重点 **第七章 函数** 难点

本章导读

C源程序是由函数组成的。虽然在前面各章的程序中都只有一个主函数main()，但实用程序往往由多个函数组成。函数是C源程序的基本模块，通过对函数模块的调用实现特定的功能。C语言不仅提供了极为丰富的库函数还允许用户建立自己定义的函数。用户可把自己的算法编成一个个相对独立的函数模块，然后用调用的方法来使用函数。

由于采用了函数模块式的结构，C语言易于实现结构化程序设计。使程序的层次结构清晰，便于程序的编写、阅读、调试。

《C程序设计基础》(Visual C++ 6.0环

第七章 函数

《C程序设计基础》(Visual C++ 6.0环

7.1 问题的提出

知识点回顾

1、前面已经提到，C程序是由函数构成的。每个程序由一个或多个函数组成，其中有且仅有一个main( )函数。

2、一个可执行的C语言程序总是从 main函数开始执行，而不论其在整个程序中的位置如何。

3、我们已经学习过一些C语言提供的库函数，如标准输入、输出函数printf( )、scanf( )；数学函数sqrt( )、pow( )等。

《C程序设计基础》(Visual C++ 6.0环

7.1 问题的提出

我们来看一个程序段：

```
int  x = 1, i;
for(i=1; i<=10; i++)
    x = x * i;
```

该程序段的功能是求10!

编程序求组合函数 $C_m^n = \frac{m!}{n!*(m-n)!}$.

《C程序设计基础》(Visual C++ 6.0环

7.1 问题的提出

```
main( )
{
    int  m, n, x=1, y=1, z=1, i, cmn;
    scanf("%d%d", &m, &n);
    for(i = 1; i <= m; i++)     /* 求 m! */
        x = x * i;
    for(i = 1; i <= n; i++)     /* 求 n! */
        y = y * i;
    for(i = 1; i <= m-n; i++)   /* 求 (m-n)! */
        z = z * i;
    cmn = x / (y * z);
    printf("组合函数值 = %d\n", cmn);
```

问题：

是否可以减少相似类型的操作而使程序变得更加简洁一些?

```
int jc(int);
```

函数声明

```
void main(void) /* 求组合函数的解 */
{
   int  m, n, cmn;
   scanf("%d%d", &m, &n);
   cmn = jc(m) / ( jc(n) * jc(m-n));
   printf("cmn = %d \n ", cmn);
}
```

函数调用

```
int jc(int  n) /* 求某个数 n 的阶乘 */
{
   int  i, x=1;
   for(i = 1; i <= n; i++)
       x = x * i;
   return (x);
}
```

函数定义

7.1 问题的提出

使用函数的目的

目的1:为了方便地使用其他人编写的代码，就像我们调用系统提供的库函数一样。

目的2:为了在新的程序中使用自己编写的代码,避免重复劳动。

目的3:为了实现结构化程序设计的基本思想（自顶向下、逐步求精），即将大型程序分割成小块程序，把问题分解为若干较小的、功能简单的、相互独立又相互关联的模块来进行设计。

7.2 函数定义

三个重要的概念

函数定义：即定义函数的功能。未经定义的函数不能使用。

函数调用：即执行函数,函数调用时,程序跳转到被调用函数的第一句开始执行,执行至被调用函数的最后一句,然后程序返回调用该函数处。

函数申明：即通知编译系统该函数已经定义过了。

7.2 函数定义

函数定义的格式如下:

```
[函数类型]  函数名([形参表])
{
  函数体语句
}
```

函数首部

函数体

说明:

(1) 函数由“函数首部”和“函数体”两部分构成。

(2) 其中函数首部由三部分——“函数类型”、“函数名”、“形参表” 组成。

(3) 函数体是由一对大括号括起来的若干语句。

7.2 函数定义

续上

(4) 函数类型：即函数返回值的类型说明符，表示被调用函数返回给主调函数的数据的类型。

- 函数无返回值时，“函数类型”的关键字为void。
- 函数有返回值时，“函数类型”的关键字为char、int、float、指针类型、结构体类型等。
- 当函数类型缺省时，C系统认为此时的返回类型是int型。

(5) 形参表（即形式参数）：当发生函数调用时，由主调函数传递给被调函数的参数。

- 函数有参数时,“参数表”里有一个或若干个参数,参数与参数之间用逗号隔开，称为有参函数。
- 函数没有参数时，“参数表”空，称为无参函数。

7.2 函数定义

函数定义时，需仔细考虑以下几点:

（1）函数是否有返回值类型？（即判断函数调用结束后是否需要向主调函数返回什么数据）

（2）函数是否有参数？（即判断被调函数使用的数据是否是由主调函数传递过来的）

（3）取一个什么样的函数名有利用程序的阅读?

