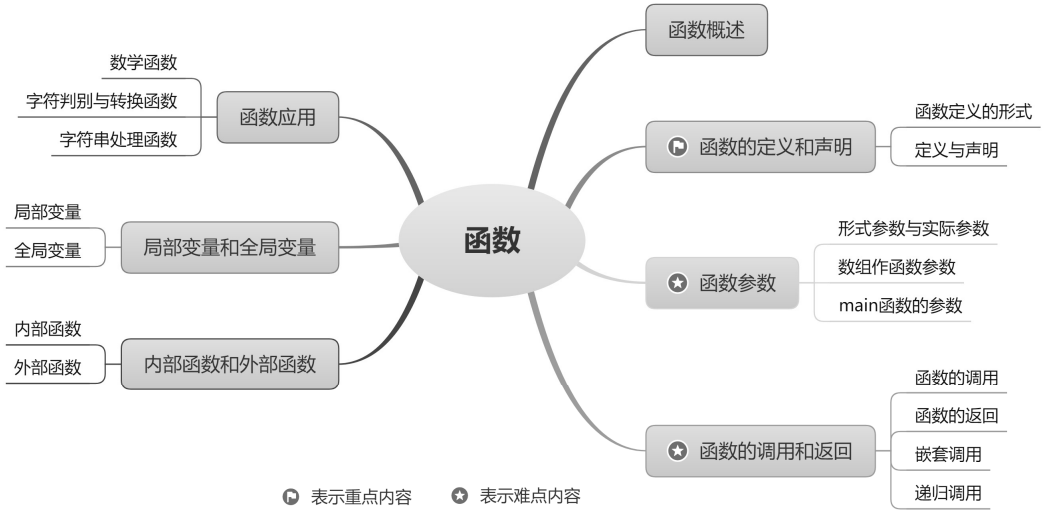
第 9 章

函数

大型程序一般会被分为若干个程序模块，每个模块实现一个特定功能。C 语言中，由函数实现子程序，由子程序实现模块功能。

本章致力于使读者了解函数的概念，掌握函数的定义及调用方式；了解内部函数和外部函数的作用范围，能区分局部变量和全局变量的不同；最后能将函数应用于程序中，将程序分成不同的功能模块。

本章的知识架构及重难点如下：



9.1 函 数 概 述

构成 C 程序的基本单元是函数，函数中包含着程序的可执行代码。

每个 C 程序的入口和出口都位于 main 函数中，但并不需要把所有指令都放在 main 函数中。一般的做法是将程序划分成若干个模块，每个模块完成一部分功能，不同的程序模块可以由不同的人来完成，从而提高软件开发的效率。

这就好比是盖楼房，一栋摩天大楼是不可能靠一个人完成的，而要靠多部门、多工种之间协力完成。通常是有一个总工程师，在他的指挥下，有部门运输建筑材料，有部门建造楼房主体，还有部门粉刷内外墙涂料。编写程序的道理与盖楼是一样的，主函数就像总工程师一样，控制着整体程序的推进和执行，其中定义的其他函数就好比参与盖楼的多个部门或工种，他们要通过某种调度（函数调用）才能完成特定的功能。

主函数可以调用其他函数，其他函数间也可以相互调用。函数可以有参数和返回值，通过它们实现数据间的传递。在主函数中调用其他函数，这些函数执行完毕之后会返回 main 函数中。通常把这些被调用的函数称为下层函数。函数调用发生时，立即执行被调用的函数，而调用者则进入等待的状态，直到被调用函数执行完毕。

【例 9.1】编写 3 个函数：做饭，钓鱼，写诗（**实例位置：资源包\TM\sl\09\01**）

在本实例中，定义 3 个函数来完成做饭、钓鱼、写诗等特定的功能，然后在主函数中调用它们。为了简化函数的功能，这里只让其输出一条提示信息。读者可通过本实例对函数有一个直观的认识。

```
#include<stdio.h>                    /*包含头文件*/
void Cook();                         /*声明 Cook 函数*/
void Fish();                         /*声明 Fish 函数*/
void Poem();                         /*声明 Poem 函数*/
int main()                           /*主函数 main*/
{
    Cook();                          /*调用 Cook 函数*/
    Fish();                          /*调用 Fish 函数*/
    Poem();                          /*调用 Poem 函数*/
    return 0;                        /*程序结束*/
}
void Cook()                          /*自定义 Cook 函数*/
{
    printf("会做饭\n");
}

void Fish()                          /*自定义 Fish 函数*/
{
    printf("会钓鱼\n");
}

void Poem()                          /*自定义 Poem 函数*/
{
    printf("会写诗\n");
}
```

在分析本实例之前，我们先来了解一下什么是 C 程序源文件、库函数和用户自定义函数。。

- ☑ 源文件：由一个或者多个函数组成。C 语言以源程序为单位进行编译，而不是以函数为单位进行编译。
- ☑ 库函数：由 C 语言系统提供，用户无须定义，调用前也不必做类型说明，但需要在程序开始部分包含有该函数原型的头文件。例如，要使用能在控制台显示信息的 printf 函数，需在程序开始时包含 stdio.h 头文件；要使用字符串操作函数 strlen、strcmp 等时，需在程序开始时包含 string.h 头文件。
- ☑ 用户自定义函数：用户编写的用来实现特定功能的函数。例如，Cook、Fish 和 Poem 函数都是自定义函数。

在本例程序中，首先包含了 stdio.h 头文件，然后声明了 3 个自定义函数，最后在主函数 main 中调用了这 3 个函数。在主函数 main 外，可以看到这 3 个函数的定义。

运行程序，显示效果如图 9.1 所示。

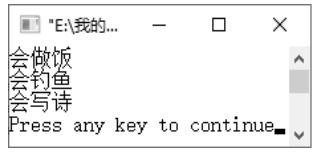


图 9.1　在主函数中调用其他函数

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 1：盖楼房　编写 3 个函数，分别实现搬运建筑材料、建造楼房主体、粉刷内外墙功能，并在主函数中调用这 3 个函数。输出结果如下：

```
执行搬运功能
执行建造功能
执行粉刷功能
```

训练 2：写情书　定义一个函数，内容为一封情书，在主函数中调用该函数，将情书内容展示出来。运行结果如下：

```
人生最美好的是相遇
我一生最奢侈的事
就是途中与你相遇
然后相濡以沫，共闻花香，有生之年
只诉温暖不言殇，倾心相遇，安暖相陪
```

9.2　函数的定义和声明

C 语言的库函数可以直接调用，如 printf 输出函数。而自定义函数则必须由用户进行定义，确定其要实现的功能，这样才能被其他函数调用。

9.2.1　函数的定义

一个函数应包括函数头和函数体。定义一个函数的语法格式如下：

```
返回值类型  函数名(参数列表)
{
    函数体                                  /*函数实现特定功能的过程*/
}
```

首先来看一段代码，然后通过分解，了解函数的构成。

```
int AddTwoNumber(int iNum1,int iNum2)          /*函数头部分*/
{
    int result;                                /*定义整型变量*/
    result = iNum1+iNum2;                      /*进行加法操作*/
    return result;                             /*返回操作结果，结束*/
}
```

函数头

函数体

1．函数头

函数头是函数的入口，标志着一段函数代码的开始。函数头包括返回值类型、函数名和参数列表 3

个部分，如图 9.2 所示。

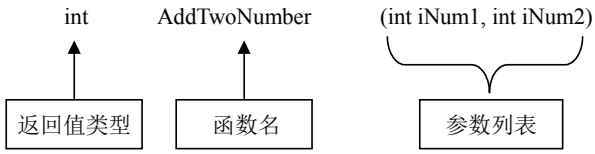


图 9.2　函数头组成

- ☑ 返回值类型：函数返回值的类型，必须是 C 语言中的某个数据类型。这里函数的返回值将是一个 int 型值。
- ☑ 函数名：函数的标识符，在一个 C 程序中应保持唯一。因为是标识符，所以函数名要遵守标识符命名规则。这里，函数名是 AddTwoNumber，可以推测出该函数的功能是两数相加求和。
- ☑ 参数列表：调用函数时，用于将主调函数中的实际参数复制到该列表对应的形式参数中。可以没有参数，也可以有多个参数。这里定义了两个 int 变量，表示要相加的两个数。

2．函数体

函数体位于函数头的下方位置，由一对大括号括起来，大括号决定了函数体的范围。函数要实现的特定功能，都是在函数体部分通过代码语句完成的，最后通过 return 语句返回实现的结果。

在上面的代码中，函数体内首先定义了一个 int 型变量，用来保存加法的计算结果，之后利用传递进来的参数进行加法操作，并将结果保存在 result 变量中，最后函数要将所得到的结果进行返回。通过这些语句的操作，实现了求解两数和的特定功能。

在定义函数时会出现以下几种特殊的情况。

- ☑ 无参函数：没有参数列表的函数。如例 9.1 中的 Cook、Fish、Poem 都是无参函数。
- ☑ 空函数：没有任何内容，也没有什么实际功能的函数。空函数的形式如下：

```
类型说明符 函数名()
{
}
```

实际开发中，有时某个函数还未编好，或者后续要拓展某个函数，这时就会先用一个空函数代替，先占个位置，待后续时机成熟再用编好的函数取代它。

注意，C 语言中，函数的定义是互相平行、独立的。也就是说，函数体内不能再包含其他函数的定义。例如，下面的代码是错误的：

```
int main()
{
    void Display()                                  /*错误！不能在函数体内定义另一个函数*/
    {
        printf("I want to show the Nesting function");
    }
    return 0;
}
```

这里，主函数 main 中定义了一个 Display 函数，目的是输出一句提示。由于 C 语言不允许进行嵌

套定义，因此编译时会出现如图 9.3 所示的错误提示。

```
error C2143: syntax error : missing ';' before '{'
```

图 9.3　错误提示

9.2.2　函数的声明

在程序中编写函数时，要先对函数进行声明，再对函数进行定义。函数定义是为了让编译器知道函数的功能，而函数声明是为了让编译器预先知道有这么一个函数，以及函数的名称、参数、返回值类型等信息。

