

第8章 善于利用指针

8.1 指针是什么

8.2 指针变量

8.3 通过指针引用数组

8.4 通过指针引用字符串

8.5 指向函数的指针

8.6 返回指针值的函数

8.7 指针数组和多重指针

8.8 动态内存分配与指向它的指针变量

8.9 有关指针的小结

8.3通过指针引用数组

8.3.1 数组元素的指针

8.3.2 在引用数组元素时指针的运算

8.3.3 通过指针引用数组元素

8.3.4 用数组名作函数参数

8.3.5 通过指针引用多维数组

8.3.1 数组元素的指针

- 一个变量有地址，一个数组包含若干元素，每个数组元素都有相应的地址
- 指针变量可以指向数组元素（把某一元素的地址放到一个指针变量中）
- 所谓数组元素的指针就是数组元素的地址



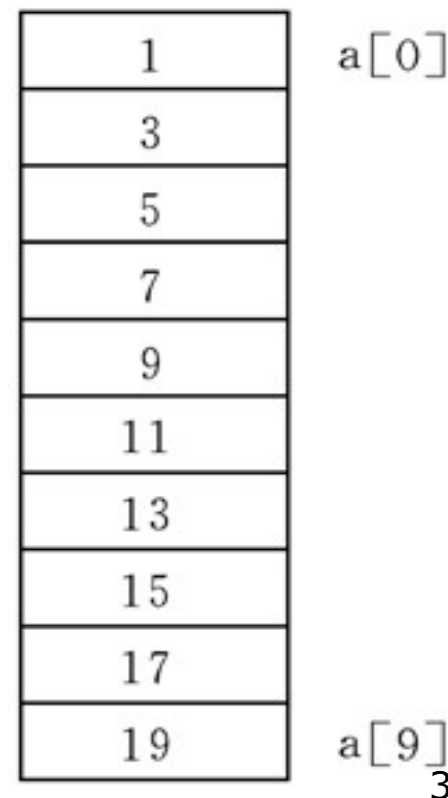
- 可以用一个指针变量指向一个数组元素

```
int a[10]={1,3,5,7,9,11,13,15,17,19};
```

```
int *p;
```

```
p=&a[0];
```

注意：数组名**a**不代表整个数组，只代表数组首元素的地址。“**p=a;**”的作用是“把**a**数组的首元素的地址赋给指针变量**p**”，而不是“把数组**a**各元素的值赋给**p**”。



8.3.2 在引用数组元素时指针的运算

- 在指针指向数组元素时，允许以下运算：
 - ◆ 加一个整数(用+或+=)，如**p+1**
 - ◆ 减一个整数(用-或-=)，如**p-1**
 - ◆ 自加运算，如**p++**，**++p**
 - ◆ 自减运算，如**p--**，**--p**
 - ◆ 两个指针相减，如**p1-p2** (只有**p1**和**p2**都指向同一数组中的元素时才有意义)
- **(1)** 如果指针变量**p**已指向数组中的一个元素，则**p+1**指向同一数组中的下一个元素，**p-1**指向同一数组中的上一个元素。

