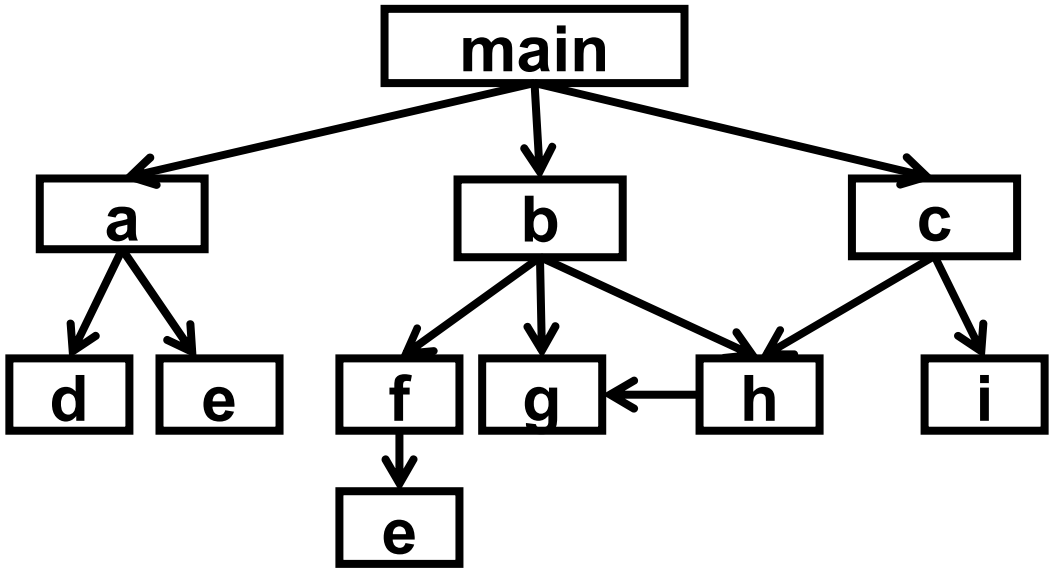
第7章 用函数实现模块化程序设计

7.1为什么要用函数

- 问题：
  - ◆前面的程序基本不超过**20**行，一般仅实现比较单一的功能，但如果程序的功能比较多，规模比较大，代码就很多，若把所有代码都写在**main**函数中，就会使主函数变得庞杂、头绪不清，阅读和维护变得困难
  - ◆有时程序中要多次实现某一功能，就需要多次重复编写实现此功能的程序代码，这使程序冗长，不精炼
- 解决的方法：用模块化程序设计的思路
  - ◆采用“组装”的办法简化程序设计的过程
  - ◆事先编好一批实现各种不同功能的函数
  - ◆把它们保存在函数库中，需要时直接用
  - ◆函数（**function**）就是功能
  - ◆每一个函数用来实现一个特定的功能
  - ◆函数的名字应反映其代表的功能

7.1为什么要用函数

- 在设计一个较大的程序时，往往把它分为若干个程序模块，每一个模块包括一个或多个函数，每个函数实现一个特定的功能
- C程序可由一个主函数和若干个其他函数构成
- 主函数调用其他函数，其他函数也可以互相调用
- 同一个函数可以被一个或多个函数调用任意多次



- 库函数
- 用户自定义函数

利用函数，可以

- 减少重复编码
- 实现模块化程序设计

7.1为什么要用函数

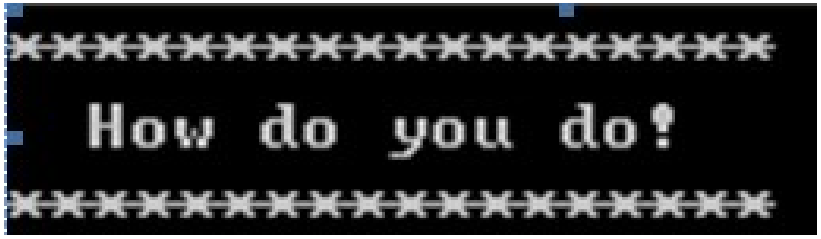
例**7.1** 输出以下的结果，用函数调用实现。

****

How do you do!