函数声明的一般形式如下：

```
返回值类型 函数名(参数列表);
```

要注意的是，函数声明语句的最后要用分号“;”作为结尾。例如，声明一个函数的代码如下：

```
Int ShowNumber(int iNumber);
```

【例 9.2】交换两个数值（**实例位置：资源包\TM\sl\09\02**）

通过本实例了解函数声明与函数定义的位置，及其在程序中的作用。

```
#include<stdio.h>
void exchange(int a,int b);                          /*声明 exchange 函数*/
int main()
{
    int a=3,b=4;                                     /*定义整型变量*/
    printf("交换之前的值 a=%d,b=%d\n",a,b);          /*提示信息*/
    exchange(a,b);                                   /*调用 exchange 函数*/
    return 0;
}
void exchange(int a,int b)                           /*定义 exchange 函数，用于交换两个数*/
{
    int c;                                           /*交换数值*/
    c=a;
    a=b;
    b=c;
    printf("交换后的值 a=%d,b=%d\n",a,b);            /*输出交换之后的数据*/
}
```

（1）观察上面的程序，可以看到在 main 函数的开头先进行了 exchange 函数的声明，声明的作用是告知其函数将在后面进行定义。

（2）在 main 函数体中，首先定义两个整型变量 a、b，之后输出一条提示消息，然后调用 exchange 函数。

（3）在 main 函数的定义之后可以看到 exchange 函数的定义，功能是实现两个数的数值互换。

运行程序，结果如图 9.4 所示。

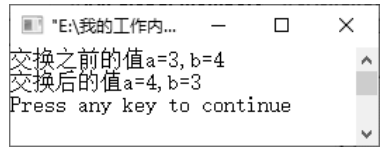


图 9.4　函数的定义与声明

注意

如果先定义函数，再调用函数，则不再需要进行函数声明，此时函数定义已包含了函数声明的作用。

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 3：输出谚语　编写一个程序，输出“最大的挑战和突破在于用人，而用人最大的突破是信任人”。运行效果如下：

```
最大的挑战和突破在于用人，而用人最大的突破是信任人
```

训练 4：打印新年菜单　春节是中国最重要的节日，家家户户都会张灯结彩，丰盛的年夜饭更是必不可少的。定义一个函数，打印 2021 年新年菜单，再在主函数中调用函数。运行结果如下：

```
-----新年菜单如下-------
 1.凉菜
 2.红烧鱼
 3.煮虾
 4.酱猪蹄
 5.红烧排骨
 6.孜然牛肉
 7.木须柿子
 8.水煮肉片
-----------------------
```

9.3　函 数 参 数

多数情况下，主调函数和被调函数之间存在着数据传递关系，这种数据传递是通过函数参数来实现的。函数参数的作用是传递数据给函数使用，函数利用接收到的数据进行具体的操作处理。

9.3.1　形式参数与实际参数

函数的参数分为两种：形式参数和实际参数。同为参数，读者要仔细体会其中的区别。

1．形式参数

声明和定义函数时，函数名后面括号中的参数称为形式参数。这些参数只是定义了类型，在实际参数传入前并没有实际意义，因此叫作形式参数，简称形参。

2．实际参数

调用函数时，函数名后面括号中的参数称为实际参数。调用函数的过程就是真正使用这个函数的过程，此时调用者会传递一些要实际参与运算的参数给被调用函数，这些实际参与运算的参数就是实际参数，简称实参。

下面来看一段代码，加深对形参和实参的理解。

```
int Minus(int iNumber1,int iNumber2)                /*定义 Minus 函数，有两个 int 型参数*/
{
    int iResult;                                    /*定义整型变量*/
    iResult=iNumber1-iNumber2;                      /*两个 int 型参数相减*/
    return iResult;                                 /*返回计算结果*/
}
int main()
{   …
    iResult=Minus(9,4);                             /*调用 Minus 函数，传入实参 9,4，函数执行 9-4 运算*/
    …
}
```

上述代码定义了一个 Minus 函数，功能是两数相减。这里，iNumber1 和 iNumber2 是形式参数，表示相减的两个整型数，没有具体值（等待实参传入）。主函数中，通过“iResult=Minus(9,4);”语句调用 Minus 函数，这里 9 和 4 为实际参数，调用函数后，将用 4 代替 iNumber1，用 9 代替 iNumber2，因此 Minus 函数实际执行的运算的是 9−4。

注意，函数参数可以是常量、变量、数组、指针等，也可以是表达式。

9.3.2　数组作函数参数

1．数组元素作为函数参数

数组元素作为函数实参，与普通变量作为函数实参一样，是单向的值传递。

【例 9.3】输出数组元素（**实例位置：资源包\TM\sl\09\03**）

定义一个数组并为其赋值，然后将数组元素作为函数实参进行传递。自定义函数体中，形参得到实参传递的数值后，将其显示输出。

```
#include<stdio.h>
void ShowMember(int iMember);                       /*声明 ShowMember 函数，形参为一个整型数*/

int main()
{
    int iCount[10];                                 /*定义一个整型数组*/
    int i;                                          /*定义整型变量*/
    for(i=0;i<10;i++)                               /*for 循环，对数组元素依次赋值*/
    {
        iCount[i]=i;
    }
    for(i=0;i<10;i++)                               /*循环调用 ShowMember 函数*/
    {
        ShowMember(iCount[i]);                      /*函数实参为数组元素*/
    }
    return 0;
}

void ShowMember(int iMember)                        /*自定义 ShowMember 函数，形参为一个整型数*/
```

```
{
    printf("Show the member is%d\n",iMember);    /*输出数据*/
}
```

（1）首先进行函数声明，在主函数 main 中定义一个整型数组和一个整型变量 i。

（2）使用 for 循环语句对数组中的元素依次赋值，在这里，变量 i 既是循环条件，也是引用数组元素的下标。

（3）通过 for 循环语句调用 ShowMember 函数，显示数据。

运行程序，显示效果如图 9.5 所示。

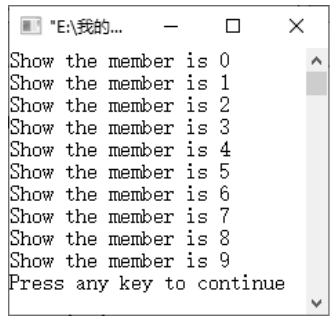


图 9.5　数组元素作为函数参数

2．数组名作为函数参数

C 语言中，数组名表示的是数组中第一个元素的地址。因此，当数组名作为函数实参时，传递的是数组的地址。这点和数组元素做实参时是不一样的，注意体会其不同。

【例 9.4】数组名作为函数参数（**实例位置：资源包\TM\sl\09\04**）

在本实例中，使用数组名作为函数的实参和形参，实现数组的赋值和输出。

```
#include<stdio.h>
void Evaluate(int iArrayName[10]);                  /*声明赋值函数，形参为一个数组名*/
void Display(int iArrayName[10]);                   /*声明输出函数，形参为一个数组名*/
int main()
{
    int iArray[10];                                 /*定义一个具有 10 个元素的整型数组*/
    Evaluate(iArray);                               /*调用函数，实现数组元素赋值，数组名为实参*/
    Display(iArray);                                /*调用函数，实现数组元素输出，数组名为实参*/
    return 0;
}
/*////////////////////////////////////////////////////////////////////////////////////////////////////*/
/*                                  进行数组元素的输出                                                */
/*////////////////////////////////////////////////////////////////////////////////////////////////////*/
void Display(int iArrayName[10])                    /*定义输出函数，形参为一个数组*/
{
    int i;                                          /*定义整型变量*/
    for(i=0;i<10;i++)                               /*在循环语句中执行数组输出*/
    {
        printf("the member number is %d\n",iArrayName[i]);
    }
}
/*////////////////////////////////////////////////////////////////////////////////////////////////////*/
/*                                  进行数组元素的赋值                                                */
/*////////////////////////////////////////////////////////////////////////////////////////////////////*/
void Evaluate(int iArrayName[10])                   /*定义赋值函数，形参为一个数组*/
{
    int i;                                          /*定义整型变量*/
    for(i=0;i<10;i++)                               /*在循环语句中执行数组赋值*/
    {
```

```
            iArrayName[i]=i;
        }
}
```

（1）首先对 Evaluate 函数和 Display 函数进行声明，在声明语句中可以看到数组名作为形参。

（2）在主函数 main 中定义一个具有 10 个元素的整型数组 iArray。

（3）调用 Evaluate 函数，数组名 iArray 作为函数实参，传递的是数组的地址。在 Evaluate 函数中，使用数组 iArrayName 作为形参，接受对应的地址空间，并对数组进行赋值操作。

（4）调用 Display 函数，将数组输出，可以看到在函数参数中使用的也是数组名称。

运行程序，显示效果如图 9.6 所示。

3．可变长度数组作为函数参数

数组作为函数参数时，如果未指明数组大小，就属于长度可变的数组作为函数参数。

【例 9.5】不使用库函数，实现字符串连接功能（**实例位置：资源包\TM\sl\09\05**）

在本实例中，不使用编译器提供的库函数，将函数的参数声明为长度可变数组。具体代码如下：

```
#include<stdio.h>                                   /*包含头文件*/
void _strcat(char str1[],char str2[])               /*自定义 strcat 函数，形参为两个长度可变的数组*/
{
      int i,j;                                      /*定义控制变量*/
      for(i=0;str1[i]!='\0';i++);                   /*字符数组 1 中循环*/
            for(j=0;str2[j]!='\0';j++)              /*字符数组 2 中循环*/
                  str1[i+j]=str2[j];                /*字符串连接*/
       str1[i+j]='\0';                              /*结束*/
 }
int main()                                          /*主函数 main*/
{
      char str1[100],str2[100];                     /*定义字符数组*/
      printf("请输入字符串 1:\n");                  /*提示信息*/
      gets(str1);                                   /*输入字符串 1*/
      printf("请输入字符串 2:\n");                  /*提示信息*/
      gets(str2);                                   /*输入字符串 2*/
      _strcat(str1,str2);                           /*调用函数连接 2 个字符串*/
      printf("连接之后的字符串是：%s\n",str1);      /*显示连接后的字符串*/
     return 0;                                      /*程序结束*/
}
```

运行程序，显示效果如图 9.7 所示。

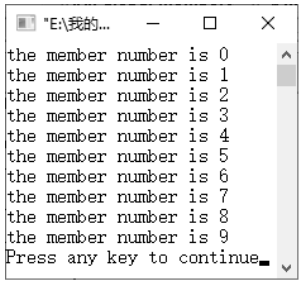


图 9.6　数组名作为函数参数

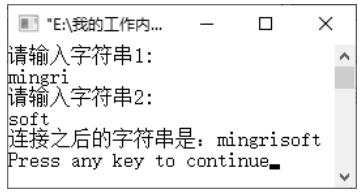


图 9.7　可变长度数组作为函数参数

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 5：用户登录验证　编写登录函数，函数有两个形参：账号名和密码。如果账号名为“张三”，密码为“123”，则登录成功，否则登录失败。运行结果如下：

```
登录成功
```