下面我们将从函数是否有返回值、是否有参数等方面阐述函数定义的方法。

7.2 函数定义

【例1】定义一个fun函数打印以下图形。

```
*
**
***
****
```

分析：该图形的打印可以使用printf函数直接完成，因此不需要main函数向fun函数传递什么信息，fun函数调用结束后也不需要向main函数返回什么。

无返回值

无参数

```
void fun(void)
{
    printf("*\n");
    printf("**\n");
    printf("***\n");
    printf("****\n");
}
```

```
void fun(void)
{    …… }

main()
{
  fun( );
}
```

7.2 函数定义

【例2】给定两个整数，定义一个fun函数，其功能是选出这两个数中的较大值。

分析：该函数的定义有以下几种情况

（1）如果待判断的两个数是在fun函数内部输入的，则该函数设计为无参类型。

（2）如果待判断的两个数是在main函数输入的，则需要通过参数的形式传递给fun函数使用，则该函数设计为有参类型。

（3）如果选出的较大值在fun函数中打印，则该函数设计为无返回值形式。

（4）如果选出的较大值在main函数中打印，则该函数设计为有返回值形式。

7.2 函数定义

两个整数在fun函数内部赋值——无参函数

情况一：结果打印在fun函数中完成

```
void fun(void)
{
  int  x=10, y=20
  if(x > y)
       z = x;
  else
       z = y;
  printf ("max=%d", z);
}
```

无参数 / 无返回值

情况二：结果打印在main函数中完成

```
int fun(void)
{
  int x=10, y=2
  if(x > y)
       z = x;
  else
       z = y;
  return(z);
}
```

无参数 / 有返回值

完整程序 7.2 函数定义

情况一：无参, 无返回值

```
void fun(void)
{
  int  x=10, y=20, z;
  if(x > y)  z = x;
  else       z = y;
  printf ("max=%d", z);
}
main( )
{
  fun( );
}
```

函数调用,不传递参数

情况二：无参, 有返回值

```
int fun(void)
{
  int  x=10, y=20, z;
  if(x > y)  z = x;
  else       z = y;
  return(z);
}
main( )
{  int
  m = fun( );
  printf("max=%d", m);
}
```

函数调用,无参数传递，但有返回值

7.2 函数定义

两个整数在main函数内部赋值——有参函数

情况一：结果打印在fun函数中完成

```
void fun(int x, int y)
{
  int  z;
  if(x > y)
       z = x;
  else
       z = y;
  printf ("max=%d", z);
}
```

有参数 / 无返回值

情况二：结果打印在main函数中完成

```
int fun(int x, int y)
{
  int  z;
  if(x > y)
       z = x;
  else
       z = y;
  return(z);
}
```

有参数 / 有返回值

完整程序 7.2 函数定义

情况一：有参, 无返回值

```
void fun(int x, int y)
{
  int  z;
  if(x > y)  z = x;
  else       z = y;
  printf ("max=%d", z);
}
main( )
{
  int  x=
  fun(x, y);
}
```

函数调用,要传递参数

情况二：有参, 有返回值

```
int fun(int x, int y)
{
  int  z;
  if(x > y)  z = x;
  else       z = y;
  return(z);
}
main( )
{  int  x=
  m = fun(x, y);
  printf("max=%d", m);
}
```

函数调用,传递参数,且有返回值

7.2 函数定义

结论:

（1）当被调用函数使用的数据全部由它自己产生时，函数应定义为无参类型（void型）；反之，当被调用函数使用的数据需要由主调函数提供时，函数应定义为有参类型。

（2）当被调用函数运行结束后，不需要向主调函数返回任何信息，函数定义为无返回值类型（void型）；反之，如果需要向主调函数返回某个数值，则函数定义为有返回值类型。

7.2 函数定义

续上

（3）当函数定义为有返回值类型时，函数体内必须有return语句，并且必须写成“return(表达式)”的形式，其中表达式值的类型应与返回值类型一致。

（4）当函数定义为无返回值类型时，函数体内可以缺省return语句，即使有return语句，也只能写成“return ;”的形式，其后没有表达式。

7.3 函数声明

当函数定义在函数调用之后时，需要进行函数声明。函数声明的形式如下:

函数类型 函数名(类型1，类型2，…，类型n);

说明:

（1）函数声明时的“函数类型”、“函数名”、“形参表里参数的类型、个数、顺序”必须与函数定义时保持一致，否则出错。

（2）函数声明时的形参名可以省略，但形参类型不能省略。

（3）函数声明必须以“分号”结束。

（4）函数声明可以有多次，但函数定义只能有一次。

7.3 函数声明

```
void max(int a,int b);
main( )
{
  int  x=10, y=10;
  max(x,y);
}
void max(int x,int y)
{
  int  z;
  if(x > y)  z = x;
  else       z = y;
  printf ("max=%d", z);
}
```

函数声明

参数表里包含参数类型及参数名

函数调用

函数定义

7.4 函数调用

7.4.1 函数调用的格式

函数调用的格式如下:

函数名([实参表])

当没有参数传递时，实参表为空

说明:

(1) 函数调用时的参数表称为“实参表”，即实际参数，其作用是，当发生函数调用时，主调函数的实参将其数值传递给被调函数的形参。

(2) 当调用一个无参函数时，*小括号为空*，如“fun( );”。

(3) 当调用一个有参函数时，实参表的每一项为一个表达式，*没有数据类型说明符*，如“max(x, y);”