float a[10], *p=a; 假设**a[0]**的地址为**2000**，则

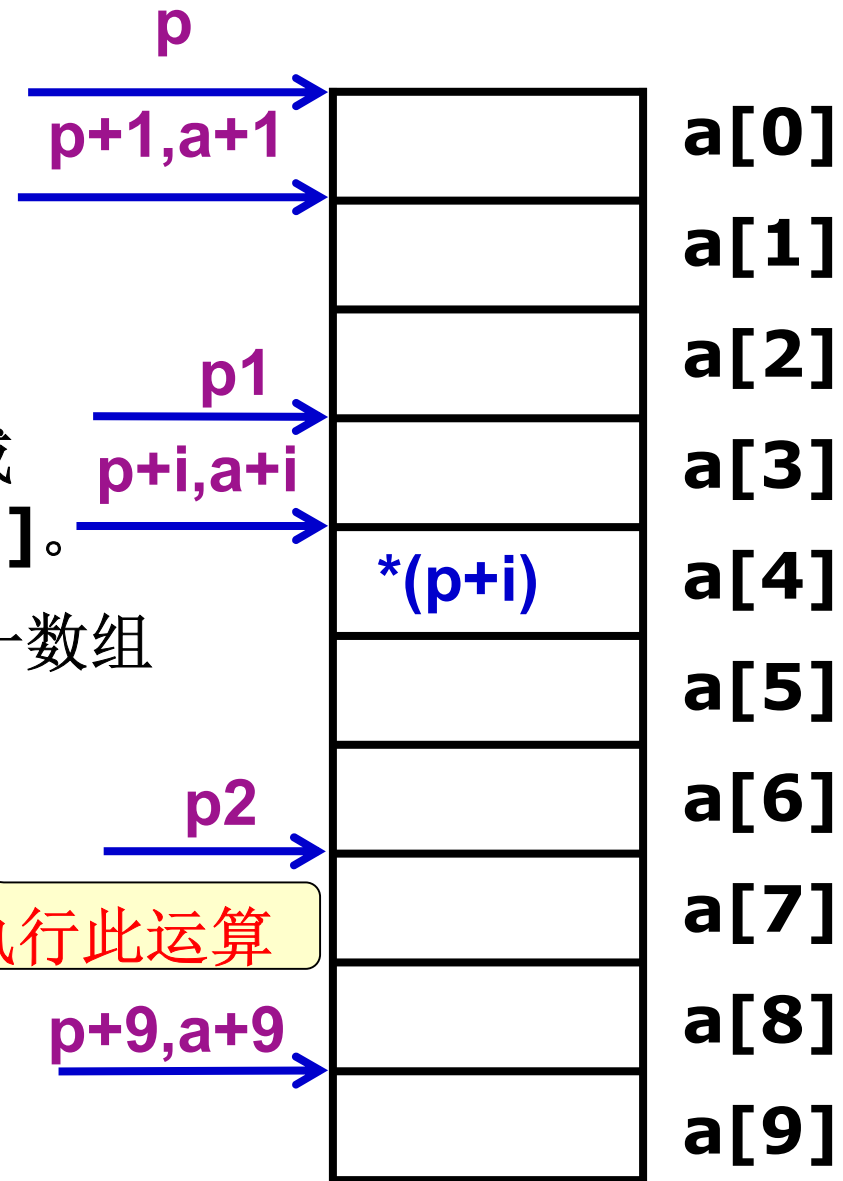
- ◆ **p**的值为**2000**
- ◆ **p+1**的值为**2004**
- ◆ **p-1**的值为**1996**

越界

(2) 如果 p 的初值为 $\&a[0]$,
则 $p+i$ 和 $a+i$ 就是数组元素
 $a[i]$ 的地址, 或者说, 它们指
向 a 数组下标为 i 的元素

(3) $*(p+i)$ 或 $*(a+i)$ 是 $p+i$ 或
 $a+i$ 所指向的数组元素, 即 $a[i]$ 。

(4) 如果指针 $p1$ 和 $p2$ 都指向同一数组
 $p2-p1$ 的值是 4
那么 $p1+p2$ 的值呢?



