第3章

函　数

在开发较大的应用程序时，通常将程序分解为若干功能模块，每一模块实现一个特定的功能。采用模块化程序设计的优点在于可以控制程序的复杂性，便于人员分工，提高开发效率，提升软件的可靠性、可维护性和重用性等。在C++程序中，函数是程序的基本模块，通过组合函数模块的调用实现程序特定的功能，如图3.1所示。

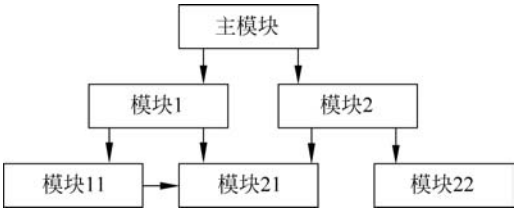

图3.1　组合函数模块的调用

在前文的程序中，都是在主函数main()中编写代码逻辑，有时还需要调用标准函数库中的一些函数。本章主要学习如何实现自定义函数，即将代码逻辑或算法编写成一个个相对独立的函数模块，通过组合调用的方式来实现程序。

3.1　概述

从用户使用的角度看，函数有两种：标准函数和自定义函数。标准函数是库函数，由系统提供，用户可以直接调用。自定义函数则是用户根据需求，自己编写用于实现特定功能的代码块。

C++语言提供极为丰富的库函数，这些库函数有数学函数、字符和字符串处理函数、输入输出函数和动态存储分配函数等。库函数的学习可以从掌握一些最基本、最常用的

库函数开始，之后再逐步深入学习。

我们来看一个使用C++标准库中的数学函数的示例。

【例 3-1】 求整数 x 的绝对值。

编写程序

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    int x;
    x=-6;
    cout<<"|"<<x<<"|"<<"is equal to"<<abs(x)<<endl;
    x=0;
    cout<<"|"<<x<<"|"<<"is equal to"<<abs(x)<<endl;
    x=+6;
    cout<<"|"<<x<<"|"<<"is equal to"<<abs(x)<<endl;
    return 0;
}
```

运行结果

```
|-6|is equal to 6
|0|is equal to 0
|6|is equal to 6
```

依靠系统给出的标准库函数是远远不够用的，更多时候则需要根据实际业务需求，编写自定义函数。

我们来看一个使用自定义函数的示例，初步了解自定义函数。

【例 3-2】 自定义函数用于实现两个整数求和（被调函数在前）。

编写程序

```
#include <iostream>
using namespace std;
int sum(int x,int y)
{
    int s;
    s=x+y;
    return s;
}
int main()
{
    int z;
    z=sum(2,6);
```

```
13      cout <<"2 和 6 两数相加的结果是: "<< z;
14      return 0;
15 }
```

运行结果

```
2 和 6 两数相加的结果是: 8
```

该程序中的第 3～8 行代码定义了名称是 sum 的函数,第 12 行中的 sum(2,6)是函数调用,利用 sum 函数求 2 加 6 之和。其中 main 函数是主调函数,sum 函数是被调函数。

该例将被调函数的定义代码放到主调函数的定义代码之前,也可以将主调函数的定义代码放到被调函数的定义代码之前,但必须在被调函数调用之前写上如下语句。

```
int sum(int x,int y);
```

该语句称为对 sum 函数声明,也称为函数原形,之所以需要在调用函数之前对该函数声明,是因为 C++编译器还做不到预扫描,只能从文件的开始到结束顺序编译,并且要求编译的执行语句中每个符号或在系统中已经规定了明确的含义或在程序中已经定义,如果没有以上的声明语句,当编译到 sum 函数调用语句时,则会提示 sum 是未声明标识符的错误。

该声明语句告知编译器 sum 是个函数,该函数有两个整型形式参数,函数的返回值的类型是整型(关于形式参数和返回值在后续章节详细介绍)。函数声明也可以理解为告知编译器该函数如何使用,编译器阅读该函数的使用说明之后,就可以使用该函数了。将被调函数定义代码写在主调函数定义代码之前的程序如例 3-2 所示。

有经验的程序编制人员一般会把主调函数(如 main 函数)写在最前面,这样程序的结构和作用一目了然,统揽全局,然后再具体了解各函数的细节。否则,当一个程序包含多个函数时,阅读程序的人需要十分耐心地逐一仔细阅读每个被调函数,直到最后才看到主调函数,这样的程序可读性较差。把主调函数写在前面的代码如例 3-3 所示。

【例 3-3】 自定义函数用于实现两个整数求和(主调函数在前)。

编写程序

```
1  #include<iostream>
2  using namespace std;
3  int sum(int x,int y);
4  int main()
5  {
6       int z;
7       z=sum(2,6);
8       cout <<"2 和 6 两数相加的结果是: "<< z;
9       return 0;
10 }
11 int sum(int x,int y)
```

```
12 {
13      int s;
14      s = x + y;
15      return s;
16 }
```

运行结果

```
2 和 6 两数相加的结果是:8
```

3.2 函数的定义

函数是指由多行代码组成,可以实现某个具体功能的程序。通俗地说,每个函数都可以视为一个工具,库函数相当于现成的工具集,自定义函数相当于自己打造的工具。在使用库函数时,需要使用 include 指令把相应的头文件包含到程序中;而对于标准库没有提供的函数,则需要在程序中按照一定的形式规范进行定义。

3.2.1 函数定义的形式

函数定义的一般形式可分为以下两种。

(1) 无参函数的定义形式如下。

```
类型标识符 函数名() {
声明部分
语句
}
```

其中类型标识符和函数名称为函数头。类型标识符指明本函数的类型,函数的类型实际上指的是函数返回值的类型。函数名是由用户定义的标识符,函数名后有一个空括号,其中无参数,但括号不能省略。{}中的内容称为函数体。在函数体中的声明部分,是对函数体内部所用到的变量的类型说明。

例如:打印一个列表的表头。

```
void printTableHead(){
    cout <<" ***************** "<< endl;
    cout <<" *** 资金收益表 *** "<< endl;
    cout <<" ***************** "<< endl;
}
```

第一行代码说明 printTableHead 函数是一个没有返回值的函数,其返回的函数值为空。没有形式参数。在{}中的函数体内,没有使用任何变量,因此只有语句而没有声明部分。函数的语句实现三行表头内容的输出。

(2) 有参函数定义的一般形式如下。

```
类型标识符 函数名(形式参数列表) {
```

```
声明部分
语句
}
```

有参函数比无参函数多了一项内容,即形式参数列表。在形式参数列表中给出的参数称为形式参数(简称“形参”),它们可以是各种类型的变量,各参数之间用逗号间隔。在进行函数调用时,主调函数将赋予形参实际的值。形参既然是变量,必须在形参列表中给出形参的类型说明。

例如,定义一个函数,用于求两个数中的较大值,可以编写如下。

```
int getMax(int a, int b) {
if (a > b) return a;
else return b;
}
```

第一行说明 getMax 函数是一个整型函数,其返回的函数值是一个整数。形参为 a,b,均为整型变量。在{}中的函数体内,除形参外没有使用其他变量,因此只有语句而没有声明部分;若要用到新的变量,需要进行声明和初始化等。在 getMax 函数体中的 return 语句是把 a(或 b)的值作为函数的值返回给主调函数。有返回值的函数中至少有一个 return 语句。

函数在使用时将被视为“黑匣子”,除输入输出外,其他部分可以不必关心。从函数的定义中可以看出,函数头反映函数的功能和使用接口,它定义用什么输入数据执行一个处理逻辑后产生什么输出。它明确“黑匣子”的输入输出部分,输入就是形参,输出就是函数的返回值。而函数体中具体描述“如何做”,但是对函数的使用者而言,不需要知道其具体是如何实现的。

需要说明的是,函数的定义既可以放在主函数 main 之前,也可放在主函数 main 之后。此外,文件中所有的函数定义,包括 main 函数在内,都是平行的,即不能在一个函数的函数体内再定义另一个函数。

3.2.2 定义与声明

函数的定义是指对函数功能的确立,包括为函数命名、指定函数返回值类型和形参类型以及编写函数体等。函数定义之后,需要进行函数声明后才能使用。函数的声明是把函数名、函数类型以及形参类型、个数和顺序通知编译系统,以便在调用该函数时系统按此进行对照检查(如函数名是否正确,实参与形参的类型和个数是否一致)。调用函数之前对被调函数进行声明与使用变量之前先进行变量声明类似。

函数声明遵循如下原则。

(1) 在实践中一般将函数声明放在程序的起始位置,可以使得代码结构清晰。

(2) 若函数定义在先,调用在后,可以省略声明;若定义在后,调用在先,则必须声明。

函数声明的一般形式如下。