训练 6：自定义运算函数　编写函数 fun(int a[][N], int n)，使数组左下半边角的元素值均乘以 n。例如，n = 3, a = {{1,9,7},{2,3,8},{4,5,6}}，运行程序后得到{{3,9,7},{6,9,8},{12,15,18}}。运行结果如下：

```
a[0][0]=1
a[0][1]=9
a[0][2]=7
a[1][0]=2
a[1][1]=3
a[1][2]=8
a[2][0]=4
a[2][1]=5
a[2][2]=6
左下半边角元素中的值乘以 n 的结果是：
3          9          7
6          9          8
12         15         18
```

9.4　函数的调用和返回

通过前面的学习，我们已经知道，使用函数的过程就是调用函数的过程。

9.4.1　函数的调用

函数的调用方式有 3 种，包括语句调用、表达式调用和函数参数调用。

1．语句调用

函数调用作为一个独立语句出现，就称为语句调用。这是最常用的函数调用方式，可以有返回值，也可以没有返回值。例如：

```
Display();                                    /*通过语句调用 Display 函数*/
```

2．表达式调用

当函数调用出现在一个表达式中时，函数必须返回一个确定的值，作为表达式运算的一部分。例如：

```
iResult=iNum3*AddTwoNum(3,5);                 /*函数调用出现在表达式中，返回值参与乘法运算*/
```

这条语句中，函数 AddTwoNum 的返回值将与 iNum3 执行乘法，得到的结果赋值给 iResult 变量。

【例 9.6】实现欧姆定律功能（**实例位置：资源包\TM\sl\09\06**）

在本实例中，定义一个函数，其功能是利用欧姆定律（R=U/I）计算电阻值，并在表达式中调用该函数，使其返回值参与运算。

```
#include<stdio.h>                          /*包含头文件*/
void TwoNum(float iNum1, float iNum2);     /*声明函数，函数进行计算*/
int main()
{
    TwoNum(5,10);                          /*调用函数*/
    return 0;                              /*程序结束*/
}
void TwoNum(float iNum1, float iNum2)      /*定义函数*/
{
    float iTempResult;                     /*定义整型变量*/
    iTempResult=iNum1/iNum2;               /*进行计算，并将结果赋值给 iTempResult*/
    printf("电阻值是%f\n",iTempResult);    /*显示输出电阻值*/
}
```

（1）先对要使用的函数进行声明操作。

（2）在主函数 main 中，调用 TwoNum 函数来计算数值 5 和 10 的相除。

（3）定义函数 TwoNum，定义变量用来存储计算的结果，利用表达式计算欧姆值，并且将运算结果赋值给变量 iTempResult。使用 printf 函数对所得到的结果进行输出显示。运行程序，显示效果如图 9.8 所示。

3．函数参数调用

函数调用还可以出现在函数参数中。此时，函数的返回值将作为参数使用。例如：

```
iResult=AddTwoNum(10,AddTwoNum(3,5));      /*函数调用出现在函数参数中*/
```

在这条语句中，AddTwoNum 函数的功能还是进行两个数相加，然后将相加的结果作为函数的一个参数，继续进行计算。

【例 9.7】判断体温是否正常（**实例位置：资源包\TM\sl\09\07**）

本实例编写 getTemperature 函数得到体温值，将其返回的体温数据交给 judgeTemperature 函数，作为参数使用。具体代码如下：

```
#include<stdio.h>                          /*包含头文件*/
void judgeTemperature(float temperature);  /*声明函数*/
float getTemperature();                    /*声明函数*/

int main()                                 /*主函数 main*/
{
    judgeTemperature(getTemperature());    /*调用 judgeTemperature 函数，实参为 getTemperature 函
数的返回值*/
    return 0;                              /*程序结束*/
}
float getTemperature()                     /*定义 getTemperature 函数*/
{
```

```
    float temperature;                              /*定义变量*/
    printf("please input a temperature:\n");        /*输出提示信息*/
    scanf("%f",&temperature);                       /*输入体温*/
    printf("当前体温是：%.1f\n",temperature);        /*输出当前体温值*/
    return temperature;                             /*返回体温值*/
}

void judgeTemperature(float temperature)            /*定义 judgeTemperature 函数，形参为一个温度值*/
{
    if(temperature<=37.3f&& temperature>=36)        /*判断体温值是否正常*/
        printf("体温正常\n");
    else
        printf("体温不正常\n");
}
```

运行程序，显示效果如图 9.9 所示。

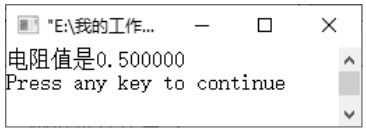


图 9.8　表达式调用

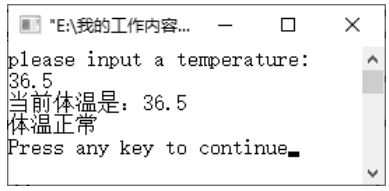


图 9.9　函数参数调用

9.4.2　函数的返回

在函数体中常会看到这样一条 return 语句。

```
return 0;
```

这就是返回语句。return 语句的作用有两个，下面进行详细介绍。

1．退出函数，返回主调程序中

C 程序中，有两种方法可以终止函数执行，返回到调用函数位置。

☑　函数体中的所有语句都已执行完毕，遇到结束符号“}”后自动返回。

☑　遇到 return 语句，随即返回。

2．返回一个值，供主调函数使用

用户调用函数时，通常是希望得到一个确定的返回值。该返回值是通过 return 语句实现的。返回值也需要约定类型，因此定义函数时，需要明确指定函数返回值的类型。例如：

```
int Max(int iNum1,int iNum2);                   /*函数返回值类型为 int*/
double Min(double dNum1,double dNum2);          /*函数返回值类型为 double*/
char Show(char cChar);                          /*函数返回值类型为 char*/
```

如果函数返回值的类型和 return 语句中表达式的值不一致，则以函数返回值的类型为准。数值型数据可以自动进行类型转换，即函数定义的返回值类型决定最终返回值的类型。

注意，return 语句后面的括号可以省略，即 return 0 和 return(0)是相同的。另外，函数也可以没有返回值。例如，返回值类型为 void 的函数就没有返回值。

【例 9.8】返回体温值（实例位置：资源包\TM\sl\09\08）

在本实例中，编写函数 getTemperature，返回一个体温值。具体代码如下：

```
#include<stdio.h>                                   /*包含头文件*/
float getTemperature();                             /*声明函数*/
int main()                                          /*主函数 main*/
{
    getTemperature();                               /*调用函数*/
    return 0;                                       /*程序结束*/
}

float getTemperature()                              /*自定义温度函数*/
{
    float temperature;                              /*定义浮点型变量*/
    printf("please input a temperature:\n");        /*输出提示信息*/
    scanf("%f",&temperature);                       /*输入一个浮点型变量*/
    printf("当前体温是：%.1f\n",temperature);       /*输出当前温度*/
    return temperature;                             /*返回温度值*/
}
```

（1）首先为程序声明一个 getTemperature 函数，在主函数调用自定义 getTemperature 函数。

（2）getTemperature 函数的定义中，定义了一个浮点型变量，用户通过提示信息输入一个温度值，最后将温度值返回。在这里可以看到，getTemperature 函数中定义的温度值是 float 型，所以返回的也是 float 型变量。运行程序，显示效果如图 9.10 所示。

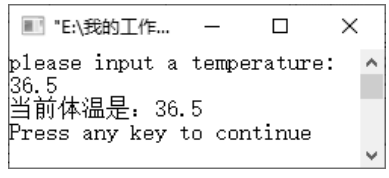


图 9.10　返回值类型与 return 值类型

9.4.3　函数的嵌套调用

C 语言虽然不允许进行函数嵌套定义，但却可以函数嵌套调用，即可在一个函数体内可以调用另外一个函数。例如，在 main 函数中调用 Display 函数，在 Display 函数中再调用 ShowMessage 函数，这种就叫作嵌套调用。调用时，一般从最外层 main 函数开始，一层层的调用，直到深入最内层的函数。嵌套调用返回时，恰好相反，从最内层的函数开始，一层层返回，直到返回最外层的主调函数中。

这就好比某公司要完成业绩，CEO 需要将运营指标交代给总经理，总经理将工作分拆给各部门经理（副经理），部门经理们再将工作分拆给下属职员，职员按照上级指示完成具体工作。之后，职员将完成结果汇报给部门经理，部门经理汇总后将结果汇报给总经理，总经理再汇总后将结果汇报给 CEO。

【例 9.9】CEO 下达工作内容（实例位置：资源包\TM\sl\09\09）

在本实例中，利用嵌套函数模拟 CEO 分派工作的过程。这里简化任务的完成过程，只输出一条语句表示逐级分派任务（即调用函数）的过程。

```
#include<stdio.h>
```

```
void CEO();                                    /*声明 4 个函数*/
void Manager();
void AssistantManager();
void Clerk();

int main()
{
    CEO();                                     /*调用 CEO 函数*/
    return 0;
}
void CEO()                                     /*定义 CEO 函数*/
{
    printf("CEO 交代给总经理\n");
    Manager();                                 /*调用 Manager 函数*/
}
void Manager()                                 /*定义 Manager 函数*/
{
    printf("总经理交代给部门经理\n");
    AssistantManager();                        /*调用 AssistantManager 函数*/
}
void AssistantManager()                        /*定义 AssistantManager 函数*/
{
    printf("部门经理交代给各个职员\n");
    Clerk();                                   /*调用 Clerk 函数*/
}
void Clerk()                                   /*定义 Clerk 函数*/
{
    printf("职员开始执行\n");
}
```

（1）首先在程序中声明将要使用的 4 个函数，其中的 CEO 代表公司总裁，Manager 代表总经理，AssistantManager 代表副经理，Clerk 代表职员。

（2）main 函数下是 4 个函数的定义。先来看一下 CEO 函数，通过输出一条信息来表示这个函数的功能和作用，然后嵌套调用 Manager 函数。Manager 和 CEO 函数运行的步骤是相似的，最后在其函数体内调用 AssistantManager 函数。在 AssistantManager 函数中调用 Clerk 函数。