7.4.1 函数调用的格式

续上

(4) 有参函数调用时，实参要向形参传递数值，为实现正确的数值传递，必须保证实参与形参的三个一致——类型一致、个数一致、顺序一致。

重要

函数调用过程中的数值传递

主调函数 —— 实参向形参传值 ——> 被调函数

被调函数 —— 通过return语句返回数值 ——> 主调函数

7.4.2 函数调用的三种方式

函数调用的三种方式: 重点

方式一：函数语句。C语言中的函数可以作为一条独立的语句进行调用，这种方式不规定函数一定要有返回值。

方式二：函数表达式。函数作为表达式的一项，出现在表达式中，用函数的返回值参与表达式运算。这种方式要求函数必须有返回值。

方式三：函数实参。函数作为其它函数实参的形式出现，这种方式是把该函数的返回值作为其它函数的实参进行传值，因此要求该函数必须有返回值。

7.4.2 函数调用的三种方式

【例3】在主函数中输入两个整数值，定义一个fun函数，其功能是选出这两个数中的较大值。

方法一：“函数语句”调用形式

```
void fun(int x, int y)
{  int  z;
   if(x > y)  z = x;
   else       z = y;
   printf("max=%d", z);
}
main( )
{   int  x, y;
   scanf("%d%d", &x, &y);
   fun(x, y);
}
```

函数可以无返回值

函数语句

7.4.2 函数调用的三种方式

方法二：“函数表达式” 调用形式

```
int fun(int x, int y)
{  int  z;
   if(x > y)  z =
   else       z = y;
   return (z);
}
main( )
{   int  x, y, m;
   scanf("%d%d", &x, &y);
   m = fun(x, y);
   printf("max=%d", m);
}
```

函数必须有返回值

函数表达式

7.4.2 函数调用的三种方式

方法三：“函数实参” 调用形式

```
int fun(int x, int y)
{  int  z;
   if(x > y)  z =
   else       z = y;
   return (z);
}
main( )
{   int  x, y;
   scanf("%d%d", &x, &y);
   printf("max=%d", fun(x, y));
}
```

函数必须有返回值

函数实参

7.4.3 实参与形参的关系

关于函数实参、形参、返回值的说明:

- 实参是“实际参数”，可以是常量、变量、表达式、函数等。无论*实参*是何种类型的量，在进行函数调用时，它们都*必须具有确定的值*，以便把这些值传送给形参。因此，应预先用初始化、赋值语句、键盘输入等办法，使实参获得确定的值。
- 形参是“形式参数”，只有在发生函数调用时，才分配内存单元；调用结束时，即刻释放所分配的内存单元。因此，一旦*函数调用结束后，就不能再使用形参变量*。

7.4.3 实参与形参的关系

续上

- 实参向形参的数据传递是单向的，即只能是实参把数值传递给形参，形参的值不能反过来传递给实参。
- 实参和形参占用不同的内存单元，即使同名也互不影响。
- 函数只能有一个返回值，函数返回值的类型必须与“return(表达式);” 中表达式值的类型一致，一旦执行了return语句，函数调用就到此结束。
- 函数也可以没有返回值，此时程序中可以没有return语句，即使有也只能写成“return;”的形式。

7.4.3 实参与形参的关系

（1）以下函数返回值的类型是 （ A ）

```
fun ( float  x )
{
     float   y;
     y = 3 * x - 4;
     return  y;
}
```

A) int　　B) 不确定　　C) void　　D) float

函数定义时，如果缺省返回值类型，则系统默认返回值类型为 int 型。

7.4.3 实参与形参的关系

（2）若有以下函数首部

int fun(double x[10], int *n)

则以下函数声明语句中正确的是 （ D）

A) int fun(double x, int *n);

B) int fun(double, int);

C) int fun(double *x, int n);

D) int fun(double , int *);

考查函数声明与函数定义的对应关系。必须保证:

- 返回值类型一致;
- 函数名一致
- 参数表中的参数类型、个数、顺序一致。

重点 7.4.4 函数调用的传值方式 难点

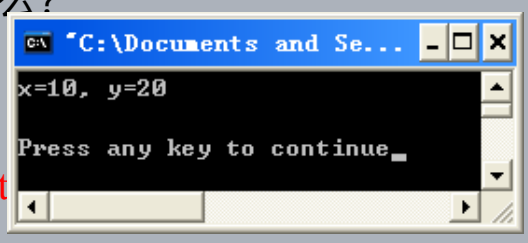
- 传值方式 —— 此时的实参是具体的数值，形参是相同类型的普通变量。
- 发生函数调用时，形参临时分配存储空间，实参将自己的数值复制一份给形参。函数调用结束后，形参的存储空间自动释放。
- 由于实参和形参占用的是各自的存储空间，因此函数调用过程中形参的改变只发生在被调函数内部，不会影响到主调函数中的实参。
- 切记：实参会向形参传递数值，但是形参不会反过来向实参传递数值。

7.4.4 函数调用的传值方式

【例4】请思考以下程序是否实现了将main函数中变量x、y值的交换？为什么？

```
void swap(int x, int y)
{
  int  t;
  t = x;      x = y;      y = t
}
main( )
{
  int  x = 10, y = 20;
  swap(x, y);
  printf("x=%d, y=%d\n", x, y);
}
```

传值方式

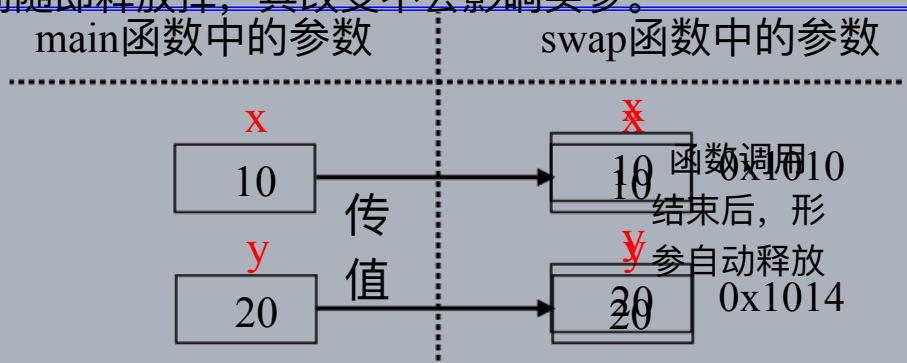


没有实现 x 和 y 的交换

《C程序设计基础》(Visual C++ 6.0环

7.4.4 函数调用的传值方式

【分析】实参x、y与形参x、y虽然同名，它们占用的确是各自的存储空间。前者只在main函数内部有效，后者只在swap函数内部有效。发生函数调用时，实参将数值复制一份给形参，但是函数调用结束后，形参的存储空间随即释放掉，其改变不会影响实参。