不能执行此运算

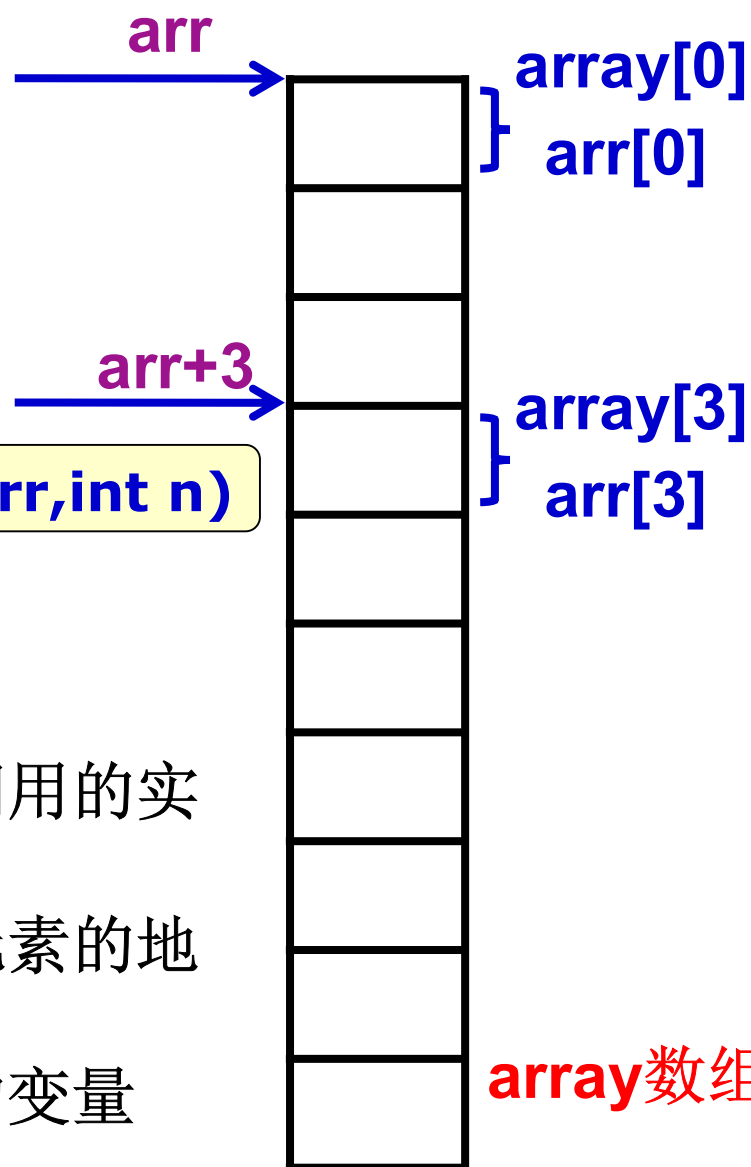
8.3.3 通过指针引用数组元素

- 引用一个数组元素，可用下面两种方法：
 - ◆(1) 下标法，如 $\mathbf{a[i]}$ 形式
 - ◆(2) 指针法，如 $\mathbf{*(a+i)}$ 或 $\mathbf{*(p+i)}$
 - 其中 \mathbf{a} 是数组名， \mathbf{p} 是指向数组元素的指针变量，其初值 $\mathbf{p=a}$
- 例8.6 有一个整型数组 \mathbf{a} ，有10个元素，要求输出数组中的全部元素。
- 解题思路：引用数组中各元素的值有3种方法
 - ◆(1) 下标法；
 - ◆(2) 通过数组名计算数组元素地址，找出元素的值；
 - ◆(3) 用指针变量指向数组元素
- 分别写出程序，并比较分析。

8.3.4 用数组名作函数参数

```
int main()  
{ void fun(int arr[],int n);  
  int array[10];  
  .....  
  fun (array,10);  
  return 0;  
}  
void fun(int arr[ ],int n)  
{  
  |  
}
```

void fun(int *arr,int n)




- 用数组名作函数参数时，函数调用的实参是数组名
- 因为实参数组名代表该数组首元素的地址，实际上传递的地址值
- 故，形参可以是一个指针类型的变量

- **C**语言编译器都是将形参数组名作为指针变量来处理的
- 实参数组名是指针常量，但形参数组名也是按指针变量来处理
- 在函数调用进行虚实结合后，它的值就是实参数组首元素的地址
- 在函数执行期间，形参数组就被编译器处理成指针变量，可以被再次赋值

```
void fun (int arr[ ], int n)  
{ printf("%d\n", *arr);  
  arr=arr+3;  
  printf("%d\n", *arr);  
}
```


例8.7 将数组a中n个整数逆置存放

➤ 解题思路：以n为10为例，将a[0]与a[9]对换，.....将a[4]与a[5]对换。



```
#include <stdio.h>
void inv(int p[], int n);
int main()
{
    int a[10]={3,7,9,11,0,6,7,5,4,2}, i;
    inv(a, 10);
    for(int i=0; i<10; i++)
        printf("%d ", a[i]);
    printf("\n");
    return 0;
}
```

例**8.8** 以指向数组的指针为函数参数，使用插入排序方法对长度为**n**的整型数组排序，主函数调用此函数（以长度为**10**的数组为例），并将排序结果输出。