```
类型说明符 被调函数名(类型 形参,类型 形参…);
```

或为：

```
类型说明符 被调函数名(类型,类型…);
```

括号内给出了形参的类型和形参名，或只给出形参类型，便于编译系统进行检查排错，保证程序正确执行。

例如，对上例中的 getMax 函数的声明如下。

```
int getMax (int a,int b);
```

或写为：

```
int getMax (int,int);
```

3.3 返回语句

调用函数时，主调函数和被调函数之间往往存在数据传递关系。主调函数通过参数将数据传递给被调函数，而被调函数则通过返回语句的返回值向主调函数传递数据。其中函数的返回值是指函数被调用后，执行函数体中的程序段所取得的返回给主调函数的值。如调用正弦函数取得正弦值，调用 getMax 函数取得的最大值等。

函数的返回值通过 return 语句返回给主调函数，语句的一般形式如下。

```
return 表达式;
```

return 语句的功能是计算表达式的值，并返回主调函数。在函数中允许有多个 return 语句，但每次调用只能有一个 return 语句被执行，因此只能返回一个函数值。例如，前文示例中的 getMax 函数，有两个 return 语句，但每次有且仅有一个被执行。

既然函数有返回值，这个值就必然有一个确定的类型，应当在定义函数时指定函数值的类型。而对于不返回函数值的函数，可以明确定义为“空类型”，类型说明符为“void”。如前文示例中的 printTableHead 函数并不向主函数返回函数值，因此定义如下。

```
void printTableHead ( )     { …         }
```

为了使程序有良好的可读性并减少出错，凡不要求返回值的函数都应定义为空类型。

3.4 函数的参数

函数的参数分为形式参数和实际参数两种。在定义函数时，函数名后面括号中的变量名称为“形式参数”，简称“形参”，在主调函数中调用函数时，函数后面括号中的参数称为“实际参数”，简称“实参”。

形参出现在函数定义中，作用范围是整个函数体，离开该函数则不能使用。实参出现在主调函数中，进入被调函数后，实参变量也不起作用。形参和实参的功能是用来进行数据传输。发生函数调用时，主调函数把实参的值传送给被调函数的形参，从而实现主调函数向被调函数的数据传输。

关于函数的形参和实参,有以下几点说明。

(1) 形参变量只有在被调用时才分配内存单元,调用结束时释放所分配的内存单元。因此,形参只有在函数内部有效。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,在进行函数调用时,它们都必须已经具有确定的值,以便把这些值传送给形参。

(3) 实参和形参在数量、类型,以及顺序上应严格一致,否则会出现类型不匹配的错误。

(4) 函数调用中发生的数据传输是单向的。即只能把实参的值传送给形参,而不能把形参的值反向地传送给实参。这意味着在函数调用过程中,形参的值发生改变,而实参中的值不会变化。

【例 3-4】 城市有大量出租汽车运营,其按行车里程收费的方式简单易于理解。首先在固定里程内的部分(三千米以内)按起步价 10 元收取,超出的部分则按每千米 1.5 元收取。程序要求输入行驶里程,经过计算后输出收费金额。

问题分析

设计子函数 CalcFee 实现根据里程数进行计费的功能,而在 main 主函数中实现读取用户输入的里程数,并调用子函数 CalcFee 进行计费运算,在调用时需要向子函数传递参数:里程数。子函数根据一定的计算规则进行计算,完成后将结果返回给主函数,主函数输出结果。

编写程序

```
#include<iostream>
#include<math.h>
using namespace std;
double CalcFee(double miles);
int main(){
    double miles;
    cout<<"请输入行驶里程:"<<endl;
    cin>>miles;
    cout<<"应该收取的费用为: "<<CalcFee(miles)<<endl;
}
double CalcFee(double miles) {
    double money=0;
    if(miles<=3){
        money=10;
    }
    else if(miles>3){
        money=10+ceil(miles-3)*1.5;
    }
    return money;
}
```

运行结果

```
请输入行驶里程:
4↙
应该收取的费用为:11.5
```

3.5 函数的调用

函数的调用,是使程序转而去执行其他函数体。函数不能嵌套定义,但是函数之间允许相互调用,也允许嵌套调用,习惯上把调用者称为主调函数。函数还可以自己调用自己,称为递归调用。main 函数是主函数,它可以调用其他函数,而不允许被其他函数调用。因此,程序的执行总是从 main 函数开始,完成对其他函数的调用后再返回 main 函数,最后由 main 函数结束整个程序。一个 C++程序必须有且只有一个主函数 main。

在程序中往往通过对函数的组合调用来执行各种具体功能,函数调用的一般形式如下。

无参函数的调用格式:**函数名( )**

有参函数的调用格式:**函数名(实际参数表)**

对无参函数调用时不需要传递参数。有参函数中实际参数表的参数可以是常量、具有值的变量或表达式,各实参之间要用逗号进行分隔。

3.5.1 函数的调用方式

根据函数调用在程序中出现的位置,函数的调用方式可以分为以下三种。

(1) 函数表达式:函数作为表达式中的一项出现在表达式中,以函数返回值参与表达式的运算。这种方式要求函数有返回值。

例如:z=getMax (x,y)是一个赋值表达式,把 getMax 的返回值赋予变量 z。

(2) 函数语句:函数调用的一般形式加上分号即构成函数语句。

例如:printf ("%d",a);和 scanf ("%d",&b);都是以函数语句的方式调用函数。

(3) 函数实参:函数作为另一个函数调用的实参出现。这种情况是把该函数的返回值作为实参进行传送,因此要求该函数必须有返回值。

例如:printf("%d",getMax (x,y));把 getMax 调用的返回值作为 printf 函数的实参来使用。

使用函数调用语法来解决一个实际问题:快速计算搬运费用。因为要解决的问题要比前文讲述的函数实例复杂,因此用到多个子函数将其简化。

【例 3-5】 某搬家运输公司要求设计一个能快速计算搬运费用的程序。搬家费用由两部分组成:第一项是人力搬运费,每 100 千克收费 40 元;第二项是运输费用,包含 50 元的固定费用和每千米 10.5 元的里程费用。程序首先提示用户输入家具的重量和运输距离,然后程序显示人力费用、运输费用和总费用。

问题分析

可以将程序设计成顺序结构形式,如图 3.2 所示。

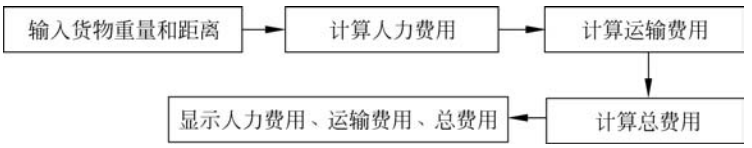

图 3.2 程序流程

可以在程序的主函数中用输入语句(如 cin 语句)输入货物重量和距离,然后依次计算人力费用、运输费用、总费用和显示人力费用、运输费用、总费用等。计算人力费用使用函数 Calc_Labor()完成;计算运输费用使用函数 Calc_Travel()完成;显示人力费用、运输费用和总费用等使用函数 Display_Charges()完成。

编写程序

```
1  //This program estimates the charge for moving furniture.
2  //It uses functions to do calculations and output the results.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6  double Calc_Labor(int);
7  double Calc_Travel(int);
8  void Display_Charges(double, double, double);
9  int main()
10 {
11   int weight;                    //Estimated weight
12   int distance;                  //Estimated distance
13   double labor_charge;           //Estimated labor charge
14   double travel_charge;          //Estimated travel charge
15   double total_charge;           //Estimated total charge
16   cout << setprecision(2)
17        << setiosflags(ios::fixed)
18        << setiosflags(ios::showpoint);
19   cout << "Enter the weight in kilograms: ";
20   cin >> weight;
21   cout << endl;
22   cout << "Enter the distance in kilometers: ";
23   cin >> distance;
24   labor_charge = Calc_Labor(weight);
25   travel_charge = Calc_Travel(distance);
26   total_charge = labor_charge + travel_charge;
27   Display_Charges(labor_charge, travel_charge, total_charge);
28   cout << endl;
29   return 0;
30 }
31 double Calc_Labor(int weight)
32 {
33   const double LABOR_RATE = 40;
34   double labor_charge;
```