（3）在主函数 main 中，调用了 CEO 函数，于是程序的整个流程按照步骤（2）进行，直到“return 0”语句返回，程序结束。

运行程序，显示效果如图 9.11 所示。

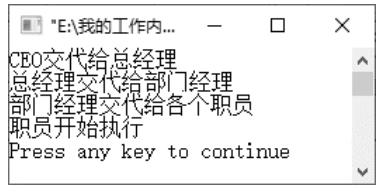


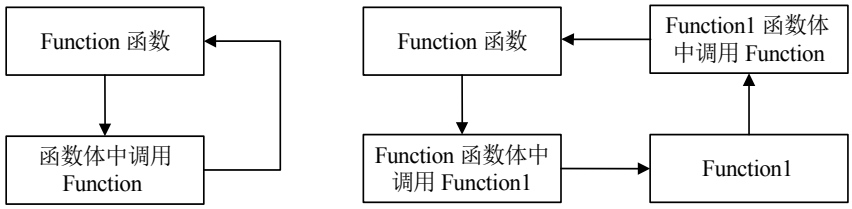
图 9.11　函数的嵌套调用

9.4.4　函数的递归调用

所谓递归调用，就是函数自己调用自己。从定义中可以看出，函数递归调用是函数嵌套调用的一种特殊形式。

函数可以直接调用自己，也可以间接调用自己，如图 9.12 所示。所谓间接调用，就是在递归函

数的下层函数中调用自己。



（a）直接递归　　（b）间接递归

图 9.12　递归调用过程

递归之所以能实现，是因为函数的每个执行过程在栈中都有自己的形参和局部变量副本，这些副本和该函数的其他执行过程不发生关系。这种机制是当代大多数程序设计语言实现子程序结构的基础。

【例 9.10】倒序输出名单（**实例位置：资源包\TM\sl\09\10**）

在本实例中，定义一个字符串数组，为其赋值一系列名称，通过递归函数调用，逆序显示名单。注意，本例之所以能逆序输出名单，是因为嵌套函数返回时是从最内层到最外层逐层返回。

```
#include<stdio.h>
void DisplayNames(char** cNameArray);              /*声明递归函数 DisplayNames*/

char* cNames[]=                                    /*定义字符串数组*/
{
    "Aaron",
    "Jim",
    "Charles",
    "Sam",
    "Ken",
    "end"                                          /*设定一个结束标志*/
};

int main()
{
    DisplayNames(cNames);                          /*调用 DisplayNames 函数*/
    return 0;
}

void DisplayNames(char** cNameArray)               /*定义 DisplayNames 函数*/
{
    if(*cNameArray=="end")                         /*判断是否是结束标志*/
    {
        return ;                                   /*如果是，函数结束返回*/
    }
    else
    {
        DisplayNames(cNameArray+1);                /*如果不是，递归调用自身*/
        printf("%s\n",*cNameArray);                /*输出字符串*/
    }
}
```

如图 9.13 所示为程序的调用流程，通过此图读者可对递归调用过程有一个更直观的认识。

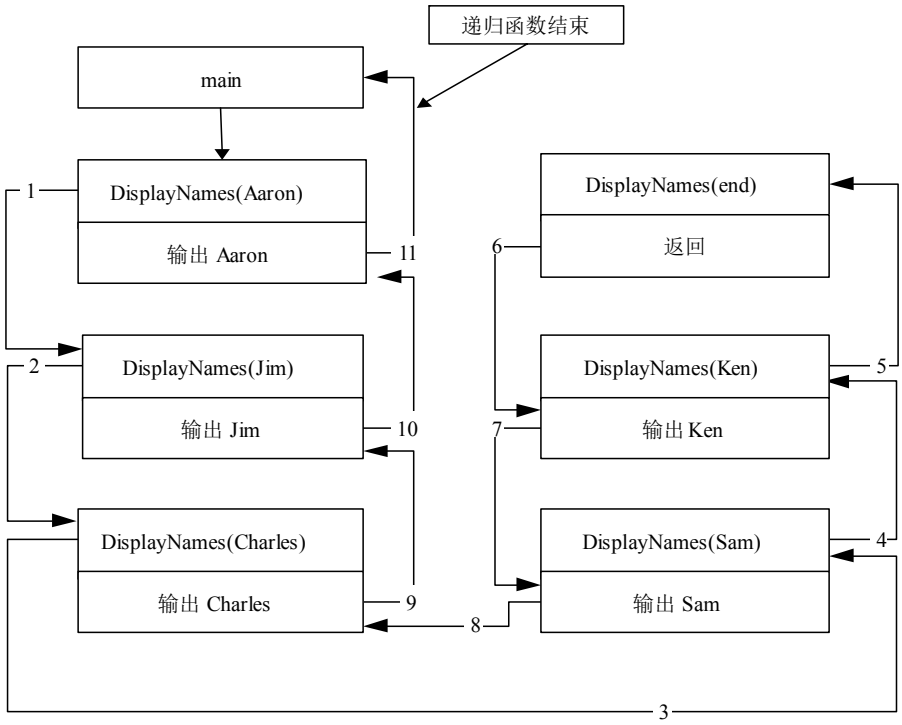


图 9.13　程序调用流程图

（1）首先声明递归函数 DisplayNames，其参数声明为指针的指针（第 10 章中将介绍，这里不用深究）。

（2）定义一个全局字符串数组，并且为其进行赋值。其中，字符串“end”作为数组的结尾标志。

（3）在主函数 main 中调用递归函数 DisplayNames。

（4）在 DisplayNames 的函数体中，通过 if 语句判断当前要输出的字符串是否是结束字符“end”，如果是，则使用 return 语句返回。否则，执行下面的 else 语句，调用递归函数，在函数参数处可以看到传递的字符串数组元素发生改变，传递下一个数组元素。调用递归函数后，仍然要先判断传递进来的字符串是否是数组的结束标志。最后输出字符串数组的元素。

运行程序，显示效果如图 9.14 所示。

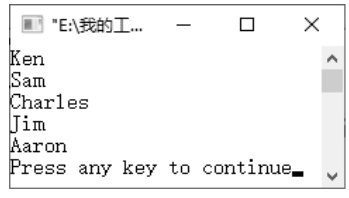


图 9.14　函数的递归调用

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 7：判断某个数是否为回文数　编写函数，判断字符串是否为回文数（即顺读和倒读是一样的）。若是，函数返回 1，否则返回 0。运行结果如下：

```
请输入一个数：
12321
是回文数
```

训练 8：某宝集福活动　每年春节，支付宝都会推出集福活动。自定义一个函数，将 2020 年支付宝推出的福打印出来，然后在主函数中调用编写的函数，运行结果如下：

```
五福为：
爱国福
富强福
和谐福
友善福
敬业福
附加福为：
全家福
沾气福
```

训练 9：魔术师及代表作　自定义一个包含魔术师及其代表作的函数，在主函数中调用这个函数，输出每个魔术师的名字及作品，运行结果如下：

```
1、Jason Latimer：“划时代的近景魔术师”，代表魔术《杯与球》《激光魔术》
2、大卫·科波菲尔：代表作《人体切割》《消失的自由女神》。
3、大卫·布莱恩：擅长跳脱术，曾在一座桥下不吃不喝坚持了 44 天 。
4、Daryl：近景魔术大师。出过很多教学光碟 。
5、Michael.Ammar：也是近景魔术大师，出过很多教学 。
```

训练 10：判断血压是否正常　正常的血压收缩压是 90～140mmHg，舒张压是 60～90mmHg。如果收缩压低于 90mmHg 或舒张压低于 60mmHg，属于低血压。如果收缩压高于 140mmHg 或舒张压高于 90mmHg，属于高血压。编写程序，判断舒张压是否正常，运行结果如下：

```
请输入舒张压数值:
88
当前舒张压是：88
血压正常
```

9.5　内部函数和外部函数

函数是 C 程序中的最小单位，可以把一个或多个函数保存为一个文件，这个文件就称为源文件。当一个源程序由多个源文件组成时，多个文件之间的函数可以相互调用。当然我们也可以通过将其设置为内部函数，禁止其被其他文件调用。因此，根据能不能被其他文件调用，C 语言又把函数分为两类：一类是内部函数，另一类是外部函数。

9.5.1　内部函数

定义一个函数，如果该函数只能被所在的源文件使用，那么就称这样的函数为内部函数。内部函数又称为静态函数。使用内部函数，可以使函数只局限在函数所在的源文件中，如果在不同的源文件中有同名的内部函数，则这些同名函数间是互不干扰的。

定义内部函数时，要在函数返回值和函数名前面加上关键字 static 进行修饰。一般形式如下：

```
static 返回值类型 函数名(参数列表)
```

例如，定义一个功能为加法运算且返回值是 int 型的内部函数，代码如下：

```
static int Add(int iNum1,int iNum2)
```

在函数的返回值类型 int 前加上关键字 static，可将原来的函数指定成内部函数。

技巧

使用内部函数的好处是，不同开发者编写函数时，不必再担心函数是否会与其他源文件中的函数同名。因为内部函数只在所在源文件中有效，不同源文件中即使有相同的函数名，也没有关系。

【例 9.11】显示“Hello MingRi!”（**实例位置：资源包\TM\sl\09\11**）

在本实例中使用内部函数，通过一个函数对字符串进行赋值，再通过一个函数对字符串进行输出显示。

```
#include<stdio.h>

static char* GetString(char* pString)                    /*定义字符串赋值函数*/
{
    return pString;                                      /*返回字符*/
}
static void ShowString(char* pString)                    /*定义字符串输出函数*/
{
    printf("%s\n",pString);                              /*显示字符串*/
}

int main()
{
    char* pMyString;                                     /*定义字符串变量*/
    pMyString=GetString("Hello    MingRi!");             /*调用函数，为字符串赋值*/
    ShowString(pMyString);                               /*调用函数，输出显示字符串*/
    return 0;
}
```