➤ 解题思路：

```
void insertion_sort(int *p, int n)  
{  
    int i, j;  
    for(i=1; i<n; i++)  
    {  
        int key = *(p+i);  
        j=i-1;  
        while(j>=0 && *(p+i)>key){  
            *(p+j+1) = *(p+j);  
            j--;  
        }  
        *(p+j+1) = key;  
    }  
}
```

8.3.5 通过指针引用多维数组

指针变量可以指向一维数组中的元素，也可以指向多维数组中的元素。但在概念上和使用方法上，多维数组的指针比一维数组的指针要复杂一些。

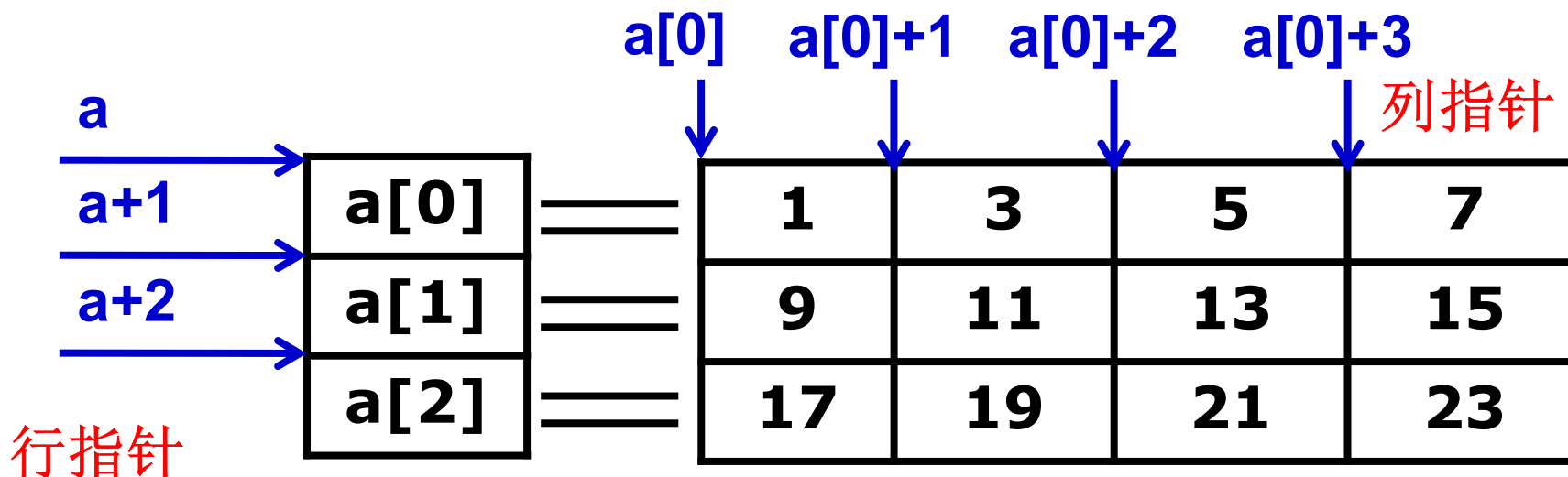
$a[0]$ 代表 $a[0][0]$ 的地址

$a[0]+1$ 代表 $a[0][1]$ 的地址

$a[0]+2$ 代表 $a[0][2]$ 的地址

$a[0]+3$ 代表 $a[0][3]$ 的地址

列指针每加1，走一列



例8.9 二维数组的有关数据(地址和值)

```
int a[3][4]={1,3,5,7,9,11,13,15,  
            17,19,21,23};
```

了解:

a, a+1, a[1], a[1]+1, a[1][2]之间的关系
怎样通过数组名访问数组中的各个元素

(1)下标法 a[i][j]

(2)指针法 *(*(a+i)+j)

2. 指向多维数组元素的指针变量

(1) 二维数组的存储

逻辑上，二维数组类似于**EXCEL**表，有行和列构成，行数和列数有确定的数值，表中的每个单元格相当于二维数组的一个元素，表格中所有元素具有相同的数据类型。

物理上，二维数组的元素在内存中是连续存储的（类似于一维数组），以行序为主序存储，即存放完序号为**0**的行中的全部元素后，接着存放序号为**1**的行中的全部元素，依此类推。各元素的地址均可由首元素的地址得到。如：