- 传地址方式——此时的实参为变量的地址、数组名、指针变量等。
- 形参是同类型的指针变量。
- 发生函数调用时，实参将其地址值复制一份给形参指针变量。于是我们说形参指针变量指向了实参地址所对应的存储空间。
- 由于形参指针变量指向了实参的存储空间，因此在被调函数内部，可以借助于形参指针变量间接地修改主调函数中实参的内容。这是传地址方式与传值方式最大的不同。

7.4.5 函数调用的传地址方式

【例5】请思考以下程序是否实现了将main函数中变量x、y值的交换？为什么？

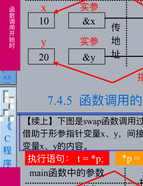
```
void swap(int *p, int *q)
{
  int t;
  t = *p;   *p = *q;   *q = t;
}
main()
{
  int x = 10, y = 20;
  swap(&x, &y);
  printf("x=%d, y=%d\n", x, y);
}
```

传地址方式

实现了 x 和 y 的交换

7.4.5 函数调用的传地址方式

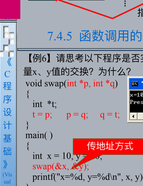
【分析】下图是swap函数调用开始时的示意图，图中显示实参是两个地址值（&x、&y），形参是同类型的指针变量，它接收实参传递过来的地址值，于是我们说，形参指针变量x、y指向了主函数中的普通变量x、y。



7.4.5 函数调用的传地址方式

【续上】下图是swap函数调用过程中的示意图，图中显示借助于形参指针变量x、y，间接地修改了main函数中普通变量x、y的内容。

执行语句： t = *p;　*p = *q;　*q = t;



7.4.5 函数调用的传地址方式

【例6】请思考以下程序是否实现了将main函数中变量x、y值的交换？为什么？

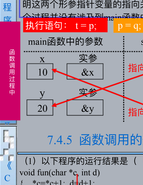
```
void swap(int *p, int *q)
{
  int *t;
  t = p;   p = q;   q = t;
}
main()
{
  int x = 10, y = 20;
  swap(&x, &y);
  printf("x=%d, y=%d\n", x, y);
}
```

传地址方式

没有实现 x 和 y 的交换

7.4.5 函数调用的传地址方式

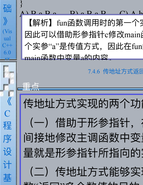
【分析】下图是swap函数调用开始时的示意图，图中显示实参是两个地址值（&x、&y），形参是同类型的指针变量，它接收实参传递过来的地址值。



7.4.5 函数调用的传地址方式

【续上】下图是swap函数调用过程中的示意图，图中显示swap函数实现的是形参指针变量p、q交换地址值，说明这两个形参指针变量的指向关系发生了交换，但是实参变量x、y的内容并没有发生变化。

执行语句： t = p;　p = q;　q = t;



7.4.5 函数调用的传地址方式

（1）以下程序的运行结果是（ D ）

```
void fun(char *c, int d)
{  *c=*c+1; d=d+1;
   printf("%c,%d,", *c, d); /*第一条打印语句*/
}
main()
{  char a='A', b='a';
   fun(&b, a);  printf("%c,%c\n", a, b); /*第二条打印语句*/
}
```

A. b,B,b,B　B. b,B,b,b　C. a,B,a,B　D. b,B,A,b

【解析】fun函数调用时的第一个实参"&b"是传地址方式，因此可以借助形参指针c修改main函数中变量b的内容；第二个实参"a"是传值方式，因此在fun函数内部的操作不会影响main函数中变量a的内容。

重点　难点

传地址方式实现的两个功能：

（一）借助于形参指针，在被调函数内部可以间接地修改主调函数中变量的值，被修改的变量就是形参指针所指向的变量。

（二）传地址方式能够实现被调函数向主调函数"返回"多个数值的目的。这与之前学习的return语句不同，return语句仅能向主调函数返回一个数值，当有多个数值需要返回主调函数时，必须使用参数的传地址方式。

【例7】输入一元二次方程的三个系数a、b、c，求方程的实根。定义三个函数 fun1()、fun2()、fun3()分别处理方程有实根、有两个相同实根、有两个不同实根的情况。要求：数据的输入、打印实根值都在main函数中完成。