****

- 解题思路：
  - ◆在输出的文字上下分别有一行“*”号，显然不必重复写这段代码，用一个函数**print_asterisk**来实现输出一行“*”号的功能。
  - ◆再写一个**print_msg**函数来输出中间一行文字信息
  - ◆用主函数分别调用这两个函数




```
#include <stdio.h>
int main()
{  void print_asterisk();
   void print_msg();
   print_ asterisk();   print_ msg();
   print_ asterisk();
   return 0;
}
void print_asterisk ()
{  printf("**********\n"); }
void print_msg ()
{  printf("  How do you do!\n");  }
```

声明函数

定义函数

输出18个*

输出一行文字

- 说明：

(1) 一个C程序由一个或多个程序模块组成，每一个程序模块作为一个源程序文件。对较大的程序，一般不希望把所有内容全放在一个文件中，而是将它们分别放在若干个源文件中，由若干个源程序文件组成一个**C**程序。这样便于分别编写、分别编译，提高调试效率。一个源程序文件可以为多个**C**程序共用。

(2) 一个源程序文件由一个或多个函数以及其他有关内容（如预处理指令、数据声明与定义等）组成。一个源程序文件是一个编译单位，在程序编译时是以源程序文件为单位进行编译的，而不是以函数为单位进行编译的。

(3) C程序的执行是从**main**函数开始的，如果在**main**函数中调用其他函数，在调用后流程返回到**main**函数，在**main**函数中结束整个程序的运行。

(4) 所有函数都是平行的，即在定义函数时是分别进行的，是互相独立的。一个函数并不从属于另一个函数，即函数不能嵌套定义。函数间可以互相调用，但不能调用**main**函数。**main**函数是被操作系统调用的。

(5) 从用户使用的角度看，函数有两种。

◆库函数，它是由系统提供的，用户不必自己定义而直接使用它们。应该说明，不同的**C**语言编译系统提供的库函数的数量和功能会有一些不同，当然许多基本的函数是共同的。

◆用户自己定义的函数。它是用以解决用户专门需要的函数。

(6) 从函数的形式看，函数分两类。

① 无参函数。无参函数一般用来执行指定的一组操作。无参函数可以带回或不带回函数值，但一般以不带回函数值的居多。

② 有参函数。在调用函数时，主调函数在调用被调用函数时，通过参数向被调用函数传递数据，一般情况下，执行被调用函数时会得到一个函数值，供主调函数使用。

7.2 怎样定义函数

7.2.1 为什么要定义函数

7.2.2 定义函数的方法

7.2.1 为什么要定义函数

- **C**语言要求，在程序中用到的所有函数，必须“先定义，后使用”
- 指定函数名字、函数返回值类型、函数实现的功能以及参数的个数与类型，将这些信息通知编译系统。
- 指定函数的名字，以便以后按名调用
- 指定函数类型，即函数返回值的类型
- 指定函数参数的名字和类型，以便在调用函数时向它们传递数据
- 指定函数的功能。这是最重要的，这是在函数体中解决的，解决该功能是如何实现的。
- 对于库函数，程序设计者只需用**#include**指令把有关的头文件包含到本文件模块中即可
- 程序设计者需要在程序中自己定义想用的而库函数并没有提供的函数

