第八章 善于利用指针

作者：石璞东

参考资料：《C程序设计（第四版）》谭浩强

前言

不掌握指针就是没有掌握C的精华。

8.1 指针是什么

1. 概念

如果在程序中定义了一个变量，在对程序进行编译时，系统就会给这个变量分配内存单元，编译系统根据程序中定义的变量类型，分配一定长度的空间，内存区的每一个字节有一个编号，这就是**地址**，它相当于旅馆中的房间号，在地址所标志的内存单元中存放的数据则相当于旅馆房间中居住的旅客。我们将**地址**形象化为**指针**，即通过它能找到以它为地址的内存单元。

对于如下语句：

```
printf("%d\n",i);
```

在程序中一般是通过变量名来引用变量的值，实际上是通过变量名i找到存储单元的地址，从而对存储单元进行存取操作的，程序经过编译以后已经将变量名转换为变量的地址，对变量值的存取都是通过地址进行的。

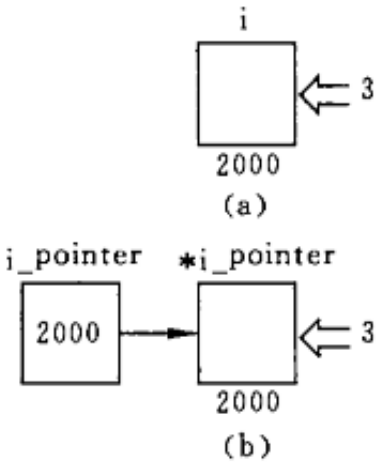
对于如下语句：

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

程序执行时，是将键盘输入的值送到地址为2000开始的整型存储单元中。

2. 直接访问与间接访问

打个比方，为了开一个抽屉A，有两种办法，一种是将A钥匙带在身上，需要时直接找出该钥匙打开抽屉，取出所需的东西；另一种办法是：为了安全起见，将该A钥匙放到另一抽屉B中锁起来，如果需要打开A抽屉，就需要找出B钥匙，打开B抽屉，取出A钥匙，再打开A抽屉，即可取出取出A抽屉中的东西，这就是间接访问。



如果需要将数值3送到变量中去，可以有两种方法：

- 将3直接送到变量`i`所标志的单元中，如`i=3`，如图`a`所示；
- 将3送到变量`i_pointer`所指向的单元，即变量`i`的存储单元，如`*i_pointer=3`，其中`*i_pointer`表示`i_pointer`指向的对象；

指针是一个地址，而指针变量是存放地址的变量。

8.2 指针变量

存放地址的变量是指针变量，它用来指向另一个对象，如变量、数组、函数等。

8.2.1 使用指针变量的例子

例8.1 通过指针变量访问整型变量

```c
#include<stdio.h>
int main(){
    int a = 100,b = 10;
    int *pointer_1,*pointer_2;
    pointer_1 = &a;
    pointer_2 = &b;
    printf("a = %d,b = %d\n",a,b);
    printf("*pointer_1 = %d,*pointer_2 = %d\n",*pointer_1,*pointer_2);
    printf("addr_a = %x,addr_b = %x\n",&a,&b);
    return 0;
}
```

请看演示效果：

```c

#include<stdio.h>
int main(){
    int a = 100,b = 10;
    int *pointer_1,*pointer_2;
    pointer_1 = &a;
    pointer_2 = &b;
    printf("a = %d,b = %d\n",a,b);
    printf("*pointer_1 = %d,*pointer_2 = %d\n",*pointer_1,*pointer_2);
    printf("addr_a = %x,addr_b = %x\n",&a,&b);
    return 0;
}

```

Format specifies type 'u

```
a = 100,b = 10
*pointer_1 = 100,*pointer_2 = 10
addr_a = efbff518,addr_b = efbff514
Program ended with exit code: 0
```

8.2.2 怎样定义指针变量

定义指针变量的一般形式为：`类型名 *指针变量名`，如`int *pointer_1,*pointer_2`，其中`int`表示在定义指针变量时必须指定的基类型，指针变量是基本数据类型派生出来的类型，它不能离开基本类型而独立存在。

一个变量的指针的含义包含两个方面，一方面是以存储单元编号表示的地址（如编号为2000的字节），另一方面是它指向的存储单元的数据类型，如`int`、`char`、`float`等，在说明变量类型时不能一般地说`a`是一个变量，而应完整地说`a`是指向整型数据的指针变量，`b`是指向单精度型数据的指针变量，`c`是指向字符型数据的指针变量。