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

【分析】

- fun1()处理无实根的情况，仅打印提示信息，因此设计为无参数、无返回值的形式。
- fun2()处理有两个相同实根的情况，并向main函数返回一个实根，因此设计为"有参数"、"有返回值"的形式。
- fun3()处理有两个不同实根的情况，只能通过传地址方式实现，因此设计为"有参数"、"无返回值"的形式。


```
#include <stdio.h>
#include <math.h>
void fun1();   //函数声明
double fun2(int, int);
void fun1(int, int, int, double *, double *);
void main(void)
{
  int a, b, c, t;
  double x1, x2;
  printf("请输入一元二次方程的系数:");
  scanf("a=%d, b=%d, c=%d", &a, &b, &c);
  t = b * b - 4 * a * c;
  if(t < 0)
    fun1();   //fun1函数调用
  else if(t == 0)
  {
    x1 = fun2(a, b);   //fun2函数调用
    printf("x1 = x2 = %f\n", x1);
  }
  else
  {
    fun3(a, b, t, &x1, &x2);   //fun3函数调用
    printf("x1=%f\t, x2=%f\n", x1, x2);
  }
}
```

续上

```
//fun1函数处理方程无实根的情况
void fun1(void)
{
  printf("方程无实根\n");
}

//fun2函数处理有两个相同实根的情况，将求得的实根值返回给main函数
double fun2(int a, int b)
{
  double x;
  x = -b / (2 * a);
  return (x);
}
```

续上

```
//fun3函数处理有两个不同实根的情况，无法使用return语句返回两个实根值，只能使用
//参数的传地址方式实现
void fun3(int a, int b, int t, double *x1,
          double *x2)
{
  *x1 = (-b + sqrt(t)) / (2 * a);
  *x2 = (-b - sqrt(t)) / (2 * a);
}
```

参数传地址方式

重点

7.5 数组与函数实参

有以下三种情况：

- 情况一：数组元素作实参，实现的是值传递。
- 情况二：数组名作实参，实现的是传地址方式。
- 情况三：数组元素的地址作实参，实现的也是传地址方式。

7.5.1 一维数组与函数实参

- 数组元素、数组名都可以作为函数实参。
- 数组元素作实参时，通常数组元素是一般类型的数据，因此实现传值方式，在被调函数内部的任何操作都不会影响主调函数中数组元素的初始值。
- 数组名作实参时，由于数组名是地址常量，因此实现的是传地址方式，此时的形参是同类型的指针变量。借助于形参指针变量可以在被调函数内部间接地修改主调函数中数组元素的初始值。

7.5.1 一维数组与函数实参

情况一： 数组元素作实参——传值方式

- 数组元素就是下标变量，它与普通变量并无区别。当数组元素作函数实参时，其处理与普通变量完全相同，实现的是传值方式。
- 在发生函数调用时，把数组元素的值传送给形参，实现数值的单向传递。该情况下，对应的形参是普通类型的变量，与数组同类型。
- 由于数组元素作实参是传值方式，因此在子函数内对形参的任何修改都不会反过来影响主函数中数组元素的初始值。

7.5.1 一维数组与函数实参

以下程序运行后的输出结果是_____3, 5_____。

```
void swap(int c0, int c1)
{    int  t;
     t=c0;    c0=c1;     c1=t;
}
main( )
{     int  b[2]={3, 5};
      swap(b[0], b[1]);
      printf("%d, %d\n", b[0], b[1]);
}
```

7.5.1 一维数组与函数实参

情况二： 数组名作实参——传地址方式

- 数组名是一个地址常量。当数组名作函数实参时，实现的是传地址方式。
- 该情况下，对应的形参是一个基类型与数组类型一致的指针变量，该指针变量指向数组的首地址。
- 在子函数内部，借助于形参指针变量不但可以访问数组元素，更重要的是可以修改数组元素的数值。

《C程序设计基础》(Visual C++ 6.0环

7.5.1 一维数组与函数实参

练习题

```
void reverse(int a[], int n)
{    int  i, t;
     for(i=0; i<n/2; i++)
     {    t=a[i];   a[i]=a[n-1-i];   a[n-1-i]=t;   }
}
 main( )
{    int b[10]={1,2,3,4,5,6,7,8,9,10};  int  i, s=0;
     reverse(b, 8);
     for(i=6; i<10; i++)   s+=b[i];
    printf("%d\n", s);
}
```

形参a是指针变量, 它指向原始数组b的首地址

数组名b作实参

《C程序设计基础》(Visual C++ 6.0环

7.5.1 一维数组与函数实参

程序运行后的输出结果是（ A）。

A) 22　　B) 10　　C) 34　　D) 30

【分析】数组名作实参时，实现的是传地址方式。形参指针变量指向原数组的首地址，在子函数中借助于形参指针变量可以对原数组元素进行引用，并能对数组元素的值作修改。reverse函数中的循环过程如下：

(1) i=0交换a[0]和a[7], 交换后的数组为: 8,2,3,4,5,6,7,1,9,10
(2) i=1交换a[1]和a[6], 交换后的数组为: 8,7,3,4,5,6,2,1,9,10
(3) i=2交换a[2]和a[5], 交换后的数组为: 8,7,6,4,5,3,2,1,9,10
(4) i=3交换a[3]和a[4], 交换后的数组为: 8,7,6,5,4,3,2,1,9,10

《C程序设计基础》(Visual C++ 6.0环

7.5.1 一维数组与函数实参

情况三： 数组元素的地址作实参——传地址方式

- 数组元素的地址作函数实参时，实现的是传地址方式。
- 该情况下，对应的形参是一个基类型与数组类型一致的指针变量，该形参指针变量指向对应的数组元素。此时对数组中其它元素的访问都是以该地址为基准，即以某个元素的地址（通常不是数组的首地址）为基础向前或者向后移动。
- 在子函数内部，借助于形参指针变量可以访问数组元素，并可以修改数组元素的值。