在程序中，使用 static 关键字对函数进行修饰，使其只能在其源文件中进行调用。

运行程序，显示效果如图 9.15 所示。

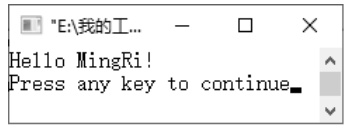


图 9.15　内部函数的使用

9.5.2　外部函数

外部函数是可以被其他源文件调用的函数。定义外部函数时，使用关键字 extern 进行修饰。同样，使用外部函数时，要先用 extern 声明所用的函数是外部函数。

例如，外部函数的函数头可以写成下面的形式：

```
extern int Add(int iNum1,int iNum2);
```

这样，Add 函数就可以被其他源文件调用，进行加法运算。

注意

C 语言中，定义函数时如果未指明是内部函数还是外部函数，系统将默认此函数为外部函数。也就是说，定义外部函数时可以省略关键字 extern。本书中多数实例使用的函数都为外部函数。

【例 9.12】求三角形的内角度数（**实例位置：资源包\TM\sl\09\12**）

一个三角形的 3 个内角度数之比为 2∶3∶3，求各内角的度数。使用外部函数编写程序求解。

```
/*////////////////////////////////////////////////////////////////////////////////////////////////*/
/*                                cal1.c 文件                                                     */
/*////////////////////////////////////////////////////////////////////////////////////////////////*/
#include<stdio.h>
extern void GetAngle(int a,int b,int c)                          /*定义 GetAngle 函数*/
{
    float a1=0,b1=0,c1=0;                                        /*定义 3 个变量，用来表示三角形的 3 个内角*/
    a1=(float)180*a/(a+b+c);                                     /*求每个内角的度数*/
    b1=(float)180*b/(a+b+c);
    c1=(float)180*c/(a+b+c);
    printf("内角的度数分别是：%.1f,%.1f,%.1f\n",a1,b1,c1);    /*输出内角度数*/
}
```

```
/*////////////////////////////////////////////////////////////////////////////////////////////////*/
/*                                Cal.c                                                           */
/*////////////////////////////////////////////////////////////////////////////////////////////////*/
#include<stdio.h>
extern void GetAngle(int a,int b,int c);                         /*声明外部函数*/
int main()
{
    GetString(2,3,3);                                            /*调用 cal1 文件中的 GetString 函数*/
    return 0;
}
```

主函数 main 在源文件 Cal.c 中。首先声明一个函数，其中使用 extern 关键字说明函数为外部函数。然后在 main 函数体中调用这个函数。在 cal1.c 源文件中对 GetString 函数进行定义，通过对传入的参数执行返回操作，完成对变量的赋值功能。

运行程序，显示效果如图 9.16 所示。

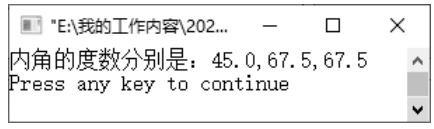


图 9.16　外部函数的使用

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 11：嵌套计算　利用嵌套函数计算 s=12!+ 22!+ 32!+ 42!+ 52!。运行结果如下：

```
计算的结果是：362905
```

训练 12：输出名言　编写外部函数并调用，输出名言“我们必须在别人改变之前改变自己”。

9.6 局部变量和全局变量

在讲解局部变量和全局变量的知识之前，先来了解一下作用域。作用域决定了程序中变量的作用范围：局部变量的作用域是局部的，如函数体内；全局变量的作用域是整个程序。下面具体看一下有关局部变量和全局变量的内容。

9.6.1 局部变量

在函数内部定义的变量是局部变量。我们前面学习的实例中绝大多数变量都只是局部变量，这些变量声明在函数内部，无法被其他函数所使用。函数的形式参数也属于局部变量，作用范围仅限于函数内部的所有语句块。

图 9.17 表示的是不同情况下局部变量的作用域范围。

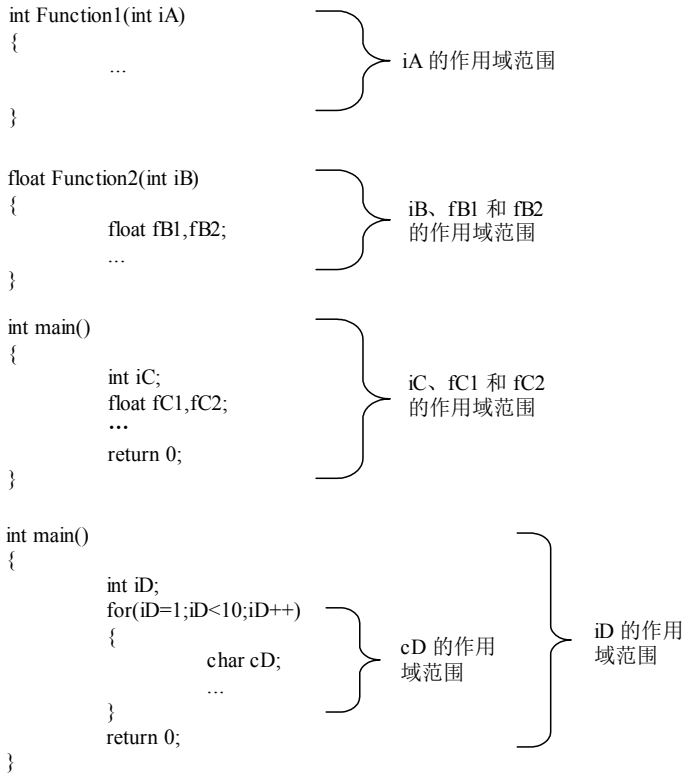


图 9.17 局部变量的作用域范围

说明

在语句块内声明的变量仅在该语句块内部起作用，当然也包括嵌套在其中的子语句块。

【例 9.13】模拟美团外卖商家的套餐（**实例位置：资源包\TM\sl\09\13**）

某米线店推出如下套餐：考神套餐 13 元，单身套餐 9.9 元，情侣套餐 20 元。

本实例在不同的位置定义一些变量，并为其赋值来表示变量的所在位置，最后输出显示其变量值，通过输出的信息来观察局部变量的作用范围。

```
#include<stdio.h>                                          /*包含头文件*/

int main()                                                 /*主函数 main*/
{
    int iNumber1;                                          /*定义变量*/
    printf("米线店套餐如下：1：考神套餐 2：单身套餐 3：情侣套餐\n");
    if(merchant1=1)
    {
        int iNumber2;                                      /*iNumber2 的作用域在 if 语句块中*/
        printf("考神套餐 13 元\n");
        if(iNumber2=2)
        {
            int iNumber3;                                  /*iNumber3 的作用域在 if 语句块中*/
            printf("单身套餐 9.9 元\n");
            if(iNumber3=3)
            {
                printf("情侣套餐 20 元\n");
            }
        }
    }
    return 0;
}
```

（1）程序中有 3 个作用域范围，主函数 main 是其中最大的作用域范围。因为定义变量 iNumber1 在 main 函数中，所以 iNumber1 的作用范围是整个 main 函数体。

（2）iNumber2 定义在第一个 if 语句块中，因此它的使用范围就是在第一个 if 语句块内。变量 iNumber3 在最内部的嵌套层，因此使用范围只在最里面的 if 语句块中。

局部变量的作用范围由包含该变量的一对大括号所限定，据此可以更好地观察出局部变量的作用域。运行程序，显示效果如图 9.18 所示。

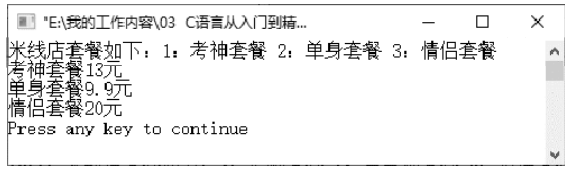


图 9.18　局部变量的作用域

C 语言中，位于不同作用域的变量可以使用相同的标识符，也就是变量名可以相同。如果内层作用域中定义的变量和已经声明的某个外层作用域中的变量有相同名称，在内层中使用这个变量名，此时的变量名表示的是外层变量还是内层变量呢？答案是：内层作用域中的变量将屏蔽外层作用域中的同名变量，直到结束内层作用域为止。这就是局部变量的屏蔽作用。

【例 9.14】局部变量的屏蔽作用（**实例位置：资源包\TM\sl\09\14**）

在本实例中，在不同的语句块内定义了 3 个相同名称的变量，通过输出的变量值来感受局部变量的屏蔽作用效果。

```
#include<stdio.h>

int main()                                  /*主函数 main*/
{
    int iNumber1=1;                         /*定义第 1 个 iNumber1 变量*/
    printf("%d\n",iNumber1);                /*输出第 1 个 iNumber1 的变量值*/
    if(iNumber1>0)
    {
        int iNumber1=2;                     /*定义第 2 个 iNumber1 变量*/
        printf("%d\n",iNumber1);            /*输出第 2 个 iNumber1 的变量值*/
        if(iNumber1>0)
        {
            int iNumber1=3;                 /*定义第 3 个 iNumber1 变量*/
            printf("%d\n",iNumber1);        /*输出第 3 个 iNumber1 的变量值*/
        }
        printf("%d\n",iNumber1);            /*再次输出第 2 个 iNumber1 的变量值*/
    }
    printf("%d\n",iNumber1);                /*再次输出第 1 个 iNumber1 的变量值*/
    return 0;
}
```

（1）在主函数 main 中，定义了第一个整型变量 iNumber1，将其赋值为 1，然后使用 printf 函数进行输出。可以看到，此时 iNumber1 的值为 1。

（2）使用 if 语句进行判断，这里使用 if 语句的目的在于划分出一段语句块。因为位于不同作用域的变量可以使用相同的标识符，所以在 if 语句块中也定义一个 iNumber1 变量，并将其赋值为 2。再次使用 printf 函数输出变量 iNumber1 的操作，观察一下程序的运行结果，发现输出的值为 2。说明值为 2 的 iNumber1 在此作用域中将值为 1 的 iNumber1 屏蔽掉了。

（3）在 if 语句中再次进行嵌套，其嵌套语句中定义第 3 个相同的标识符 iNumber1 变量。为了进行区分，将其赋值为 3。调用 printf 函数输出变量 iNumber1，从程序运行的结果可以看出显示结果为 3。说明值为 3 的 iNumber1 将值为 2 与 1 的两个 iNumber1 都进行了屏蔽。

（4）在最深层嵌套的 if 语句结束之后，使用 printf 函数进行输出，发现此时显示的值为 2。说明此时已经不在值为 3 的 iNumber1 作用域范围内，而在值为 2 的 iNumber1 作用域范围。

（5）当 if 语句结束之后，再次输出 iNumber1 值，此时显示结果为 1，说明已离开了值为 2 的 iNumber1 的作用域范围，不再对值为 1 的 iNumber1 产生屏蔽作用。