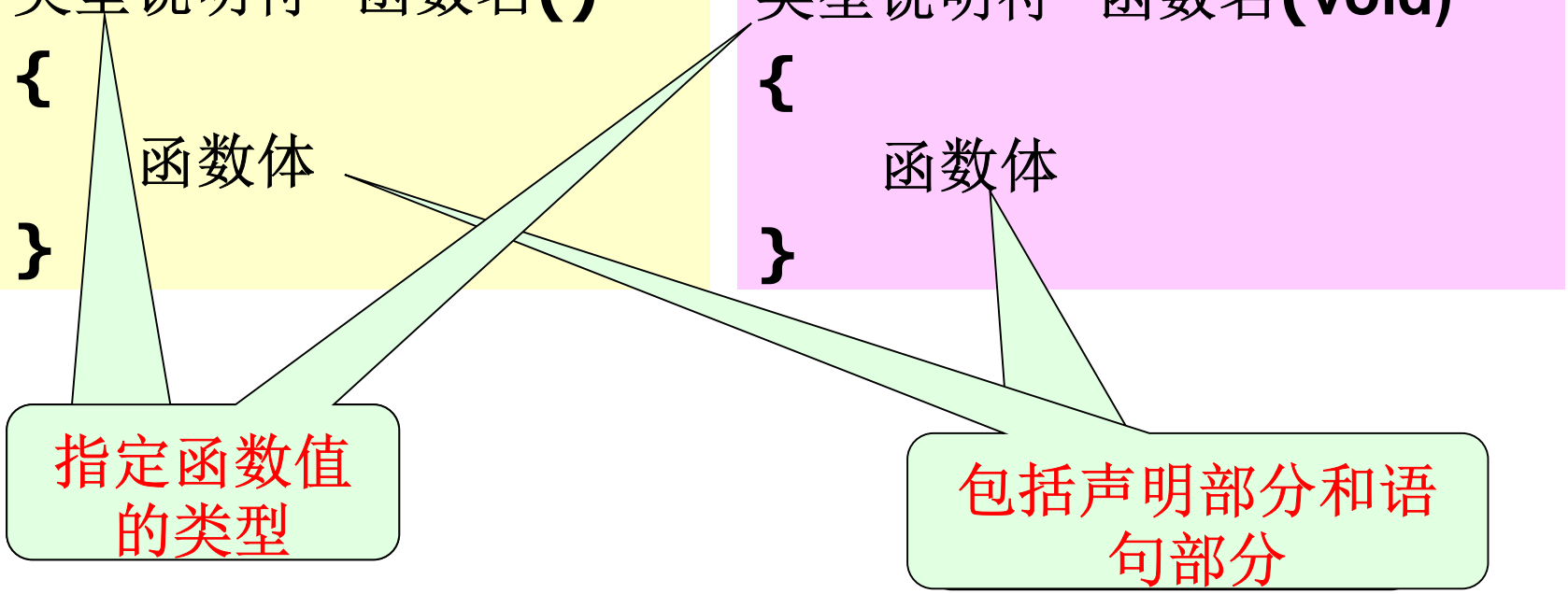
7.2.2 定义函数的方法

1.定义无参函数

定义无参函数的一般形式为:

```
类型说明符  函数名()
{
    函数体
}
```

```
类型说明符  函数名(void)
{
    函数体
}
```



7.2.2 定义函数的方法

2.定义有参函数

定义有参函数的一般形式为**:**

类型说明符 函数名**(**形式参数表列**)**

{

函数体

}

3. 定义空函数

定义空函数的一般形式为**:**

类型说明符 函数名**()**

{ }

- 先用空函数占一个位置，以后逐步扩充
- 好处：程序结构清楚，可读性好，以后扩充新功能方便，对程序结构影响不大

7.3 调用函数

7.3.1函数调用的形式

- 函数调用的一般形式为：
  函数名(实参表列)
- 如果是调用无参函数，则“实参表列”可以没有，但括号不能省略，这里括号()是函数调用的运算符
- 如果实参表列包含多个实参，则各参数间用逗号隔开

- 按函数调用在程序中出现的形式和位置来分，可以有以下3种函数调用方式：

1．函数调用语句——表达式语句的一种

- 把函数调用单独作为一个语句
  如print_asterisk();
- 这时不要求函数带回值，只要求函数完成一定的操作

7.3.1函数调用的形式

- 函数调用的**3**种方式：

２．函数表达式

- 函数调用出现在另一个表达式中

  如**c=max(a,b);**

- 这时要求函数带回一个确定的值以参加表达式的运算

３．函数参数

- 函数调用作为另一函数调用时的实参

  如**m＝max(a,max(b,c));**

- 其中**max(b,c)**是一次函数调用，它的值作为**max**另一次调用的实参

7.3.2 函数调用时的数据传递

1.形式参数和实际参数

- ◆在调用有参函数时，主调函数和被调用函数之间有数据传递关系
- ◆定义函数时函数名后面的变量名称为“形式参数”（简称“形参”）
- ◆主调函数中调用一个函数时，函数名后面参数称为“实际参数”（简称“实参”）
- ◆ 实际参数可以是常量、变量或表达式

2. 实参和形参间的数据传递

- ◆在调用函数过程中，系统会把实参的值传递给被调用函数的形参；或者说，形参从实参得到一个值
- ◆该值在函数调用期间有效，可以参加被调函数中的运算

7.3.2 函数调用时的数据传递

例**7.2** 输入两个整数，要求输出其中值较大者。要求用函数来找到大数。

- 解题思路：
  (1)函数名应是见名知意，今定名为**max**
  (2) 由于给定的两个数是整数，返回主调函数的值（即较大数）应该是整型
  (3)max函数应当有两个参数，以便从主函数接收两个整数，因此参数的类型应当是整型
- 编写**max**函数：

```
int max(int x,int y)
{
    int z;
    z=x>y?x:y;
    return z;
}
```

7.3.2 函数调用时的数据传递

在max函数上面，再编写主函数

```
#include <stdio.h>
int main()
{
   int max(int x,int y);
   int a,b,c;
   printf("Input 2 integer numbers: ");
   scanf("%d%d",&a,&b);
   c=max(a,b);
   printf("max is %d\n", c);
}
```