7.5.1 一维数组与函数实参

以下程序的运行结果是（ A）

```
void sum(int *a)
{    a[0] = a[1];     }
main( )
{   int aa[10]={1,2,3,4,5,6,7,8,9,10}, i;
    for(i=2;i>=0;i--)
          sum(&aa[i]);
    printf("%d\n", aa[0]);
}
```

A) 4　　B) 3　　C) 2　　D) 1

7.5.1 一维数组与函数实参

【分析】数组元素的地址作实参时，实现的是传地址方式，形参指针变量指向的是该数组元素。在子函数中借助于形参指针变量能够对数组元素进行引用，并能对数组元素的值作修改。sum函数中的for循环过程如下：

(1) i=2 &aa[2]作实参 □ 形参指针变量a指向元素aa[2] □ 语句 a[0]=a[1]; 相当于aa[2]=aa[3]=4

(2) i=1 &aa[1]作实参 □ 形参指针变量a指向元素aa[1] □ 语句 a[0]=a[1]; 相当于aa[1]=aa[2]=4

(3) i=0 &aa[0]作实参 □ 形参指针变量a指向元素aa[0] □ 语句 a[0]=a[1]; 相当于aa[0]=aa[1]=4

7.5.1 一维数组与函数实参

一维数组名作实参，对应的形参有三种等价形式：
例如：
```
main()
{
   char  s[10];
   ……
   fun(s);
   ……
}
```
形式一：fun(char *p) { … }
形式二：fun(char p[10]) { … }
形式三：fun(char p[ ]) { … }

说明：以上三种形式的形参，都代表一个指针变量，指向原数组的首地址。

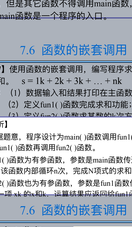
7.5.2 二维数组与函数实参

二维数组名作实参，对应的形参有三种等价形式：
例如：
```
main()
{
   char  s[4][10];
   ……
   fun( s );
   ……
}
```
形式一：fun(char *p[10]){…}
形式二：fun(char p[4][10]){…}
形式三：fun(char p[ ][10]){…}

说明：以上三种形式的形参，都代表一个行指针，其列数不能省略。

7.6 函数的嵌套调用

调用中又调用了其它函数。



7.6 函数的嵌套调用

说明：

- C语言允许函数的嵌套调用，但是不允许函数的嵌套定义，即各函数的定义是分开独立的，它们之间是平行的。
- C语言允许函数之间的嵌套调用，只有main函数例外，main函数可以调用其它任何函数，但是其它函数不得调用main函数，因为main函数是一个程序的入口。

7.6 函数的嵌套调用

【例7】使用函数的嵌套调用，编写程序求以下N项式的和。　s = 1k + 2k + 3k + … + nk
要求：（1）数据输入和结果打印在主函数中完成。
（2）定义fun1( )函数完成求和功能；
（3）定义fun2( )函数来计算幂的次方。

【分析】
- 根据题意，程序设计为main( )函数调用fun1( )函数，fun1( )函数再调用fun2( )函数。
- fun1( )函数为有参函数，参数是main函数传递过来的n和k，该函数内部循环n次，完成N项式的求和。
- fun2( )函数也为有参函数，参数是fun1函数传递过来的某一项 x 的 k 次幂。　运算结果应该回给fun1函数。

7.6 函数的嵌套调用

```
#include <stdio.h>
int fun1(int, int);  //函数声明
int fun2(int, int);
main( )
{
   int n, k, s;
   printf("请输入整数对n和k: ");
   scanf("%d%d", &n, &k);
   s = fun1(n, k); //调用fun1函数，获得N项式的和
   printf("s = %d\n", s);
}

int fun1(int n,  int k)//定义fun1函数, 有参数, 有返回值
{

   int i, x, sum=0;
   for(i=1; i<=n; i++)
   {
      x = fun2(i, k); //调用fun2求每一个当前项
      sum += x;       //将每一个当前项进行累加
   }
   return (sum); //返回N项式的总和
}

int  fun2(int x, int k)//fun2函数的定义, 有参数, 有返回值
{
   int i, j=1;
   for(i=1; i<=k; i++) //求每一个当前项，即xk
      j *= x;
   return (j); //返回当前项的值
}
```

上机实验（一）解答

1 写一个判断素数的函数，在主函数中调用该函数，统计100以内的正整数中哪些是素数，并输出结果。

方法一

```
#include<stdio.h>
#include<math.h>
int fun(int x);
main()
{
   int i,a;
   for(i = 1; i < 100; i++)
   {
      a = fun(i);
      if(a != 0)
         printf("%d ",i);
   }
   printf("\n");
}
```

```
int fun(int x)
{
   int j,y,m=1;
   for(j=2;j<x;j++)
   {
      y = x % j;
      //   m=m*y;
      if (y == 0)   //如果有一次为0，就不需再判断，break退出
      {
         return 0;
         break;
      }
   }

   return 1;      }
```