T a[m][n]; // 存储类似于长度为**m*n**的一维数组
&a[i][j]的值为: **&a[0][0]+((i*n)+j);**

➤ 例**8.10** 有一个**3×4**的矩阵，要求用指向数组元素的指针输出该矩阵。

➤ 解题思路：

◆ 二维数组在存储上类似与一维数组，首元素的地址是可求的，则后续所有元素的地址可依次求得。

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

```
int main()
```

```
{
```

```
    int a[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

```
    int *p;
```

```
    for(p=&a[0][0]; p<&a[0][0]+3*4; p++){
```

```
        if((p-&a[0][0])%4 == 0) printf("\n");
```

```
        printf("%4d", *p);
```

```
    }
```

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

```
    return 0;
```

```
}
```

这里**&a[0][0]**可改为**a[0]**

(2) 指向二维数组的指针

物理上，二维数组的元素是连续存储的，并没有行列之分，但编译器将二维数组解释成一维数组的数组，且必须明确一维数组的长度。

如：**int a[3][4]**

a是一个**3×4**的整型数组，更确切地说，**a**是一个包含**3**个元素的数组(分别为：**a[0]**,**a[1]**,**a[2]**)，每个元素又都是包含**5**个整数的数组。

a[i]等价于***(a+i)**，**a**是此数组的首地址，**a**被转换为指向包含**5**个整型元素的一维数组。

在**a[i][j]**中，**a[i]**被转换为指向数组中第一个元素的指针。故，**a[i]**是**int***类型，**a[i][j]**是**int**类型。

int *p = a[0];

从一维数组来讲，**a**是长度为**5**的一维数组的数组名，**a[i]**等价于***(a+i)**，故**a+i**与**a[i]**类型不同。

int *p = a; //error!

定义二维数组的指针

T (*p)[n]

这里，**p**指向一行有**n**个元素的二维数组

上例**8.10**用二维数组的指针来实现，程序如下：

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

```
int main()
```

```
{
```

行指针

```
int a[3][4]={1,3,5,7,9,11,13,15,17,19,21,23};
```

```
int (*p)[4],i,j;
```

```
p=a;
```

```
printf("enter row and colum:");
```

```
scanf("%d,%d",&i,&j);
```

```
printf("a[%d,%d]=%d\n",i,j,*(* (p+i)+j));
```

```
return 0;
```

```
}
```

a[i][j]

```
enter row and colum:1,2
a[1,2]=13
```


3. 用指向数组的指针作函数参数

- 一维数组名可以作为函数参数，多维数组名也可作函数参数。
- 用指针变量作形参，以接受实参数组名传递来的地址。
- 可以有两种方法：
 - ①用指向变量的指针变量
 - ②用指向一维数组的指针变量

例**8.11** 某宿舍共**3**个学生，均学相同的**4**门课，计算总平均分数以及每个同学所有课程的最高分。

- 解题思路：这个题目是很简单的。本例用指向数组的指针作函数参数。用函数**average**求总平均成绩，用函数**getMax**找出第**i**个学生的最高成绩。

```
float average(float (*p)[4], int n)  
{  
    int i, j;  
    float sum = 0;  
    for(i=0; i<n; i++)  
        for(j=0; j<4; j++)  
            sum += p[i][j]; // sum += (*(p+i)+j);  
    return sum/(n*4);  
}
```

```
float getMax(float *p, int n)  
{  
    int i;  
    float max = p[0]; // float max = *p;  
    for(i=1; i<n; i++)  
        if(*(p+i) > max) max = *(p+i);  
    return max;  
}
```

```

#include <stdio.h>
float average(float (*p)[4], int n);
float getMax(float *p, int n);
int main()
{
    float score[3][4]={ {65,67,70,60},
                        {80,87,90,81},{90,99,100,98}};
    float avg = average(score, 3);
    printf("The avg score of all is:%.1f\n", avg);
    int i;
    for(i=0; i<3; i++){
        float max = getMax(score[i], 4);
        printf("The %dth max score:%.1f\n", i, max);
    }
    return 0;
}

```