运行程序，显示效果如图 9.19 所示。

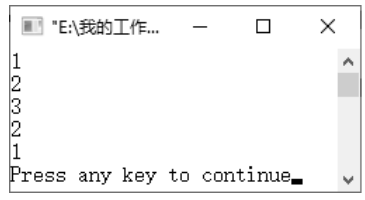


图 9.19　局部变量的屏蔽作用

9.6.2　全局变量

函数中定义的变量都是局部变量。如果一个变量在所有函数外部声明，这个变量就是全局变量。顾名思义，全局变量可以在程序的任何位置进行访问。

全局变量不属于某个函数，而属于整个源文件。如果外部文件要访问它，则要用 extern 关键字进行引用修饰。

全局变量可增加函数间的数据联系。同一文件中的所有函数都能引用全局变量的值，因此如果在一个函数中改变了全局变量的值，就能影响到其他函数，相当于各个函数间有了一个直接传递通道。

例如，有一家全国连锁商店，其商品价格是全国统一的。全国各地有很多这样的连锁商店，当进行价格调整时，应该确保每一家连锁商店的价格是相同的。全局变量就像其中所要设定的价格，而函数就像每一家连锁店，当全局变量进行修改时，函数中使用的该变量都将同步被更改。

【例 9.15】模拟连锁超市价格调整（**实例位置：资源包\TM\sl\09\15**）

在本程序中，使用全局变量模拟连锁店的全国价格调整，使用函数表示连锁店，并在函数中输出一条消息，表示连锁店中的价格。

```
#include<stdio.h>
int iGlobalPrice=100;                              /*设定商品的初始价格*/

void Store1Price();                                /*声明 4 个函数*/
void Store2Price();
void Store3Price();
void ChangePrice();

int main()
{
    printf("原价格 : %d\n",iGlobalPrice);          /*先显示价格改变之前所有连锁店的价格*/
    Store1Price();                                 /*显示 1 号连锁店的价格*/
    Store2Price();                                 /*显示 2 号连锁店的价格*/
    Store3Price();                                 /*显示 3 号连锁店的价格*/
    ChangePrice();                                 /*调用函数，改变连锁店的价格*/
    printf("修改的价格是: %d\n",iGlobalPrice);     /*给出提示，显示修改后的价格*/
    Store1Price();                                 /*显示 1 号连锁店的当前价格*/
    Store2Price();                                 /*显示 2 号连锁店的当前价格*/
    Store3Price();                                 /*显示 3 号连锁店的当前价格*/
    return 0;
}

void Store1Price()                                 /*定义 1 号连锁店的价格函数*/
{
    printf("1 号连锁店的价格 : %d\n",iGlobalPrice);
}
void Store2Price()                                 /*定义 2 号连锁店的价格函数*/
{
    printf("2 号连锁店的价格 : %d\n",iGlobalPrice);
}
void Store3Price()                                 /*定义 3 号连锁店的价格函数*/
{
    printf("3 号连锁店的价格 : %d\n",iGlobalPrice);
```

```
}
void ChangePrice()                                  /*定义更改连锁店价格的函数*/
{
    printf("您想要改价格吗？如果是，改的价格是: ");
    scanf("%d",&iGlobalPrice);
}
```

（1）在程序中，定义了一个全局变量 iGlobalPrice 来表示全国连锁店的初始商品价格，为了可以形成对比，初始化值为 100。还定义了 4 个函数，其中 3 个代表连锁店的价格，例如 Store1Price 代表 1 号连锁店；ChangPrice 函数用来改变全局变量的值，代表对所有连锁店进行调价。

（2）在主函数 main 中，首先将 3 家连锁店的初始价格进行显示，之后通过一条信息提示更改 iGlobalPrice 变量。全局变量被修改后，再次输出将所有连锁店的当前价格。

（3）通过程序的运行结果可以看出，全局变量成为函数间数据联系的渠道，修改一个全局变量后，所有函数中的该变量都会发生改变。

运行程序，显示效果如图 9.20 所示。

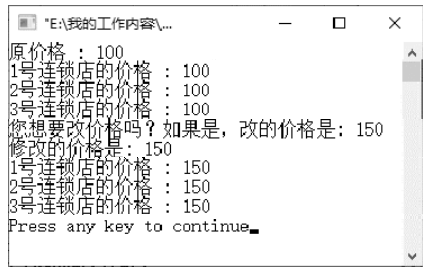


图 9.20　使用全局变量模拟价格调整

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 13：选择桌面主题　小红经常变换电脑桌面，由于收藏的桌面风格太多，他编写了一个程序，定义 style 局部变量，利用这个变量将所有风格输出。假设本例中需要输出风景、蓝天白云、奇幻梦境 3 种风格的壁纸。

训练 14：钱包里还剩多少钱　将钱包中的总金额设为全局变量，定义一个 pay(int number)付款方法，每次付款之后，都会减少钱包中的总金额。付款 3 次之后，钱包中还剩多少钱？

9.7　函 数 应 用

编译系统通常会提供一些标准库函数，以供用户快速调用。不同的编译系统，提供的库函数会略有差异。下面介绍 ANSI C 提供的部分常用的库函数。

9.7.1　数学函数

在程序中经常会进行一些数学相关的运算，这里首先介绍有关数学的常用函数。注意，使用数学函数时，要包括 math.h 头函数，即要在程序开始处添加#include<math.h>。

1．abs 函数

abs 函数的功能是求解整数的绝对值。函数形式如下：

```
int abs(int i);
```

例如，求一个负数绝对值的代码如下：

```
int iAbsoluteNumber;                    /*定义整型变量*/
int iNumber = -12;                      /*定义整型变量，为其赋值-12*/
iAbsoluteNumber=abs(iNumber);           /*调用 abs 函数*/
```

2．labs 函数

labs 函数的功能是求解长整型数的绝对值。函数形式如下：

```
long labs(long n);
```

例如，求一个长整型绝对值的代码如下：

```
long lResult;                           /*定义长整型变量*/
long lNumber = -1234567890L;            /*定义长整型变量，为其赋值-1234567890*/
lResult= labs(lNumber);                 /*调用 labs 函数*/
```

3．fabs 函数

fabs 函数的功能是求解实型数的绝对值。函数形式如下：

```
double fabs(double x);
```

例如，求一个实型数绝对值的代码如下：

```
double fFloatResult;                    /*定义实型变量*/
double fNumber = -1234.0;               /*定义实型变量，为其赋值-1234.0*/
fFloatResult= fabs(fNumber);            /*调用 fabs 函数*/
```

【例 9.16】判断两人相差几岁（**实例位置：资源包\TM\sl\09\16**）

在本实例中，使用 abs 函数计算两人相差的岁数。具体代码如下：

```
#include<stdio.h>                                   /*包含头文件 stdio.h 和 math.h */
#include<math.h>
int main()
{
    int iAbsoluteNumber;                            /*定义整型变量*/
    int iNumber;
    int differ,ab;
    printf("输入年龄：\n");
    scanf("%d,%d",&iAbsoluteNumber,&iNumber);       /*输入两个年龄*/
    differ=iAbsoluteNumber-iNumber;                 /*相差的年龄*/
    ab=abs(differ);                                 /*求相差的绝对值*/
    printf("两个人相差%d 岁 \n",ab);                 /*显示输出相差的年龄*/
    return 0;                                       /*程序结束*/
}
```

运行程序，显示效果如图 9.21 所示。

4．sin 函数

sin 函数的功能是求解正弦。函数形式如下：

```
double sin(double x);
```

例如，求一个角正弦值的代码如下：

```
double fResultSin;                          /*定义实型变量*/
double fXsin = 0.5;                         /*定义实型变量，为其赋值为 0.5*/
fResultSin = sin(fXsin);                    /*使用正弦函数*/
```

5．cos 函数

cos 函数的功能是求解余弦。函数形式如下：

```
double cos(double x);
```

例如，求一个角余弦值的代码如下：

```
double fResultCos;                          /*定义实型变量*/
double fXcos = 0.5;                         /*定义实型变量，为其赋值为 0.5*/
fResultCos = cos(fXcos);                    /*调用余弦函数*/
```

6．tan 函数

tan 函数的功能是求解正切。函数形式如下：

```
double tan(double x);
```

例如，求一个角正切值的代码如下：

```
double fResultTan;                          /*定义实型变量*/
double fXtan = 0.5;                         /*定义实型变量，为其赋值为 0.5*/
fResultTan = tan(fXtan);                    /*调用正切函数*/
```

【例 9.17】求梯形顶角及正切值（**实例位置：资源包\TM\sl\09\17**）

已知一个梯形是由一个正方形和两个等腰直角三角形组成的，求此梯形顶角的度数和它的正切值。

```
#include<stdio.h>
#include<math.h>                            /*包含头文件 math.h*/
int main()
{
    double fResultTan;                      /*用来保存正切值*/
    int Result;
    Result=90+45;                           /*求顶角*/
    fResultTan =tan(Result);                /*调用正切函数*/
    printf("另一个内角是:%d\n",Result);      /*输出顶角结果*/
    printf("正切值是：%f\n",fResultTan);     /*输出顶角的正切值*/
    return 0;
}
```

运行程序，显示效果如图 9.22 所示。

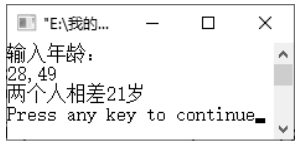


图 9.21　数学库函数使用

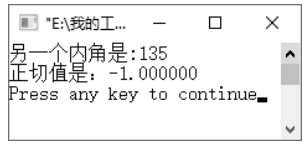


图 9.22　使用三角函数

9.7.2　字符函数

下面要介绍的是 3 个有关字符的函数，使用时要先包含头文件 ctype.h。

1．isalpha 函数

isalpha 函数的功能是检测某个字符是否是字母，如果是字母表中的字母（大写或小写），则返回非 0 值，否则返回 0。函数形式如下：

```
int isalpha(int ch);
```

例如，判断输入的字符是否为字母的代码如下：

```
char c;                                  /*定义字符变量*/
scanf("%c", &c);                         /*输入字符*/
isalpha(c);                              /*调用 isalpha 函数判断输入的字符*/
```

2．isdigit 函数

isdigit 函数的功能是检测某个字符是否是数字，如果是数字，则函数返回非 0 值，否则返回 0。函数形式如下：

```
int isdigit(int ch);
```