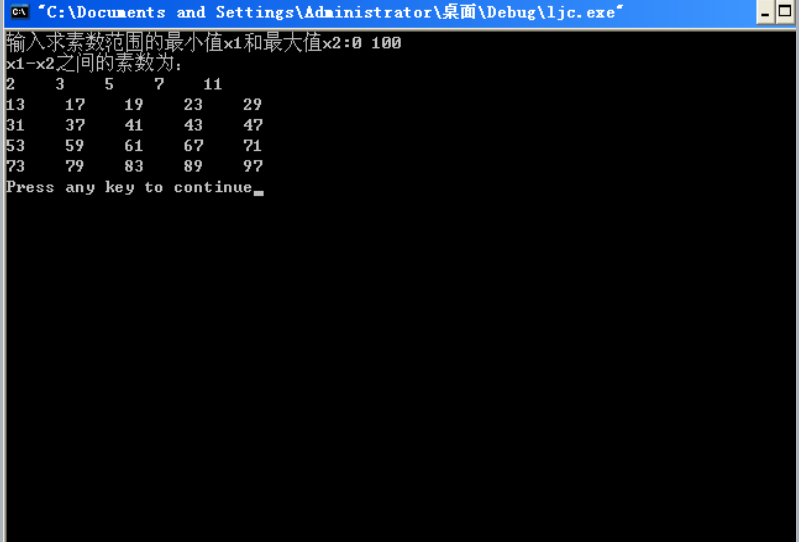
方法二

```
#include <stdio.h>
int fun(int n)
{
   int i,a;
   a= 1;
   for(i=2;i<=n/2;i++)
   {
   if(n%i==0)
   {
      a=0;
      break;
   }
   }

   if(a==1)
   {
      printf("%d ",n);
      return 1;
   }
   else
   return 0;
}
int main()
{
   int i,j=0,k;
   for(i=2;i<=100;i++)
   {
   if(fun(i))
```

```
   {
      j++;
      if(j%5==0)
      printf("\n");
   }
   }
}
```

方法三

```
#include<stdio.h>
void fun(int n,int m)
{
   int x,i,j;
   printf("x1-x2之间的素数为：\n");
   for(x=n;x<m;x++)
   {
      for(i=2;i<x;i++)
      {
         if(x%i==0)
         break;
      }

if(i==x)
      {
         printf("%d    ",x);
         j++;
         if(j%5==0)
         printf("\n");
      }
   }
}
main( )
{   int x1,x2;
   printf("输入求素数范围的最小值x1和最大值x2：");
   scanf("%d%d",&x1,&x2);
   fun(x1,x2);
}
```


"C:\Documents and Settings\Administrator\桌面\Debug\ljc.exe"
输入求素数范围的最小值x1和最大值x2:0 100
x1-x2之间的素数为:
2 3 5 7 11
13 17 19 23 29
31 37 41 43 47
53 59 61 67 71
73 79 83 89 97
Press any key to continue

```c
#include<stdio.h>
void fun(int x)
{
   int i;
      for(i=2;i<x;i++)
      {
         if(x%i==0)
         break;
      }
      if(i==x)
      {
         printf("%d\t ",x);
      }}
```

```c
main( )
{   int x1,x2,n,j;
   printf("输入求素数范围的最小值x1和最大值x2:");
   scanf("%d%d",&x1,&x2);
   printf("x1-x2之间的素数为：\n");
   for(n=x1;n<x2;n++)
   {
   fun(n);
   }
}
```

```
"C:\Documents and Settings\Administrator\桌面\Debug\ljc.exe"
输入求素数范围的最小值x1和最大值x2:0 100
x1-x2之间的素数为:
2	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	Press any key to continue
```

2 编写一个函数输出以下图形，图形的行数以参数的形式给出。

```
   *
  ***
 *****
*******
```

```c
#include<stdio.h>
void fun(int n)
{
   int i,j,k;
   for(i=1;i<=n;i++)
   {
      for(j=0;j<=n-i;j++)
         printf(" ");
      for(k=1;k<=2*i-1;k++)
         printf("*");
      printf("\n");
   }
}
```

```c
main()
{
   int n;
   printf("输入需要打印图形的行数：");
   scanf("%d",&n);
   fun(n);
}
```

```
"C:\Documents and Settings\Admin...
输入需要打印图形的行数：6
      *
     ***
    *****
   *******
  *********
 ***********
Press any key to continue
```

上机实验二解答

1 定义一个子函数fun()，其功能是求1到100的和；

方法一

```c
#include<stdio.h>
void fun(int n,int m)
{
   int  i,sum=0;
   i=n;
   while(i <= m)
   {
   sum +=  i;
   i ++;
   }
   printf("sum=%d\n", sum);
}
```

```c
void main()
{
   int x1,x2;
   printf("请输入计算求和的范围x1--x2：");
   scanf("%d%d",&x1,&x2);
   fun(x1,x2);
}
```

方法二

```c
#include<stdio.h>
int fun(int n,int m)
{
   int  i,sum=0;
   i=n;
   while(i <= m)
   {
   sum +=  i;
   i ++;
   }
   return(sum);

}
```

```c
void main()
{
   int x1,x2,sum=0;
   printf("请输入计算求和的范围x1--x2： ");
   scanf("%d%d",&x1,&x2);
   sum=fun(x1,x2);
   printf("sum=%d\n", sum);
}
```

可以用for循环，do-while循环实现本题

2 在主函数中定义一个大小为10的整型数组，并对数组元素进行初始化，定义一个fun（）函数，其功能是求数组中10个数的总和、最大值，最小值。

要求：

（1）总和、最大值、最小值都在主函数打印。

（2）**总和由 return 语句返回；最大值、最小值由参数返回。**

提示：

（1）主函数中的变量定义可参考如下

int a[10]={3,8,1,5,2,9,6,10,7,4}, sum, max, min, i;