```

The avg score of all is:82.3
The 0th max score:70.0
The 1th max score:90.0
The 2th max score:100.0

```

8.4 通过指针引用字符串

8.4.1 字符串的引用方式

8.4.2 字符指针作函数参数

8.4.3 使用字符指针变量和字符数组的比较

8.4.1 字符串的引用方式


- 字符串是存放在字符数组中的。引用一个字符串，可以用以下两种方法。

(1) 用字符数组存放一个字符串，使用**scanf**函数可以通过数组名和格式字符“**%s**”输出该字符串，也可以通过数组名和下标引用字符串中单个字符。

(2) 用字符指针变量指向一个字符串常量，通过字符指针变量引用字符串常量。

- 例**8.12** 定义一个字符数组，在其中存放字符串“**I love China!**”，输出该字符串和第**8**个字符。

```
#include <stdio.h>
int main()
{ char *s = "I love China!";
  s = "I am a student.";
  printf("%c\n",s[8]);
  return 0;
}
```



C语言中的字符串都是以地址表示的

例**8.13** 用指针变量来处理字符串复制问题。

- 解题思路：定义两个指针变量**p1**和**p2**，分别指向字符数组**a**和**b**。改变指针变量**p1**和**p2**的值，使它们顺序指向数组中的各元素，进行对应元素的复制。

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

```
int main()
```

```
{char a[]="An iPhone XR.", b[20], *p1, *p2;
```

```
  p1=a; p2=b;
```

```
  for( ; *p1!='\0'; p1++,p2++)
```

```
    *p2=*p1;
```

```
  *p2='\0'; //字符串结束标识，非常重要
```

```
  printf("string a is:%s\n", p1);
```

```
  printf("string a is:%s\n", p2);
```

//结果如何？

```
  return 0;
```

```
}
```

8.4.2 字符指针作函数参数

- 如果想把一个字符串从一个函数“传递”到另一个函数，可以用地址传递的办法，即用字符数组名作参数，也可以用字符指针变量作参数。
 - ◆ 在被调用的函数中可以改变字符串的内容
 - ◆ 在主调函数中可以引用改变后的字符串。
- 例**8.14** 用函数调用实现字符串的复制。
 - ◆ 用字符数组名做函数参数
 - **void str_copy(char from[], char to[]);**
 - ◆ 用字符指针做函数参数
 - **void str_copy(char *from, char *to);**
 - ◆ 函数调用的实参可以为数组名，亦可为字符指针
 - ◆ 函数的具体实现由多种算法

8.4.3 使用字符指针变量和字符数组的比较

- 用字符数组和字符指针变量都能实现字符串的存储和运算，但它们二者之间是有区别的，不应混为一谈，主要有以下几点。

(1) 字符数组由若干个元素组成，每个元素中放一个字符，而字符指针变量中存放的是地址（字符串首字符的地址），决不是将字符串放到字符指针变量中。

(2) 赋值方式。可以对字符指针变量赋值，但不能对数组名赋值。

char *a; a="I love China!"; 对

char str[14];str[0]='I'; 对

char str[14]; str="I love China!"; 错

(3) 初始化的含义

char *a="I love China! ";与

char *a; a="I love China! ";等价

但

char str[14]= "I love China! " ;与

char str[14]; str[]="I love China! " ;不等价

(4) 存储单元的内容

编译时为字符数组分配若干存储单元，以存放各元素的值，而对字符指针变量，只分配一个存储单元

char *a; scanf("%s",a); //错

char *a,str[10];

a=str;

scanf ("%s",a); //对

(5) 指针变量的值是可以改变的，而数组名代表一个固定的值(数组首元素的地址)，不能改变。

char *s = "I love China!";

s[7] = 'c'; //错

(6) 字符数组中各元素的值是可以改变的，但字符指针变量指向的字符串常量中的内容是不可以被取代的。

例8.15 改变指针变量的值。

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

```
int main()
```

```
{ char *a="I love China!";
```

```
  a=a+7;
```

```
  printf("%s\n",a);
```

```
  return 0;
```

```
}
```

不能改为

```
char a[]="I love China!";
```

China!

(7) 引用数组元数

对字符数组可以用下标法和地址法引用数组元素(a[5],*(a+5))。

(8) 用指针变量指向一个格式字符串，可以用它代替

scanf或printf函数中的格式字符串。

```
char *format="a=%d,b=%f\n";
```

```
printf(format,a,b);
```

相当于

```
printf("a=%d,b=%f\n",a,b);
```