        {
            sum = sum + i;
            cout << i << "  ";
            if (++count % 8 == 0)
                cout << endl;
        }
    cout << endl;
    cout << "100~200 的素数之和为:" << sum;
    return 0;
}
int IsPrime(int n)                                  //定义函数,判断一个整数是否为素数
{
    int i;
    for (i = 2; i < n; i++)
        if (n % i == 0)  return 0;                  //不是素数,返回值为 0
    return 1;                                       //是素数,返回值为 1
}
```

程序执行结果如下：

```
100~200 的素数:
101  103  107  109  113  127  131  137
139  149  151  157  163  167  173  179
181  191  193  197  199
100~200 的素数之和为:3167
```

说明：

(1) return 语句将被调函数中的一个确定值带回主调函数中。

(2) 若需要从被调函数带回一个值供主调函数使用,被调函数必须包含 return 语句。

(3) 一个函数体中可以有一个以上的 return 语句,执行到哪一个 return 语句,哪一个就起作用。return 语句后面的括号可以不要。例如,return 0;等价于 return(0);。

(4) return 后的值可以是一个表达式。

(5) 在定义函数时,函数类型一般应和 return 语句中的表达式类型一致。如果两者不一致,则以函数类型为准。对数值型数据可以自动进行类型转换。

2. 无返回值的函数

如果函数定义时确定返回 void,那么该函数的调用执行完成后将不会返回任何值。例如,setw(6)是一个函数调用,它的执行不会返回值,所以只能进行单独语句调用,而不能直接参与表达式计算。注意,无返回值并不意味着函数执行没有结果。一个函数的计算结果可能作用在函数之外的数据上,而不一定要返回。