（2）fun( )函数有参数、有返回值。函数首部可参考如下

int fun(int *p, int *pmax, int *pmin)

其中指针 p 指向数组 a，指针 pmax 和 pmin 分别指向变量 max 和 min。

```c
#include <stdio.h>
int fun(int *p, int *pmax, int *pmin)
{
   int i,sum=0;
   for( i=0;i<10;i++)
   {
      sum+=p[i];
      if(p[i]>*pmax)
         *pmax=p[i];
      if(p[i]<*pmin)
      *pmin=p[i];
   }return sum;
   }
```

```c
main()
{
   int a[10]={3,8,1,5,2,9,6,10,7,4}, sum, max, min, i;
   printf("原数组中的元素:\n");
   for(i=0;i<10;i++)
      printf("%d\t",a[i]);
      printf("\n");
   max=a[0];min=a[0];
   sum=fun(a,&max,&min);

 printf("sum=%d,max=%d,min=%d\n",sum,max,min);
}
```

3 用函数，编程打印100-700以内不能同时被5和17整除的数。

方法一

```c
#include <stdio.h>
void fun(int x,int y)
 {
   int i;
      for(i=x;i<=y;i++)
   {
       if(i%5==0 && i%17==0)
            continue;
       printf("%d\t", i );
   }

      printf("\n");
 }
```

```c
main()
{
   int n,m;
   printf("请输入一个范围:");
   scanf("%d%d",&n,&m);
   fun(n,m);
}
```

方法二（主函数跟方法一一样）

```c
void fun(int x,int y)
 {
   int i;
      for(i=x;i<=y;i++)
   {
         if(i%5||i%17)
            printf("%d\t",i);
      }

      printf("\n");
 }
```

实验解答

定义fun()函数，用数组名作函数实参，其功能是将**主函数中原始数组中每个元素的值加10**，并在main（）函数中调用它。

要求：

（1）数组中每个元素加10后的结果在主函数打印。

提示：

（1）主函数中的变量定义可参考如下

int a[10]={1,2,3,4,5,6,7,8,9,12}, i;

（2）fun( )函数有参数。函数首部可参考如下

void fun(int *p)，其中指针 p 指向数组 a

```c
#include<stdio.h>
void fun(int *);
main()
{
   int a[10]={1,2,3,4,5,6,7,8,9,12}, i;
   printf("(1)打印子函数调用前的数组元素是\n");
   for(i=0;i<10;i++)
   {
      printf("%d\t",a[i]);
   }
```

```c
fun(a);
printf("\n(3)打印子函数函数调用后的数组元素是\n");
for(i=0;i<10;i++)
   {
      printf("%d\t",a[i]);
   }
   printf("\n");
}
```

子函数方法一

```c
void fun(int *p)
{
   int i;
   printf("\n(2)打印子函数调用过程中的数组元素是\n");
   for(i=0;i<10;i++)
   {
      *(p+i)+=10;
      printf("%d\t",*(p+i));

   }
}
```

子函数方法二

```c
void fun(int *p)
 {
   int i;
   printf("\n(2)打印子函数调用过程中的数组元素是\n");
   for(i=0;i<10;i++)
   {
      *p+=10;
      printf("%d\t",*p);
      p++;
   }
 }
```

子函数方法三

```c
void fun(int *p)
{
   int i;
   printf("\n(2)打印子函数调用过程中的数组元素是\n");
   for(i=0;i<10;i++)
   {
      p[i]+=10;
      printf("%d\t",p[i]);
   }
}
```

实验解答

在主函数中定义一个大小为10的整型数组，并对数组元素进行初始化，定义一个fun（）函数，其功能是求数组中10个数的总和、最大值，最小值。

要求：

（1）总和、最大值、最小值都在主函数打印。

（2）**总和由 return 语句返回；最大值、最小值由参数返回。**

提示：

（1）主函数中的变量定义可参考如下

int a[10]={3,8,1,5,2,9,6,10,7,4}, sum, max, min, i;

（2）fun( )函数有参数、有返回值。函数首部可参考如下

int fun(int *p, int *pmax, int *pmin)

其中指针 p 指向数组 a，指针 pmax 和 pmin 分别指向变量 max 和 min。

方法一

```c
#include <stdio.h>
int fun(int *p, int *pmax, int *pmin)
{
   int i,sum=0;
   for( i=0;i<10;i++)
   {
      sum+=p[i];
      if(p[i]>*pmax)
         *pmax=p[i];
      if(p[i]<*pmin)
      *pmin=p[i];
   }return sum;
   }
```

```c
main()
{
   int a[10]={3,8,1,5,2,9,6,10,7,4}, sum, max, min, i;
   printf("原数组中的元素:\n");
   for(i=0;i<10;i++)
      printf("%d\t",a[i]);
      printf("\n");
   max=a[0];min=a[0];
   sum=fun(a,&max,&min);

 printf("sum=%d,max=%d,min=%d\n",sum,max,min);
}
```

方法二（子函数）

```c
#include <stdio.h>
int fun(int *p, int *pmax, int *pmin)
{
   int i,sum=0;
   for( i=0;i<10;i++)
   {
      sum+=*(p+i);
      if(*(p+i)>*pmax)
         *pmax=*(p+i);
      if(p[i]<*pmin)
      *pmin=*(p+i);
   }return sum;
```