```
35    labor_charge = (weight / 100) * LABOR_RATE;
36    return labor_charge;
37 }
38 double Calc_Travel(int distance)
39 {
40    const double CHARGE_PER_KM = 10.50;
41    const double FLAT_TRAVEL_COST = 50.00;
42    double travel_charge;
43    travel_charge = FLAT_TRAVEL_COST + distance * CHARGE_PER_KM;
44    return travel_charge;
45 }
46 void Display_Charges(double labor_charge, double travel_charge, double total_charge)
47 {
48    cout << endl;
49    cout << "The estimated charges are as follows:" << endl << endl;
50    cout << "Estimated Labor: " << setw(9) << labor_charge << endl;
51    cout << "Estimated Travel: " << setw(9) << travel_charge << endl;
52    cout << " ---------------------------- " << endl;
53    cout << "Total Estimate: " << setw(9) << total_charge << endl;
54 }
```

运行结果

```
Enter the weight in kilograms:1000 ↙
Enter the distance in kilometers:30 ↙
The estimated charges are as follows:
Estimated Labor:400.00
Estimated Travel:365.00
 ----------------------------
Total Estimate:765.00
```

实例首先声明了三个函数：Calc_Labor()函数返回人力费用，返回值声明为双精度(double)类型，声明一个整型参数用于以千克为单位的货物重量作为函数输入。Calc_Travel ()函数返回运输费用，返回值同样声明为双精度(double)类型，声明一个整型参数用于以千米为单位的运输距离作为函数输入。Display_Charges()函数有三个参数，分别是人力费用、运输费用和总费用，类型均为双精度(double)类型，因为该函数没有返回值，所以返回值类型为 void。

程序是一个顺序结构程序，首先用 cin 语句要求用户输入家具的重量和运输距离；然后赋值语句 labor_charge = Calc_Labor(weight)；调用执行函数 Calc_Labor()，计算出基于家具重量作参数的人力费用，该费用作为函数返回值返回到 main()函数，并赋值给 labor_charge 变量；赋值语句 travel_charge = Calc_Travel(distance)；调用执行函数 Calc_ Travel ()，计算出基于运输距离作参数的运输费用，该费用作为函数返回值返回到 main()函数，并赋值给 travel_charge 变量；赋值语句 total_charge = labor_charge + travel_charge；调用执行函数计算人力费用和运输费用之和并将结果赋值给 total_charge 变量；函数调用语句 Display_Charges(labor_charge, travel_charge, total_charge)；的三

个输入参数分别是人力费用、运输费用和总费用，用于在显示器分别显示人力费用、运输费用和总费用。

3.5.2 嵌套调用

C++语言不允许函数嵌套定义，即在一个函数中不能完整地包含另一个函数。各函数之间是平行和独立的，不存在上一级函数和下一级函数的问题，但是允许在一个函数的定义中出现对另一个函数的调用，这就是函数的嵌套调用，即在被调函数中又调用其他函数。需要注意的是，在调用函数之前，需要对每一个被调用的函数作声明(除非定义在前，调用在后)。其关系如图 3.3 所示。

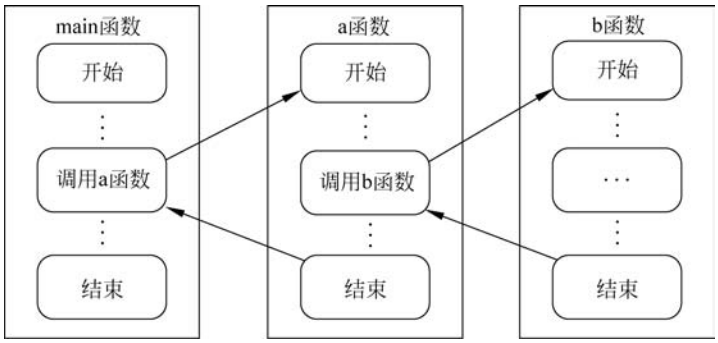

图 3.3 函数的嵌套调用

图 3.3 表示两层嵌套的情形。其执行过程是：执行 main 函数中调用 a 函数的语句时，即转去执行 a 函数，在 a 函数中调用 b 函数时，又转去执行 b 函数，b 函数执行完毕返回 a 函数的断点继续执行，a 函数执行完毕返回 main 函数的断点继续执行。

3.5.3 递归调用

一个函数在它的函数体内调用其自身称为递归调用，该函数称为递归函数。在递归调用中，主调函数同时也是被调函数。执行递归函数将反复调用其自身，每调用一次就进入新的一层。

递归调用有直接递归调用和间接递归调用两种形式。直接递归即在函数中出现调用函数本身。例如，用直接递归求解斐波那契数列第 n 项。斐波那契数列的第一项和第二项是 1，而后面每一项是前二项之和，即 1,1,2,3,5,8,13……

```
long fib(int x)
{
if(x > 2)
        return(fib(x - 1) + fib(x - 2));          //直接递归
    else
        return 1;
}
```

间接递归调用是指函数中调用了其他函数，而其他函数又调用了原来的函数。例如，

下面的代码定义两个函数，它们构成了间接递归调用。

```
int fnl(int a)
{
int b;
    b = fn2(a + 1);                          //间接递归
    //...
}
int fn2(int s)
{
    int c;
    c = fnl(s - 1);                          //间接递归
    //...
}
```

示例中，fnl()函数调用了 fn2()函数，而 fn2()函数又调用了 fnl()函数。它们构成了间接递归调用。

采用递归方法来解决问题，必须符合以下三个条件。

(1) 可以把要解决的问题转化为一个新问题，而这个新问题的解决方法仍与原来的解决方法相同，只是所处理的对象有规律地递增或递减。即解决问题的方法相同，调用函数的参数不同(有规律地递增或递减)，如果没有规律则不能适用递归调用。

(2) 可以应用这个转化过程使问题得到解决。使用其他的方法比较复杂或难以解决，而使用递归的方法可以较好地解决问题。

(3) 必须有一个明确的结束递归的条件，保证能够在适当的地方结束递归调用，否则可能导致系统崩溃。

下面我们应用递归调用的方法来估算一下基金本金的发展速度。

【例 3-6】 某基金公司初始资金 5000 万，预计每年收益是 25%，每年固定费用支出 150 万(年终提取)，管理方收益提成 20%(也就是年终净收益 1000 万，基金管理人提取 20%)，请计算该基金第 5 年的资金额是多少？

问题分析

可以将程序设计成函数的递归调用形式。第 5 年的资金额取决于第 4 年的资金额，而第 4 年的资金额取决于第 3 年的资金额，而第 3 年的资金额取决于第 2 年的资金额，依次类推，已知第 1 年的资金额是 5000 万。

```
money(5) = money(4) * (1 + 0.25) - 150 - money(4) * 0.25 * 0.2
money(4) = money(3) * (1 + 0.25) - 150 - money(3) * 0.25 * 0.2
money(3) = money(2) * (1 + 0.25) - 150 - money(2) * 0.25 * 0.2
money(2) = money(1) * (1 + 0.25) - 150 - money(1) * 0.25 * 0.2
money(1) = 5000
```

可以用式子表述如下。

```
money (n) = 5000                                          (n = 1)
money (n) =  money (n - 1)  * (1 + 0.25) - 150 - money(n - 1) * 0.25 * 0.2      (n > 1)
```

可以看到，当 $n>1$ 时，求第 n 年的资金的公式是相同的，因此可以用一个函数表示

上述关系。

编写程序

```
//This program estimates the money of the fund after n years.
//It uses recursion functions to do calculations.
#include <iostream>
using namespace std;
int money(int n)
{
    int m;
    if (n == 1)
        m = 5000;
    else
        m = money(n - 1) * (1 + 0.25) - 150 - money(n - 1) * 0.25 * 0.2;
    return m;
}
int main()
{
    int n = 5;
    cout <<"The money of the fund after "<< n <<" years is "<< money(n)<< endl;
    return 0;
}
```

运行结果