例如，判断输入的字符是否为数字的代码如下：

```
char c;                                  /*定义字符变量*/
scanf("%c", &c);                         /*输入字符*/
isdigit(c);                              /*调用 isdigit 函数判断输入的字符*/
```

3．isalnum 函数

isalnum 函数的功能是检测某个字符是否是字母或数字，如果是字母表中的某个字母或某个数字，则函数返回非 0 值，否则返回 0。函数形式如下：

```
int isalnum(int ch);
```

例如，判断输入的字符是否为数字或字母的代码如下：

```
char c;                                  /*定义字符变量*/
scanf("%c", &c);                         /*输入字符*/
isalnum(c);                              /*调用 isalnum 函数判断输入的字符*/
```

【例 9.18】判断输入字符的类型（**实例位置：资源包\TM\sl\09\18**）

在本程序中，向控制台输入字符，利用 if 判断语句和字符函数判断输入的是哪一种类型的字符，然后根据不同的字符类型输出相应提示信息。

```
#include<stdio.h>
#include<ctype.h>                        /*包含头文件 ctype.h*/

void SwitchShow(char c);
```

```
int main()
{
    char cCharPut;                          /*定义字符变量，用来接收输入的字符*/
    char cCharTemp;                         /*定义字符变量，用来接收回车符*/

    printf("请第一次输入一个字符:");          /*消息提示，第一次输入字符*/
    scanf( "%c", &cCharPut);                /*输入字符*/
    SwitchShow(cCharPut);                   /*调用函数进行判断*/
    cCharTemp=getchar();                    /*接收回车符*/

    printf("请第二次输入一个字符:");          /*消息提示，第二次输入字符*/
    scanf( "%c", &cCharPut);                /*输入字符*/
    SwitchShow(cCharPut);                   /*调用函数判断输入的字符*/
    cCharTemp=getchar();                    /*接收回车符*/

    printf("请第三次输入一个字符:");          /*消息提示，第 3 次输入字符*/
    scanf( "%c", &cCharPut);                /*输入字符*/
    SwitchShow(cCharPut);                   /*调用函数判断输入的字符*/

    return 0;                               /*程序结束*/
}

void SwitchShow(char cChar)
{
    if(isalpha(cChar))                      /*判断是否为字母*/
    {
        printf("您输入的是字母  %c\n",cChar);
    }
    if(isdigit(cChar))                      /*判断是否为数字*/
    {
        printf("您输入的是数字  %c\n", cChar);
    }
    if(isalnum(cChar))                      /*判断是否为字母或者数字*/
    {
        printf("您输入的是字母或者数字  %c\n", cChar);
    }
    else                                    /*当字符既不是字母也不是数字时*/
    {
        printf("您输入的是既不是字母也不是数字:%c\n", cChar);
    }
}
```

（1）要使用字符函数，先要引入头文件 ctype.h。

（2）在程序中定义了两个字符变量，cCharPut 用来在程序中接收用户输入的字符，而 cCharTemp 的作用是接收用户按 Enter 键所产生的回车符。

（3）定义 SwitchShow 函数，实现在程序中判断字符的功能。SwitchShow 函数体中，在 3 个 if 语句的判断条件中调用不同的字符函数，根据返回值判断传递的字符参数 cChar 是哪种情况，并给出不同的提示信息。

（4）在 main 函数中调用 getchar 函数，其作用是获取一个字符。因为用户每次输入完毕后都要按 Enter 键进行确定，为防止回车符被当做输入的字符，这里调用 getchar 函数预先提取走回车符。

运行程序，显示效果如图 9.23 所示。

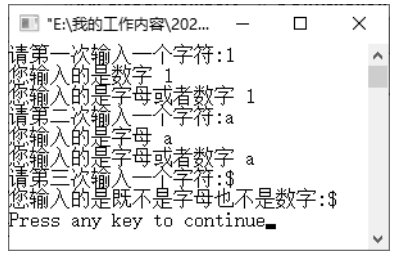


图 9.23　使用字符函数判断输入字符

说明

读者可以尝试将 main 函数中 getchar 函数所在行的代码注销掉，运行程序并观察结果，会发现第二个输入操作会被程序跳过。想一想，为什么。

9.7.3　字符串处理函数

编写程序时，经常要对字符和字符串进行操作，如转换字符大小写、求字符串长度等，这些都可以使用字符串函数来解决。C 语言标准函数库中提供了一系列字符串处理函数，合理、有效地使用这些字符串函数，可以大大提高编程效率。本节将介绍常用的字符串处理函数，使用时要先包含头文件 string.h。

1．strcpy 函数

在字符串处理函数中，strcpy 函数用于复制特定长度的字符串到另一个字符串中。其语法格式如下：

```
strcpy(字符数组 1,字符数组 2)
```

功能：把字符数组 2 中的字符串复制到字符数组 1 中，字符串结束标志“\0”也一同复制。复制过去的内容将覆盖字符数组 1 中的对应内容。

说明

（1）字符数组 1 要有足够的长度，否则无法装下待复制的字符串。

（2）“字符数组 1”必须写成数组名形式；而“字符数组 2”可以是数组名，也可以是一个字符串常量，这时相当于把一个字符串赋予该字符数组。

（3）不能用赋值语句将一个字符串常量或字符数组直接赋给一个字符数组。

【例 9.19】照葫芦画瓢（**实例位置：资源包\TM\sl\09\19**）

在 main 函数体中定义两个字符数组，通过键盘输入获取两个字符串并分别输出，调用 strcpy 函数将源字符数组中的字符串赋值给目标字符数组，最后输出目标字符数组。

```
#include<stdio.h>
#include<string.h>                                    /*包含头文件 string.h*/

int main()
{
    char str1[30],str2[30];
    printf("输入目标字符串：\n");
```

```
    gets(str1);                                          /*输入目标字符串*/
    printf("输入源字符串：\n");
    gets(str2);                                          /*输入源字符串*/
    printf("输出目标字符串：\n");
    puts(str1);                                          /*输出目标字符串*/
    printf("输出源字符串：\n");
    puts(str2);                                          /*输出源字符串*/
    strcpy(str1,str2);                                   /*调用 strcpy 函数实现字符串复制*/
    printf("调用 strcpy 函数进行字符串复制：\n");
    printf("复制字符串之后的目标字符串：\n");
    puts(str1);                                          /*输出复制后的目的字符串*/
    return 0;                                            /*程序结束*/
}
```

运行程序，字符串复制效果如图 9.24 所示。

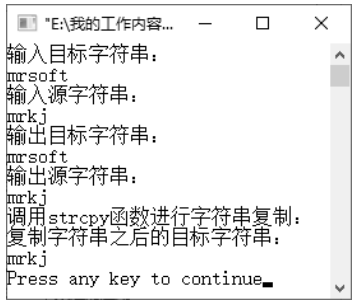


图 9.24　字符串复制

2．strcat 函数

字符串连接就是将一个字符串连接到另一个字符串的末尾，使其组合成一个新的字符串。strcat 函数就具有字符串连接的功能，其语法格式如下：

```
strcat(字符数组 1,字符数组 2)
```

功能：把字符数组 2 中的字符串连接到字符数组 1 的字符串后面，并删去字符数组 1 中原有的串结束标志“\0”。

说明

字符数组 1 应有足够的长度，以保证能装下连接后的字符串。

【例 9.20】显示文件完整路径（**实例位置：资源包\TM\sl\09\20**）

编写程序，接受用户输入的目录和文件名，然后输出文件的完整路径。

在 main 函数体中定义两个字符数组，分别存储输入的目录和文件名，调用 strcat 函数将文件名字符串连接到目录字符串的后面，最后输出目录字符数组。

```
#include<stdio.h>
#include<string.h>
```

```
int main()
{
    char str1[30],str2[30];
    printf("输入目录:\n");
    gets(str1);                          /*输入目录字符串*/
    printf("输入文件名:\n");
    gets(str2);                          /*输入文件名字符串*/

    printf("输出目录:\n");
    puts(str1);                          /*输出目录*/
    printf("输出文件名:\n");
    puts(str2);                          /*输出文件名*/
    strcat(str1,str2);                   /*调用 strcat 函数进行字符串连接*/
    printf("文件全路径:\n");
    puts(str1);                          /*输出连接后的字符串，即文件的完整路径*/

    return 0;                            /*程序结束*/
}
```

运行程序，字符串连接效果如图 9.25 所示。

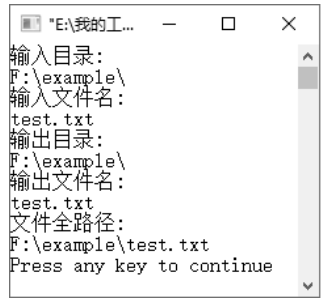


图 9.25　字符串连接

说明

字符串复制实质上是用字符数组 2 中的字符串覆盖字符数组 1 中的字符串，而字符串连接则不存在覆盖的问题，只是单纯地将字符数组 2 中的字符串连接到字符数组 1 中字符串的后面。

3．strcmp 函数

字符串比较就是将一个字符串与另一个字符串从首字母开始，按照 ASCII 码的顺序逐个进行比较。strcmp 函数具有在字符串间进行比较的功能，其语法格式如下：

```
strcmp(字符数组 1,字符数组 2)
```

功能：按照 ASCII 码顺序比较两个数组中的字符串，并返回比较结果。两个字符串比较时，若出现不同的字符，则以第一个不同字符的比较结果作为整个比较的结果。返回值如下：

- ☑ 字符串 1=字符串 2，返回值为 0。
- ☑ 字符串 1>字符串 2，返回值为正数。
- ☑ 字符串 1<字符串 2，返回值为负数。

注意

字符串比较不能使用关系运算符，也不能使用赋值运算符。因此，下面的语句是错误的：

```
if(str[2]==mingri)…
str[2]=mingri;…
```

【例 9.21】模拟银行取钱（**实例位置：资源包\TM\sl\09\21**）

在自动取款机上取钱时，需要输入密码，密码正确方能取到钱。本实例中，正确的密码为 574824，

只有 3 次密码输入机会，如果 3 次都输错了，就提示“请到人工处办理解锁”。

```
#include<stdio.h>
#include<string.h>

int main()
{
    char password[20] = {"574824"};                          /*定义变量，保存正确的密码字符串*/
    char pwstr[20];                                          /*定义变量，保存用户输入的密码字符串*/
    int i=1;
    while(i <= 3)                                            /*循环执行 3 次密码输入*/
   {
        printf("输入密码字符串:\n");
        gets(pwstr);                                         /*输入密码字符串*/
            if(strcmp(password,pwstr))                       /*比较两个密码，如果不相同*/
            {
                printf("第%d 次，密码字符串输入错误！\n",i);  /*提示密码输入错误*/
            }
            else                                             /*如果相同*/
            {
                printf("密码正确，请选择服务！\n");           /*输出下一步业务提醒*/
                break;                                       /*退出*/
            }
        i++;
    }
    if(i == 4)
    {
        printf("输入字符串错误 3 次！请到人工处办理解锁\n");  /*如果密码输入错误多于 3 次*/
    }
    return 0;                                                /*程序结束*/
}
```