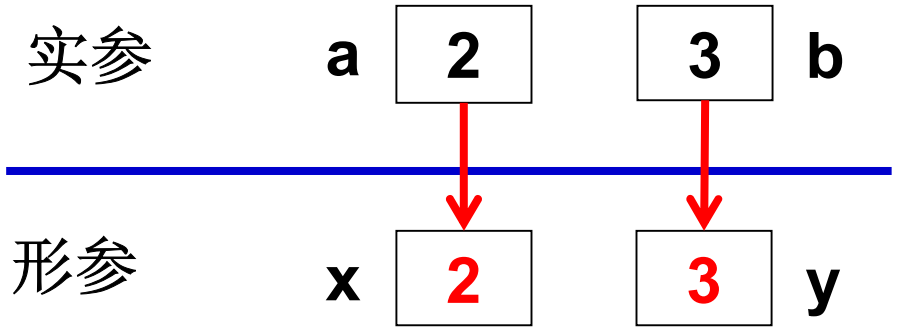
```
Input 2 integer numbers:-71 18
max is 18
```

实参可以是常量、变量或表达式

7.3.2 函数调用时的数据传递

```
c=max(a,b);        （main函数）
```

```
int max(int x, int y)   （max函数）
{
    int z;
    z=x>y?x:y;
    return z;
}
```



在定义函数中指定的形参，在未出现函数调用时，它们并不占内存中的存储单元。在发生函数调用时，函数**max**的形参被临时分配内存单元。形参和实参占用不同的存储单元，二者的值互不影响。

7.3.4. 函数的返回值

➢通常，希望通过函数调用使主调函数能得到一个确定的值，这就是函数值(函数的返回值)

(1) 函数的返回值是通过函数中的**return**语句获得的。

◆一个函数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

(2) 函数值的类型。应当在定义函数时指定函数值的类型

(3)在定义函数时指定的函数类型一般应该和**return**语句中的表达式类型一致

◆ 如果函数值的类型和**return**语句中表达式的值不一致，则以函数类型为准

7.3.4. 函数的返回值

函数调用过程中可能发生数据类型的默认转换

1. 参数传递时，形参和实参类型不一致

如上例，将**max**中的**int**改为**float**，**main**函数中**a, b, c**的类型不变

max函数不变，将**main**函数中的**a, b, c**的数据类型改为**float**

2. 返回函数值时，返回值与函数类型不一致

仍以上例为例，将**max**函数参数类型改为**float**，但函数返回值类型仍为**int**

7.4对被调用函数的声明和函数原型

- 在一个函数中调用另一个函数需要具备如下条件：

  (1) 被调用函数必须是已经定义的函数（是库函数或用户自己定义的函数）

  (2) 如果使用库函数，应该在本文件开头加相应的**#include**指令

  (3) 如果使用自己定义的函数，而该函数的位置在调用它的函数后面，应该声明

7.4对被调用函数的声明和函数原型

例**7.4** 输入两个实数，用一个函数求出它们之和。

解题思路:

- 用**add**函数实现。首先要定义**add**函数，它为**float**型，它应有两个参数，也应为**float**型。特别要注意的是：要对**add**函数进行声明。
- 分别编写**add**函数和**main**函数，它们组成一个源程序文件
- **main**函数的位置在**add**函数之前
- 在**main**函数中对**add**函数进行声明

```
2.2  5.1
2.20+5.10=7.30
```

```
#include <stdio.h>
int main()
{ float add(float x, float y);
  float a,b,c;
  scanf("%f%f",&a,&b);
  c=add(a,b);
  printf("%.2f+%.2f=%.2f\n",c);
  return 0;
}
```

对add函数声明

只差一个分号

调用add函数

```
float add(float x,float y)
{ float z;
  z=x+y;
  return z;
}
```

求两个实数之和，
函数值也是实型

定义add函数

- 函数原型的一般形式有两种：

  如 **float add(float x, float y);**

  float add(float, float);

- 函数声明可以放在一个函数内部（局部声明），此时只有该函数可以调用此函数
- 原型声明可以也放在文件的开头（全局声明），这时所有函数都可以调用此函数。
- 函数定义与函数声明的区别
  - 函数定义是函数功能的完整实现，包括指定函数名，函数值类型、形参类型、函数体等，它是一个完整的、独立的函数单位
  - 函数声明是把函数的名字、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

- 函数(function)
  - ◆函数是实现特定功能的一组代码序列
  - ◆函数是对实现特定功能的具体操作的抽象
- 函数定义

```
T func(T1 p1, T2 p2, …, Tn pn)
{
    ……;
    return value_of_T_type;
}
```

- 函数声明(函数原型)

```
T func(T1 p1, T2 p2, …, Tn pn );
```