```
The money of the fund after 5 years is 9562
```

main 函数中除了 return 语句外,还有一个定义语句和 cout 语句。整个问题的求解全靠一个函数调用 money(*n*)来解决。函数调用的过程如图 3.4 所示。从图 3.4 中可以

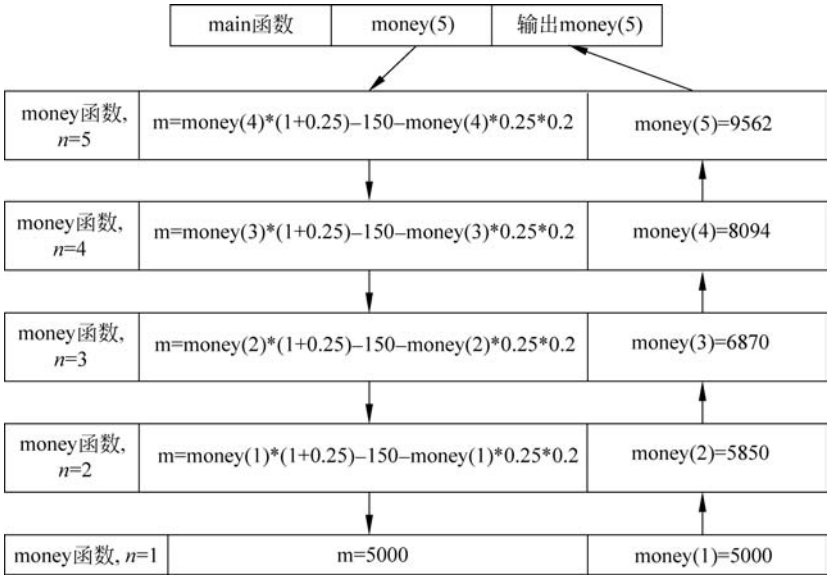

图 3.4　money 函数的递归调用过程

看出，money 函数共调用 5 次，即 money(5)、money(4)、money(3)、money(2)、money(1)。其中 money(5)是在 main 函数中调用的，其余 4 次是在 money 函数中调用的，即递归调用 4 次。请读者仔细分析调用的过程。强调说明的是：在某一次调用 money 函数时并不是立即得到 money(*n*)的值，而是一次又一次地进行递归调用，直到 money(1)时才有确定的值，然后再递推出 money(2)、money(3)、money(4)、money(5)。

3.6 局部变量和全局变量

学习函数的形参变量时已知，形参变量只在被调用期间才分配内存，调用结束后立即释放。这表明形参变量只在函数内有效，离开该函数则不再使用。这种变量有效性的范围称为变量的作用域。其实不仅是形参变量，C++语言中所有的量都有自己的作用域，变量声明的方式位置不同，其作用域也不同。C++语言中的变量，按作用域范围可分为两种，即局部变量和全局变量。

3.6.1 局部变量

局部变量也称为内部变量。局部变量是在函数内作定义说明的，其作用域仅限于函数内，离开该函数后则不再使用这种变量。示例如下。

```
int fun1(int a){
    int b,c;
    //...
}
float fun2(float x,float y){
    float z;
    //...
}
int main(){
    int m,n;
    {
        int p,q;
        //...
    }
}
```

在函数 fun1 内定义三个变量，a 为形参，b、c 为一般变量。在 fun1 的范围内 a、b、c 有效，或者说 a、b、c 变量的作用域限于 fun1 内。同理，x、y、z 的作用域限于 fun2 内。m、n 的作用域限于 main 函数内，p、q 变量的作用域限于其声明所在位置的大括号内。

关于局部变量的作用域需要注意如下几点。

(1) 主函数中定义的变量也只能在主函数中使用，同时主函数中也不能使用其他函数中定义的变量。

(2) 形参变量是属于被调函数的局部变量，实参变量是属于主调函数的局部变量。

(3) 在不同的函数中可以使用相同的变量名，它们代表不同的对象，互不干扰。

(4) 在复合语句中定义的变量只在本复合语句范围内有效。例如：

```
int main(){
    int x,y;
    //...
    {
        int b;
        b = x + y;
        //...                                    /* b 作用域到此为止 */
    }
    //...                                        /* x,y 作用域可以到此 */
}
```

无论局部变量是否显式初始化，都只有当定义它们的程序块被调用时(即执行时)才分配空间，声明或定义时并不分配。对局部变量需要显式地进行初始化，否则它们的初始值是不确定的。

3.6.2 全局变量

全局变量也称为外部变量，它是在函数外部定义的变量。它不属于某一个函数，而是属于整个源程序文件，其有效范围为从定义变量的位置开始到本源文件结束。全局变量提供函数间数据联系的渠道。

例如：

```
#include <iostream>
using namespace std;
int a,b;                                         /* 外部变量 */
int fun1(){                                      /* 函数 fun1 */
    //...
}
int x,y;                                         /* 外部变量 */
int fun2(){                                      /* 函数 fun2 */
    //...
}
int main(){                                      /* 主函数 */
    //...
    return 0;
}
```

从上例可以看出 a、b、x、y 都是在函数外部定义的外部变量，都是全局变量。但 x、y 定义在函数 fun1 之后，而在 fun1 内没有对 x、y 的说明，所以它们在 fun1 内无效。a、b 定义在源程序最前面，因此在 fun1、fun2 及 main 内不加说明也可使用。

我们来看一个计算商店销售额的示例。

【例 3-7】 商店中通常进行多种商品的销售，每过一段时间需要进行一次商品单品销售额和总销售额的统计。设计程序进行该项任务的计算，要求输入 N 种商品，分别计算 N 种商品的销售额和 N 种商品的总销售额。

问题分析

定义全局变量 salesAmount 来累计总销售额，定义子函数 calcSingleProductSales 来计算单品销售额，该函数需要两个参数，单价 price 和销售量 num 来完成计算任务。在 main 主函数中，完成接收键盘输入的商品数目、每种商品的单价和销售量，之后调用子函数 calcSingleProductSales，并将商品单价和销售量传递给子函数，子函数进行计算。计算任务完成后结果返回给 main 主函数，主函数进行结果的显示输出。

编写程序

```
#include <iostream>
using namespace std;
double salesAmount = 0;
double calcSingleProductSales(double price,double num);
int main(){
    int N;
    double productSales;
    double price,num;
    cout <<"请输入商品数量:"<< endl;
    cin >> N;
    for (int i = 1;i <= N;i++){
        cout <<"请输入商品单价和销售量:"<< endl;
        cin >> price >> num;
        productSales = calcSingleProductSales(price,num);
        cout <<"商品"<< i <<"的销售额为"<< productSales << endl;
        salesAmount += productSales;
    }
    cout <<"所有商品总的销售额为"<< salesAmount << endl;
}
double calcSingleProductSales(double price,double num) {
    double sales;
    sales = price * num;
    return sales;
}
```

运行结果

```
请输入商品数量:
2↙
请输入商品单价和销售量:
10.0 5↙
商品 1 的销售额为 50
请输入商品单价和销售量:
5.0 3↙
商品 2 的销售额为 15
所有商品总的销售额为 65
```

【例 3-8】 分别输入长方形的长和宽,求其周长与面积。

编写程序

```
#include <iostream>
using namespace std;
float circle;
float getArea(float a,float b) {
    float s;
    circle = 2 * (a + b);
    s = a * b;
    return s;
}
int main(){
    float l,w,area;
    cout <<"请输入长方形的长和宽:"<< endl;
    cin >> l >> w;
    area = getArea(l,w);
    cout <<"该长方形的周长为: "<< circle <<",面积为: "<< area << endl;
}
```

运行结果

```
请输入长方形的长和宽:
3 4↙
该长方形的周长为:14,面积为:12
```

本程序中定义了一个外部变量 circle,用来存放周长,其作用域为整个源程序。函数 getArea 用来求长方形的周长和面积,函数的返回值为面积 area。由主函数完成长和宽的输入及结果输出。由于 C++语言规定函数返回值只有一个,当需要增加函数的返回数据时,用外部变量是一种较好的方式。本例中,如不使用外部变量,在主函数中就不可能求得周长和面积两个值。而采用了外部变量,在函数 getArea 中求得的 circle 值在 main 中仍然有效。因此全局变量是实现函数之间数据通信的有效手段。

全局变量显式初始化时或未初始化时,位于程序映像中不同的分区:已初始化的全局变量是可执行模块的一部分;未初始化的全局变量则不是可执行模块的一部分,只有当定义它们的程序被调用时(即执行时)才分配空间,声明或定义时并不分配。未初始化的全局变量在运行时被初始化为 0。

如果同一个源文件中,外部变量与局部变量同名,则在局部变量的作用范围内,外部变量被“屏蔽”,即它不起作用。

例如:

```
#include <iostream>
using namespace std;
int x = 10;
int main(){
    int x = x;
```

```
    cout << x << endl;
    return 0;
}
```

以上程序定义一个全局变量 x,并初始化为 10,同时在 main 函数内部定义一个局部变量 x。在 main 函数内部对局部变量 x 进行赋值 x = x,因为局部变量会屏蔽全局变量,此时赋值语句中两个 value 都是局部变量,因为 value 还没有初始化,所以其值为随机值,而不是 10。

3.7 内部函数和外部函数

函数本质上是全局的,因为函数定义后是要被其他函数调用的,但是,也可以限定函数不能被其他文件调用。根据函数能否被外部源文件调用,可以将函数分为内部函数和外部函数。

3.7.1 内部函数

内部函数是只能被本文件中的函数所调用的函数。在定义内部函数时,在函数名和函数类型的前面加 static。

定义形式如下。

```
static 类型标识符 函数名(形参表)
```

例如:

```
static int fun(int a,int b)
```

内部函数又称为静态函数。使用内部函数,可以使函数只局限于其所在文件。即使在不同的文件中有相同名称的内部函数,也互不干扰。程序团队开发人员可以分别编写各自不同的函数,而不必担心所用函数是否会与其他文件中的函数同名。通常把只能由同一文件使用的函数和外部变量放在一个文件中,在它们前面加 static 使之局部化,这样其他文件便不能引用。

3.7.2 外部函数

若在定义函数时在函数首部的最左端加上关键字 extern,则表示该函数是外部函数,可供其他文件调用。例如函数首部可以写为:

```
extern int fun (int a, int b)
```

这样,函数 fun 就可以被外部文件调用。此外,若在定义函数时省略 extern,则默认为外部函数。

我们来看一个计算薪资的示例。

【例 3-9】 某企业有部分工人是按照时薪制聘任的,每个月末需要根据与其约定的时薪和工作小时数计算应付薪资,设计程序,使用外部子函数实现时薪制的薪资计算。

问题分析

定义一个外部子函数 CalcSalary 来计算薪资,该函数需要两个参数,时薪 hourPay 和工作小时数 hours,来完成计算任务。该函数需要被外部文件调用,并将结果返回给调用者。在另一文件中的 main 主函数接收键盘输入的时薪和工作小时数,之后调用子函数 CalcSalary,并将时薪和工作小时数传递给子函数,子函数进行计算。计算任务完成后结果返回给 main 主函数,主函数进行结果的显示输出。

编写程序

```
1  /******* file1.cpp(文件 1) ******* /
2  #include < iostream >
3  using namespace std;
4  int main()
5  {
6      extern double CalcSalary(double hourPay,double hours);
7                                                          //声明调用其他文件中定义的函数
8      double a,b;
9      cout <<"请输入时薪和工作小时数: "<< endl;
10     cin >> a >> b;
11     cout <<"该工人的工资为: "<< CalcSalary(a,b)<< endl;
12     return 0;
13 }
14 /******* file2.cpp(文件 2) ******* /
15 #include < iostream >
16 using namespace std;
17 double CalcSalary(double hourPay,double hours)
18 {
19     double salary = hourPay * hours;
20     return salary;
21 }
```

运行结果

```
请输入时薪和工作小时数:
10 12
----------------------------
该工人的工资为: 120
```

在需要调用外部函数的源文件中,需要用 extern 声明所用的函数是外部函数。

3.8 综合实例

【例 3-10】 编写程序,实现搬家公司计算客户搬家的总费用。搬家总费用由人工费和搬运费两个因素决定。首先,搬运每 100 千克的家具收取人工费 40 元。其次,50 元为搬家的固定费用,再加上每千米 10.50 元的搬运费。程序首先提示用户输入家具的重量

和搬运的距离，然后计算搬运费，最后显示人工费、搬运费和搬家总费用。

问题分析

采用的算法用伪代码描述如下。

```
提示用户输入家具的重量 weight 和搬运的距离 distance
计算人工费 labor_charge
计算搬运费 travel_charge
计算搬家总费用 total_charge
显示人工费 labor_charge,搬运费 travel_charge 和搬家总费用 total_charge
```

程序计算人工费 labor_charge 和搬运费 travel_charge，使用 Calc_Labor()函数和 Calc_Travel()函数来实现。同样，显示结果使用函数 Display_Charges()来实现。

编写程序

```
#include <iostream>
#include <iomanip>
using namespace std;
double Calc_Labor(int);
double Calc_Travel(int);
void Display_Charges(double, double, double);
int main()
{
        int weight, distance;
        double labor_charge, travel_charge, total_charge;
        cout << "请输入家具的重量(千克): ";
        cin >> weight;
        cout << "请输入搬运的距离(千米): ";
        cin >> distance;
        labor_charge = Calc_Labor(weight);
        travel_charge = Calc_Travel(distance);
        total_charge = labor_charge + travel_charge;
        Display_Charges(labor_charge, travel_charge, total_charge);
        return 0;
}
double Calc_Labor(int weight)
{
        const double LABOR_RATE = 40;
        double labor_charge;
        labor_charge = (weight / 100) * LABOR_RATE;
        return labor_charge;
}
double Calc_Travel(int distance)
{
        const double CHARGE_PER_KM = 10.50;
        const double FLAT_TRAVEL_COST = 50.00;
        double travel_charge;
        travel_charge = FLAT_TRAVEL_COST + distance * CHARGE_PER_KM;
        return travel_charge;
```

```
35 }
36 void Display_Charges(double labor_charge, double travel_charge, double total_charge)
37 {
38        cout << "计算出的搬家费用如下: " << endl;
39        cout << "人工费: " << labor_charge << endl;
40        cout << "搬运费: " << travel_charge << endl;
41        cout << "---------------------------------------" << endl;
42        cout << "搬家总费用: " << total_charge << endl;
43 }
```

运行结果

```
请输入家具的重量(千克): 7400 ↙
请输入搬运的距离(千米): 230 ↙
计算出的搬家费用如下:
人工费: 2960.00
搬运费: 2465.00
---------------------------------------
搬家总费用: 5425.00
```

首先声明了程序使用的 Calc_Labor()、Calc_Travel()和 Display_Charges()三个函数。Calc_Labor()计算人工费,所以声明该函数返回一个 double 类型的值,因为重量是以千克为单位输入,所以其参数是 int 类型。Calc_Travel()函数也返回一个 double 类型的值。因为搬运距离是 int 类型,所以其参数是 int 类型。最后,Display_Charges()包含三个参数,即人工费 labor_charge、搬运费 travel_charge 和搬家总费用 total_charge。因为 Display_Charges()没有返回值,所以返回类型为 void。

然后程序提示用户输入家具的重量和搬运的距离。赋值语句 labor_charge = Calc_Labor(weight);调用函数 Calc_Labor(),该函数的功能是根据家具的重量计算人工费。在 main()函数中家具的重量 weight 的值,作为实参传递给形参 weight 计算人工费,并用变量 labor_charge 的值返回人工费。

下一个赋值语句 travel_charge = Calc_Travel(distance);调用 Calc_Travel(),该函数的功能是根据搬运的距离计算搬运费。同样,在 main()中将变量 distance 的值传递给 Calc_Travel()函数的参数 distance 计算搬运费,并用变量 travel_charge 的值返回搬运费。变量 total_charge 计算人工费 labor_charge 和搬运费 travel_charge 的总和。

最后程序将变量 labor_charge、travel_charge 和 total_charge 的值传递给 Display_Charges()函数相应的参数,显示输出。

【例 3-11】 编写一个程序,实现银行为使用自动取款机(ATM)的客户提供个人贷款业务,并显示客户个人的贷款利息、贷款总额和每月的还贷金额。

首先,程序应该显示一段问候语,内容如下。