运行程序，字符串比较效果如图 9.26 所示。

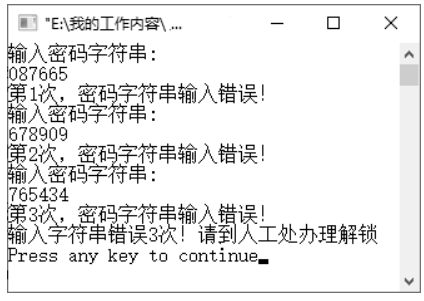


图 9.26　字符串比较

4．strupr 函数和 strlwr 函数

字符串的大小写转换需要使用 strupr 和 strlwr 函数。strupr 函数的语法格式如下：

```
strupr(字符数组名)
```

功能：将数组中存放的字符串的小写字母转换成大写字母，其他字母不变。

strlwr 函数的语法格式如下：

```
strlwr(字符数组名)
```

功能：将数组中存放的字符串的大写字母转换成小写字母，其他字母不变。

【例 9.22】字符串大小写转换（**实例位置：资源包\TM\sl\09\22**）

将张三的邮箱地址 ZhangSan@MRSOFT.COM 全部转换为小写或大写。

在 main 函数体中定义两个字符数组，分别用来存储要转换的字符串和转换后的字符串，然后根据用户输入的操作指令，判断应调用 strupr 函数还是 strlwr 函数，进行大小写转换。

```
#include<stdio.h>
#include<string.h>

int main()
{
    char text[20]="ZhangSan@MRSOFT.COM",change[20];
    printf("原字符串为:%s\n",text);                          /*输出要转换的字符串*/
    strcpy(change,text);                                    /*复制要转换的字符串*/
    strlwr(change);                                         /*字符串转换为小写*/
    printf("转换成小写字母的字符串为:%s\n",change);          /*输出转换后的字符串*/
    strupr(change);                                         /*字符串转换为大写*/
    printf("转换成大写字母的字符串为:%s\n",change);          /*输出转换后的字符串*/
    return 0;                                               /*程序结束*/
}
```

运行程序，字符串大小写转换效果如图 9.27 所示。

图 9.27　字符串大小写转换

5．strlen 函数

有时需要知道字符串的长度，虽然通过循环判断字符串结束标志“\0”也能获得字符串的长度，但毕竟实现起来相对烦琐。这时可以使用 strlen 函数来计算字符串的长度。

strlen 函数的语法格式如下：

```
strlen(字符数组名)
```

功能：计算数组中存放的字符串的实际长度（不含字符串结束标志“\0”）。

【例 9.23】判断用户输入的密码是否为 6 位（**实例位置：资源包\TM\sl\09\23**）

在本实例中的 main 函数体中定义一个字符数组，用来存储用户输入的字符串，然后调用 strlen 函数计算字符串长度，并使用 if…else...语句判断密码长度是否等于 6。

```
#include<stdio.h>
#include<string.h>
```

```
int main()
{
        char text[50];                                  /*定义字符数组，保存密码字符串*/
        printf("输入一个密码:\n");
        scanf("%s", &text);                             /*输入用户密码*/

        if(strlen(text)==6)                             /*计算密码长度并判断是否等于 6*/
            printf("输入密码是 6 位\n");
        else
            printf("输入密码不是 6 位\n");
        return 0;                                       /*程序结束*/
}
```

运行程序，输出效果如图 9.28 所示。

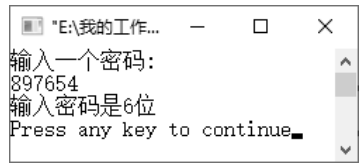


图 9.28　计算密码长度

编程训练（答案位置：资源包\TM\sl\09\编程训练\）

训练 15：判断地铁两站间行车时间　已知某地铁线路有 5 站，将从始发站发车行驶到各个站的时间集合{5,10,15,20,25}保存成一个一维数组 a，编写程序，判断任意两站之间需要行车多长时间。运行结果如下：

```
请任意输入两站：
3 1
两站之间行车需要 10 分钟
```

训练 16：寻找座位号　使用 isdigit 函数判断输入的座位号是否是数字，运行效果如下：

结果一：

```
请输入您的位置号：
------------------------------------------------------------
|    对不起，您输入的座位号格式不正确    |
|            不能找到您的位置            |
------------------------------------------------------------
```

结果二：

```
请输入您的位置号：5
→*→*→*→*→*→*→*→*→*→*→*→*→*→*
  您的位置号是 5
  您输入的格式正确，请坐....
→*→*→*→*→*→*→*→*→*→*→*→*→*→*
```

训练 17：显示天气预报　使用 strcat 函数，输出 2021 年 1 月 23 日的实时天气预报。输出结果如下：

```
2021 年 1 月 23 日 晴, 20℃~7℃, 微风转西风 3~4 级
08:00   晴 13℃   微风
12:00   晴 19℃   微风
16:00   晴 18℃   微风
20:00   晴 15℃   微风
```

```
00:00　晴 12℃　微风
```

训练 18：更新公告　利用 strcpy 函数，将包子店的招牌“包子一元一个”换成“包子壹圆壹个”。输出结果如下：

```
原来的招牌内容是：
      包子一元一个
经过处理之后的招牌内容是:
      包子壹圆壹个
```

9.8　实践与练习

（答案位置：资源包\TM\sl\09\实践与练习\）

综合练习 1：模拟 12306 售票系统　12306 售票系统全国联网，数据同步。因此，每售出一张票，全国各地的系统显示都会减少一张票数。利用全局变量模拟 12306 抢票系统，输出效果如下：

```
始发地：上海　目的地：长春　时间：2021 年 4 月 10 日 16：20 出发
3 个城市剩余的票数分别为：
上海的 12306 系统剩余票数: 99 张
北京的 12306 系统剩余票数: 99 张
深圳的 12306 系统剩余票数: 99 张
我抢到一张票数之后剩余票数：98
我抢到一张票之后 3 个城市剩余的票数分别为：
上海的 12306 系统剩余票数: 98 张
北京的 12306 系统剩余票数: 98 张
深圳的 12306 系统剩余票数: 98 张
```

综合练习 2：为和尚写诗　自定义一个 poetry 函数，为和尚写一首诗。诗句如下：

```
空门有路不知处
头白齿黄犹念经
何年饮着声闻酒
迄至如今醉未醒
```

综合练习 3：一棵松树的梦　在源文件中定义一个全局变量 pinetree，并为它赋值。再定义一个 christmastree 函数，在这个函数里定义名称为 pinetree 的局部变量，并输出，最后在主函数中调用 christmastree 函数，并输出全局变量 pinetree 的值。输出结果如下：

```
下雪了……

============开始做梦……==============

挂上彩灯、礼物……我变成一棵圣诞树@^.^@

============梦醒了……================

我身上落满雪花,我是一棵松树 -_-
```

综合练习 4：确定女主角　某导演有一个剧本，需要寻找演员来饰演对应的角色。利用函数的实参和形参知识编写代码，实现为剧本选女主角的功能。输出结果如下：

```
导演选定女主角是：Lucy
→*→*→*→*→*→*→*→*→*→*→*→*→*
     Lucy 开始参演李美丽角色
→*→*→*→*→*→*→*→*→*→*→*→*→*
```

综合练习 5：为 C 语言归类　要想在腾讯课堂上查看某个课程，一般需要知道其分类。例如，C 语言的所属类别从大到小依次为“IT—互联网—编程语言—C”，利用函数嵌套找到 C 语言课程。效果如下：

```
（1）找到 IT 分类
（2）IT 分类中找到互联网分类
（3）互联网分类中找到编程语言分类
（4）编程语言分类找到 C 语言课程
```

综合练习 6：递归求年龄　甲乙丙丁戊 5 个人坐在一起聊天。大家猜戊的年龄，他说比丁大 2 岁；问丁的岁数，他说比丙大 2 岁；问丙的岁数，他说比乙大 2 岁；问乙的岁数，他说比甲大 2 岁；问甲的岁数，他说自己 10 岁。编写程序求戊的年龄。输出结果如下：

```
--------------------------------
   戊的年龄是：18 岁
--------------------------------
```

综合练习 7：将美元兑换成人民币　编程实现美元兑换人民币功能。美元与人民币之间的汇率经常变更，这里按 1 美元等于 6.28 元人民币计算。输出结果如下：

```
您要兑换的美元金额：500

**************************
*  兑换成人民币金额是：3140.00  *
**************************
```

综合练习 8：你的心跳正常吗？　利用把函数作为参数使用，编写程序判断输入的心跳数是否是正常心跳数。当心跳数在 60～100 次/分钟，显示心跳数正常，运行结果如图 9.29 所示；当心跳数大于 100 次/分钟或小于 60 次/分钟时，就显示心跳不正常，运行结果如图 9.30 所示。

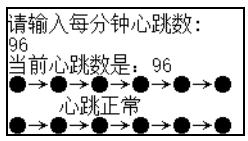


图 9.29　心跳正常

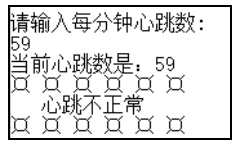


图 9.30　心跳不正常

综合练习 9：注册明日学院 VIP 账号　在明日学院注册账号时，账号长度要求为 4～12 位，使用 strlen 函数判断注册的明日学院账号是否符合要求。输出结果如下：

```
请输入您想注册的明日学院账号:
mingrisoft
注册成功
```

综合练习 10：判断车牌归属地　根据车牌号可以知道一辆车的归属地。编写程序，利用 strcmp 函数判断车牌的归属地。运行结果如下：

```
车牌号归属地查询：

津 A·12345 这个车牌号的归属地是：天津
沪 A·23456 这个车牌号的归属地是：上海
京 A·34567 这个车牌号的归属地是：北京
```