```
************************************************************
欢迎使用个人贷款程序。
本程序将显示个人的贷款利息、贷款总额和每月的还贷金额。
```

请按以下指令认真操作。

****

显示问候语后，程序要向用户询问贷款金额（本金）、贷款的年利率（百分比）和贷款年数（期限）。因为程序只用于处理个人贷款，因此，贷款金额不能少于 1000 元，且不能超过 20 000 元。如果贷款金额超出范围，程序则显示相应的提示信息并终止执行。按照我国央行基准利率规定，贷款的年利率不能超过 18.7%。如果输入的年利率大于 18.7%，程序则显示相应的提示信息并终止执行。最后，贷款的最长期限只能在五年之内。如果输入的贷款期限大于五年，程序则显示相应的提示信息并终止执行。

如果所有的输入数据有效，程序计算贷款利息、贷款的总金额，以及每月的还贷金额，使用以下公式计算。

贷款利息＝贷款金额×（年利率/100）×贷款年数

贷款总额＝贷款金额＋贷款利息

每月的还贷金额＝贷款总额/（12×贷款年数）

由于用户输入的年利率是百分比形式，所以在计算贷款利息的公式中需要将年利率值除 100。如果用户输入年利率为 10.7，必须使用公式进行换算，即 10.7/100 = 0.107。

在计算每月的还贷金额公式中，表达式（12×还贷年数）是贷款的月份数。目的是把贷款总额按月还贷。

最后，程序显示用户输入的数据、贷款利息、贷款总额和每月的还贷金额。

问题分析

程序的设计较为简单，层次结构如图 3.5 所示。

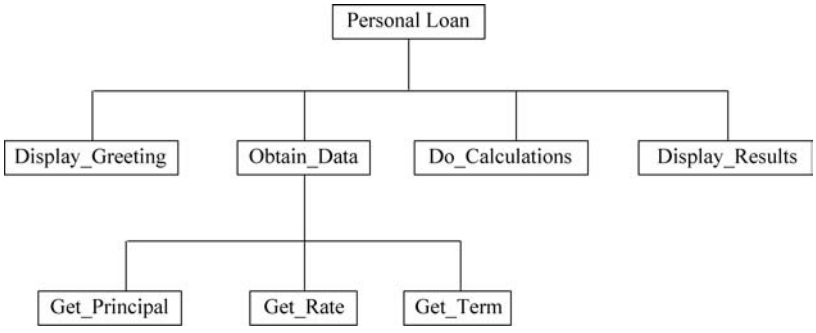

图 3.5 程序层次结构

使用 Display_Greeting 模块实现显示多行问候语，把 Obtain_Data 模块分为三个子模块，分别是 Get_Principal、Get_Rate 和 Get_Term，每个子模块实现相应的一个功能。这种每个输入值作为一个单独的子模块，是常见的一种编程方法。

因为涉及三个简单的计算公式，我们把 Do_Calculations 作为一个模块，这三个计算公式将出现在 main()中。同时，Display_Results 也作为一个模块，也将出现在 main()中。

采用的算法用伪代码描述如下。

```
Display_Greeting
Obtain_Data:
```

```
Get_Principal
Get_Rate
Get_Term
Do_Calculations
Display_Results
```

获取用户的输入数据的三个模块具有相似的结构。因此,只编写 Get_Principal 的伪代码,其他两个模块类似。

贷款金额不能少于 1000 或大于 20 000,使用嵌套 if 语句实现。

Get_Principal 的伪代码描述如下。

```
提示用户输入贷款金额 principal
 if (贷款金额 principal < 1000.00)
显示错误信息
退出程序
else
if (贷款金额 principal > 20000.00)
显示错误信息
退出程序
else
     返回贷款金额 principal
endif
endif
```

编写程序

```
1  #include <iostream>
2  #include <iomanip>
3  #include <cstdlib>
4  using namespace std;
5  void Display_Greeting();
6  double Get_Principal();
7  double Get_Rate();
8  int Get_Term();
9  int main()
10 {
11       double interest, principal, rate, total, monthly_payment;
12       int term;
13       Display_Greeting();
14       principal = Get_Principal();
15       rate = Get_Rate();
16       term = Get_Term();
17       interest = principal * (rate/100) * term;
18       total = principal + interest;
19       monthly_payment = total / (12 * term);
20       cout << "贷款年利率: " << rate << " %" << endl;
21       cout << "贷款期限: " << term << " Years" << endl;
22       cout << "贷款金额: " << principal << endl;
```

```
23        cout << "贷款利息: " << interest << endl;
24        cout << " ------------------------------------------------ " << endl;
25        cout << "贷款总额: " << total << endl;
26        cout << "每月的还贷金额: " << monthly_payment << endl;
27        return 0;
28 }
29 void Display_Greeting()
30 {
31      int position;
32      for (position = 1; position <= 61; ++position)
33          cout << " * ";
34      cout << endl;
35      cout << "\t\t 欢迎使用个人贷款程序。" << endl;
36      cout << "本程序将显示个人的贷款利息、贷款总额和每月的还贷金额。";
37      cout << endl;
38      cout << "请按以下指令认真操作" << endl << endl;
39      for (position = 1; position <= 61; ++position)
40          cout << " * ";
41      cout << endl;
42 }
43 double Get_Principal()
44 {
45        const double MIN_PRINCIPAL = 1000.00;
46        const double MAX_PRINCIPAL = 20000.00;
47        double principal;
48        cout << "请输入贷款金额: ";
49        cin >> principal;
50        if (principal < MIN_PRINCIPAL)
51        {
52          cout << "对不起,我们不提供 1000 元以下的个人贷款。" << endl;
53          exit(1);
54        }
55        else
56          if (principal > MAX_PRINCIPAL)
57          {
58            cout << endl;
59            cout << "对不起,我们不提供 20000 元以上的个人贷款。" << endl;
60            cout << "请输入合理的贷款金额。\n";
61            exit(1);
62          }
63          else
64            return principal;
65 }
66 double Get_Rate()
67 {
68        const double MAX_RATE = 18.70;
69        double rate;
70        cout << "请输入当前的年利率(百分比): ";
71        cin >> rate;
```

```
72        if (rate > MAX_RATE)
73        {
74          cout << "对不起,年利率超出了我们的最大值。" << MAX_RATE << " % " << endl;
75          exit(1);
76        }
77        else
78          return rate;
79 }
80 int Get_Term()
81 {
82        const int MAX_TERM = 5;
83        int term;
84        cout <<"请输入贷款的期限: ";
85        cin >> term;
86        if (term > MAX_TERM)
87        {
88           cout << endl;
89           cout << "对不起,您的贷款期限不能超过" << MAX_TERM << "years" << endl;
90           exit(1);
91        }
92        else
93        return term;
94 }
```

运行结果

```
*****************************************************************
欢迎使用个人贷款程序。
本程序将显示个人的贷款利息、贷款总额和每月的还贷金额。
请按以下指令认真操作。
*****************************************************************
请输入贷款金额: 10000.00↙
请输入当前的年利率(百分比): 10.7↙
请输入贷款的期限: 4↙
贷款年利率: 10.70 %
贷款期限: 4Years
贷款金额: 10000.00
贷款利息: 4280.00
-----------------------------------------------
贷款总额: 14280.00
每月的还贷金额: 297.50
```

程序开始对 main()函数里用到的四个子函数进行了原型说明,由于 Display_Greeting()子函数只完成在屏幕显示信息的功能,所以不需要任何参数和返回值。其他三个子函数都不需要参数来完成其功能,只需要获得输入数据。如果用户输入的数据有效,则每个函数都把输入的数据返回给 main()函数。因此,每个函数返回的类型和用户输入数据的类型一致。因为贷款金额和年利率是小数,所以 Get_Principal()和 Get_Rate()子函数的返回值是 double 型。因为贷款期限是整数,所以 Get_Term()子函数的返回值

是 int 型。

main()函数首先调用 Display_Greeting()子函数显示问候语，接着调用 Get_Principal()、Get_Rate()和 Get_Term()子函数得到输入数据，并把相应的值赋值给变量 principal、rate 和 term。注意，用户输入的年利率是百分比年利率。

然后，利用问题描述里给出的计算公式计算贷款利息 interest、贷款总额 total 和每月的还贷金额 monthly_payment。

最后，用 7 条 cout 输出语句显示结果。

【例 3-12】 编写函数，由实参传递主调函数中存放的字符串首地址给该函数，统计此字符串中字母、数字、空格和其他字符的个数，并在主调函数中输入字符串以及输出结果。

问题分析

在主调函数中，某一字符串可以存放在一维字符数组中，该数组名字是存放字符串的一维数组首地址，是地址常量，所编写的处理字符串的函数，其形参定义为数组名，调用该函数时，把主调函数中的存放字符串的一维字符数组首地址，通过实参传递到该函数中，能够处理主调函数的字符串。

编写程序

```
#include <iostream>
using namespace std;
void judge(char a[]);
void main()
{
   const int size = 100;
   char a[size];
   cin.getline(a,size);
   judge(a);
}
void judge(char a[100])     //判断字符类型
{
   int letter = 0,number = 0,others = 0,i = 0;
   while(a[i]!= '\0')
   {
     if ((a[i]>= 'a'&&a[i]<= 'z')||(a[i]>= 'A'&&a[i]<= 'z')) letter++;   //统计字母个数
     else if (a[i]>= '0' && a[i]<= '9') number++;      //统计数字个数
          else others++;                               //统计其他字符个数
     i++;
   }
   cout <<"letter = "<< letter <<" number = "<< number <<" others = "<< others << endl;
}
```

运行结果

```
输入:!@#$%^QWERT    1234567
```

```
letter = 5 number = 7   others = 6
```

【例 3-13】 键盘输入两个整数,编写两个函数,分别求两个整数的最大公约数和最小公倍数,用主函数调用这两个函数,并输出结果。

编写程序

```
#include <iostream>
using namespace std;
int cdivisor(int,int);
int cmultiple(int,int,int);
void main()
{
  int x,y,d,m;
  cout <<"input two number: ";
  cin >> x >> y;
  d = cdivisor(x,y);
  m = cmultiple(x,y,d);
  cout <<"common divisor is "<< d << endl <<"common multiple is "<< m << endl;
}
int cdivisor(int x1,int y1)          //最大公约数
{
  int r,temp;
  if (x1 < y1)
  {
    temp = x1;
    x1 = y1;
    y1 = temp;
  }
  while(x1 % y1)                     //当较大数除以较小数余数等于 0 时,较小数为最大公约数
  {
    r = x1 % y1;
    x1 = y1;
    y1 = r;
  }
  return y1;
}
int cmultiple(int x2,int y2,int d1) //最小公倍数
{
return x2 * y2/d1;                   //两数相乘结果除以它们的最大公约数为最小公倍数
}
```

运行结果

```
input two number:6 8
common divisor is 2
common multiple is 24
```

【例 3-14】 键盘输入一个四位数,编写函数,要求输出这 4 个数字字符,且每两个数

字间空一空格。如输入1880,则输出“1 8 8 0”。

编写程序

```
#include <iostream>
#include <string>
using namespace std;
void outs(char a[]);
void main()
{
  const int size = 10;
  char a[size];
  cin.getline(a,size);
  outs(a);
}
void outs(char a[10])
{
  int i;
  if(strlen(a)<=4)
  {
    for(i=0;i<4;i++)
    cout<<a[i]<<" ";
  }
  else cout<<"input error."<<endl;
}
```

运行结果

```
1880
1 8 8 0
```

【例 3-15】 编写函数,将一个字符串的元音字母复制到另一个字符串并输出。

编写程序

```
#include <iostream>
#include <string>
using namespace std;
void scpy(char *,char *);
void main()
{
  const int size = 100;
  char a[size] = "Hello world";
  char b[size] = "Net";
  cout<<"a = "<<a<<"b = "<<b<<endl;
  scpy(a,b);
  cout<<"a = "<<a<<endl;
}
void scpy(char *p,char *q)
```

```
15 {
16   while( * q!= '\0')
17   {
18     if
19     ( * q == 'a'|| * q == 'A'|| * q == 'e'|| * q == 'E'|| * q == 'i'|| * q == 'I'|| * q == 'o'|| *
       q == 'O'|| * q == 'u'|| * q == 'U')
20      * p++ = * q;
21     q++;
22   }
23 }
```

运行结果

```
a = Hello world b = Net
a = eello world
```

【例 3-16】 编写一个判别素数的函数,主函数输出 100～200 的所有素数,每行只输出 5 个素数。

编写程序

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main()
5    {int f(int n);
6     int m,i,n = 0,flag;
7     for(m = 100;m <= 200;m++)
8       {flag = f(m);
9       if (flag) {cout << setw(5)<< m;n++; if (n % 5 == 0) cout << endl;}
10    }
11   cout << endl;
12   system("PAUSE");
13   return 0;
14 }
15 int f(int n)
16    {int i,flag;
17    for(i = 2;i <= n - 1;i++)
18        if (n % i == 0) break;
19    if (i == n) flag = 1;
20    else flag = 0;
21   return flag;
22 }
```

运行结果

```
101  103  107  109  113
```

```
127  131  137  139  149
151  157  163  167  173
179  181  191  193  197
199
```

【例 3-17】 编写一个函数，计算三角形的面积，公式如下。

$$s=\frac{1}{2}(a+b+c)$$

$$\text{area}=\sqrt{s(s-a)(s-b)(s-c)}$$

三角形的三边 a、b、c 的值由主函数输入。

编写程序

```
1  #include <iostream>
2  #include <cmath>                          //使用数学函数时要包含头文件 cmath
3  #include <iomanip>                        //使用 I/O 流控制符要包含头文件 iomanip
4  using namespace std;
5  int main( )
6  {  double f(double a,double b,double c);
7     double a,b,c;
8     cout <<"输入 a,b,c:";
9     cin >> a >> b >> c;
10    if (a > 0 && b > 0 && c > 0 && a + b > c && b + c > a && c + a > b)
11      {
12        cout << setiosflags(ios::fixed)<< setprecision(4);
13    // 指定输出的数包含 4 位小数
14        cout <<"三角形面积 = "<< f(a,b,c)<< endl;
15      }
16     else
17       cout <<"不是三角形!"<< endl;
18     return 0;
19   }
20 double f(double a,double b,double c)
21 {  double s,area;
22    s = (a + b + c)/2;
23    area = sqrt(s * (s - a) * (s - b) * (s - c));
24    return area;
25 }
```

运行结果

```
输入 a,b,c: 4  5  6
三角形面积 = 9.9216
```

【例 3-18】 使用递归法编写函数，主函数调用函数并输出 3!+5!+8!。

编写程序

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

```
3  int main( )
4  {  long f(int n);
5     long s;
6     s = f(3) + f(7) + f(8);
7     cout <<"3! + 5! + 8!= "<< s << endl;
8     return 0;
9  }
10 long f(int n)
11 {  if (n == 1) return 1;
12    else return f(n - 1) * n;
13 }
```

运行结果

```
3!+ 5!+ 8!= 45366
```

【例 3-19】 使用递归法求 n 阶勒让德多项式的值,公式如下。

$$p_n(x)=\begin{cases}1 & (n=0)\\ x & (n=1)\\ \dfrac{(2n-1)x-p_{n-1}(x)-(n-1)p_{n-2}(x)}{n} & (n>1)\end{cases}$$

编写程序

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main( )
5  {  double p(int n,int x);
6     long s = 0;
7     int i,x,n;
8     cout <<"输入 n,x:";
9     cin >> n >> x;
10    cout << n <<"阶的结果 = "<< p(n,x)<< endl;
11    return 0;
12 }
13 double p(int n,int x)
14 {  if (n == 0) return 1;
15    else  if (n == 1) return x;
16        else return ((2 * n - 1) * x - p(n - 1,x) - (n - 1) * p(n - 2,x))/n;
17 }
```

运行结果

```
输入 n,x:3  4
3 阶的结果 = 2.83333
```

【例 3-20】 使用递归法求 $f(x)=\sum_{i=0}^{n} i^2$，n 的值由主函数输入。

编写程序

```
#include <iostream>
#include <cmath>
using namespace std;
int main( )
{   long f(int n);
    long s = 0;
    int i,n;
    cout <<"输入一个正整数 n:";
    cin >> n;
    cout <<"sum = "<< f(n)<< endl;
    return 0;
}
long f(int n)
{   if (n == 0) return 0;
    else return f(n - 1) + pow(n,2);
}
```

运行结果

```
输入一个正整数 n:5 sum = 55
```

【例 3-21】 中国银联和 VISA、万事达卡(MasterCard)都是国际信用卡组织，都负责信用卡业务的展开。不同的是，中国银联(全称中国银联股份有限公司)作为中国的银行卡联合组织，其业务主要在中国。VISA、MasterCard 作为全球领先的支付公司，因能提供全球消费者一个更便利与更有效率的金融支付环境而遍及全球。我国改革开放以来，越来越多的人到国内和国外旅游。境内(或境外某个地区)旅游使用中国银联信用卡即可，如果是境外全球旅游使用 VISA、MasterCard 信用卡比较方便。虽然目前中国人均银行卡持有率、银行卡消费占社会零售总额比例等指标低于发达和中等发达市场，但改革开放以来，中国经济的持续发展、个人收入水平提高、个人金融需求全面增加及结算观念转变等因素，将推动中国的银行卡业务步入一个高速增长阶段，中国已经成为银行卡产业全球发展潜力最大的国家。中国银行卡产业多年的持续不断地创新，使社会支付体系更趋完善。

问题描述：求出斐波那契数列前 40 个数，输出时每行 5 个数。这个数列有如下特点：第 1 个数为 1，第 2 个数为 1，从第 3 个数开始，该数是其前面两个数的和。

$$F_1=1 \quad (n=1)$$

$$F_2=1 \quad (n=2)$$

$$F_n=F_{n-1}+F_{n-2} \quad (n\geqslant 3)$$

要求：采用局部变量或全局变量编程，现在已经编写了下列程序，但是此程序运行结果错误。源程序及错误的程序运行结果如下所示。

(1) 分析程序运行输出错误结果的原因,找出使程序出错的语句。

(2) 若在 f 函数中只允许改变一个语句来得到正确的运行结果,正确的语句是什么?

(3) 若不改写程序中的所有语句,只允许改变其中一个语句的位置来得到正确的运行结果,这个语句应移动到程序的哪个位置?

编写程序

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main()
5  {long f(int);
6   int n=40,i;
7   for(i=1;i<=n;i++)
8   {cout<<setw(10)<<f(i);
9    if (i%4==0) cout<<endl;}
10  cout<<endl;
11  return 0;
12 }
13
14 long f(int n)
15 {long f1=1,f2=1;
16  long f3;
17  if (n==1) return 1;
18  else if (n==2) return 1;
19  else {f3=f1+f2;f1=f2;f2=f3;return f3;}
20 }
```

运行结果(错误的)

```
1         1         2         2
2         2         2         2
2         2         2         2
2         2         2         2
2         2         2         2
2         2         2         2
2         2         2         2
2         2         2         2
2         2         2         2
2         2         2         2
```

分析解答:

(1) 第 15 行 long f1=1,f2=1; 是错误的,相当于语句 auto long f1=1,f2=1;。

第 15 行中的 f1 和 f2 是 auto 存储型的局部变量,表示 f1 和 f2 两个动态局部变量只在 f 函数中有效且没有存储值的能力,程序每次执行 f 函数时,f1 和 f2 都进行初始化且取值为 1,所以程序运行的结果是错误的。

```
15    {long f1 = 1,f2 = 1;     //相当于 auto long f1 = 1,f2 = 1; 定义了动态局部变量
```

(2) 将第 15 行改为 static long f1＝1,f2＝1;。

第 15 行中的 f1 和 f2 是 static 存储型的局部变量,f1 和 f2 两个静态局部变量只在 f 函数中有效且有存储最新值的记忆能力,第 1 次执行 f 函数时,f1 和 f2 进行初始化且取值为 1,以后再次执行 f 函数时,f1 和 f2 都取已存储的最新的值。

```
15    {static long f1 = 1,f2 = 1;     // 定义了静态局部变量
```

运行程序正确的输出结果:

```
       1         1         2         3
       5         8        13        21
      34        55        89       144
     233       377       610       987
    1597      2584      4181      6765
   10946     17711     28657     46368
   75025    121393    196418    317811
  514229    832040   1346269   2178309
 3524578   5702887   9227465  14930352
24157817  39088169  63245986 102334155
```

(3) 将第 15 行的 long f1＝1,f2＝1;语句移到第 13 行 long f1＝1,f2＝1;第 15 行语句只留下左括号{。

第 13 行是在 f 函数外面定义的变量是全局变量,f1 和 f2 全局变量从定义开始直至程序结束都有效且有存储最新值的记忆能力,第 1 次执行 f 函数时,f1 和 f2 进行初始化且取值为 1,以后再次执行 f 函数时,f1 和 f2 都取存储的最新的值。运行结果正确。

```
13  long f1 = 1,f2 = 1;      // 定义了全局变量
14  long f(int n)
15  {
```

本章小结

在软件开发过程中,通常会将大的软件功能进行分解,分解后的实现则依靠函数。在函数实现后,通过对函数的组合调用来实现具体的功能。

函数有两种,即标准函数和自定义函数。标准函数由系统提供,而自定义函数则是用户自己编写用于实现特定功能的代码块。

函数的定义包括为函数命名,指定函数返回值类型和形参类型,以及编写函数体等。

函数的声明是把函数名、函数类型以及形参类型、个数和顺序通知编译系统，以便在调用该函数时系统按此进行对照检查。

主调函数通过参数将数据传递给被调函数，而被调函数则通过返回语句的返回值向主调函数传递数据。函数参数分为形参和实参两种，返回值是指函数被调用后返回给主调函数的值。

函数调用是使程序转而去执行其他函数体。函数允许相互调用，也允许嵌套调用，以及递归调用。

C++语言中的变量按作用域可分为局部变量和全局变量。而函数根据能否被外部源文件调用，可以分为内部函数和外部函数。

源程序往往由多个函数组成，但必须有且仅有一个 main 函数。程序从 main 函数开始执行，在执行过程中可以调用其他函数，最后回到 main 函数结束运行。

思考题

1. 简述什么是函数，如何进行函数定义，函数声明与定义的不同之处。
2. 简述什么是函数的返回类型，通过什么方式传递返回值。
3. 形式参数与实际参数分别是什么，阐明二者的区别和联系。
4. 简述如何调用一个已经定义的函数，参数是如何传递的。
5. 局部变量和全局变量有什么区别和联系？
6. 比较值调用和引用调用的相同点与不同点。
7. 函数原型中的参数名与函数定义中的参数名以及函数调用中的参数名是否必须一致。
8. 简述什么是嵌套调用和递归调用。

练习题

1. 计算函数 $F(x,y,z)=(x+z)/(y-z)+(y+2\times z)/(x-2\times z)$ 的值，要求将 F 函数实现为子函数，由主函数向子函数传参调用。
2. 要求用户输入一个年月日，判断用户输入的日期是否合法。定义子函数实现合法性验证。
3. 银行要受理客户的贷款请求，设计程序要求根据客户的贷款额度、期限和利率，计算并输出客户最终需要的还款总额。
4. 我国的个人所得税按照累进税计算，要求用户输入其收入，程序输出其应缴纳的个人所得税。
5. 编写函数计算三角形的面积，公式如下。

$$s=\frac{1}{2}(a+b+c)$$

$$\text{area}=\sqrt{s(s-a)(s-b)(s-c)}$$

三角形的三边 a、b、c 的值由主函数输入。

6. 编写函数：使用递归法求 $f(x)=\sum_{i=0}^{n} i^2$，n 的值由主函数输入。

7. 编写函数：统计字符串中字母的个数、数字的个数、空格的个数、其他字符的个数。主函数中从键盘输入一行字符，并输出统计结果。

8. 所谓“回文”是一种特殊的数或文字短语。它无论顺着读还是倒着读，结果都一样。例如，以下几个 6 位整数都是回文数：222222，522225，352235，662266。以下 5 位整数都是回文数：12321、77777、89998 和 44744。编写函数，输入一个 6 位或 7 位整数，判断它是否是回文数。如果是回文数，输出“Yes”；如果不是回文数，输出“No”。整数输入和输出是否为回文数，在主函数中完成。

例如：

请输入一个 6 位或 7 位整数：349943

```
Yes
```

请输入一个 6 位或 7 位整数：3499432

```
No
```

9. 在主函数输入二维平面上的 3 个点(x1,y1)，(x2,y2)和(x3,y3)，编写函数判断是否存在以这 3 个点为顶点的三角形(假定所有坐标值均为整数，且各坐标值的绝对值小于 215)，将判断结果返回到主函数中输出。