8.2.3 怎样引用指针变量

在引用指针变量时，可能有3种情况：

- 给指针变量赋值，如`p=&a;`，即将`a`的地址赋给指针变量`p`，此时指针变量`p`的值是变量`a`的地址，`p`指向`a`；
- 引用指针变量指向的变量，如果已执行`p=&a;`，即指针变量`p`指向了整型变量`a`，则`printf("%d",*p);`的作用是以整数形式输出指针变量`p`所指向的变量的值，即变量`a`的值，如有以下赋值语句`*p=1;`表示将整数1赋给当前所指向的变量，如果`p`指向变量`a`，则相当于把1赋给`a`，即`a=1;`；
- 引用指针变量的值，如`printf("%o",p);`作用是以八进制数形式输出指针变量`p`的值，如果`p`指向了`a`，就是输出了`a`的地址，即`&a`；

例8.2 输入`a`和`b`两个整数，按先大后小的顺序输出`a`和`b`，要求不交换整型变量的值，而是交换两个指针变量的值。

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

```c
int main(){
    int *p1,*p2,*p,a,b;
    printf("请输入两个整数: ");
    scanf("%d %d",&a,&b);
    p1 = &a;
    p2 = &b;
    if(a<b){
        p  = p1;
        p1 = p2;
        p2 = p;
    }
    printf("a = %d,b = %d\n",a,b);
    printf("max = %d,min = %d\n",*p1,*p2);
    return 0;
}
```

请看演示效果：

```c
#include<stdio.h>
int main(){
    int *p1,*p2,*p,a,b;
    printf("请输入两个整数: ");
    scanf("%d %d",&a,&b);
    p1 = &a;
    p2 = &b;
    if(a<b){
        p  = p1;
        p1 = p2;
        p2 = p;
    }
    printf("a = %d,b = %d\n",a,b);
    printf("max = %d,min = %d\n",*p1,*p2);
    return 0;
}
```

```
请输入两个整数: 1 2
a = 1,b = 2
max = 2,min = 1
Program ended with exit code: 0
```

8.2.4 指针变量作为函数参数

函数的参数不仅仅可以是整型、浮点型、字符型等数据，还可以是指针类型，它的作用是将一个变量的地址传送到另一个函数中。

单向传递的值传递方式，形参值的改变不能使实参的值随之改变，因此为了使函数中改变了的变量值能被主调函数`main`所用，不能采取将要改变的变量值作为参数的方法，而应该用指针变量作为函数参数，在函数执行过程中使指针变量所指向的变量值发生变化，函数调用结束后，这些变量值的变化依然保留下来，这样就实现了通过调用函数使变量的值发生变化，在主调函数中可以使用这些改变了的值的目的。

例8.3 对输入的两个整数按大小顺序输出，要求通过函数处理，并且用指针类型的数据作为函数参数。

```c
#include<stdio.h>
int main(){
    void swap(int *p1,int *p2);
    int a,b;
    int *pointer_1,*pointer_2;
    printf("请输入两个整数: ");
    scanf("%d %d",&a,&b);
    pointer_1 = &a;
    pointer_2 = &b;
    if(a<b){
        swap(pointer_1, pointer_2);
    }
    printf("a = %d,b = %d\n",a,b);
    printf("max = %d,min = %d\n",*pointer_1,*pointer_2);
    return 0;
}
void swap(int *p1,int *p2){
    int temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
```

请看运行效果：

```c
#include<stdio.h>
int main(){
    void swap(int *p1,int *p2);
    int a,b;
    int *pointer_1,*pointer_2;
    printf("请输入两个整数: ");
    scanf("%d %d",&a,&b);
    pointer_1 = &a;
    pointer_2 = &b;
    if(a<b){
        swap(pointer_1, pointer_2);
    }
    printf("a = %d,b = %d\n",a,b);
    printf("max = %d,min = %d\n",*pointer_1,*pointer_2);
    return 0;
}
void swap(int *p1,int *p2){
    int temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
```

```
请输入两个整数: 1 2
a = 2,b = 1
max = 2,min = 1
Program ended with exit code: 0
```

`C`语言中实参变量和形参变量之间的数据传递是单向的值传递方式，用指针变量作函数参数时同样要遵循这一规则，不可能通过执行调用函数来改变实参指针变量的值，但是可以改变实参指针变量所指变量的值。

例8.4 输入3个整数`a`、`b`、`c`，要求按由大到小的顺序将它们输出。

```c
#include<stdio.h>
int main(){
    void exchange(int *p1,int *p2,int *p3);
    int a,b,c,*p1,*p2,*p3;
    printf("请输入三个整数: ");
    scanf("%d %d %d",&a,&b,&c);
    p1 = &a;
    p2 = &b;
    p3 = &c;
    exchange(p1, p2, p3);
    printf("由大到小的顺序为:%d,%d,%d\n",a,b,c);
    return 0;
}
void exchange(int *p1,int *p2,int *p3){
    void swap(int *q1,int *q2);
    if(*p1 < *p2){
```

```
        swap(p1, p2);
    };
    if (*p1 < *p3){
        swap(p1, p3);
    };
    if(*p2 < *p3){
        swap(p2, p3);
    }
}
void swap(int *q1,int *q2){
    int temp;
    temp = *q1;
    *q1 = *q2;
    *q2 = temp;
}
```

请看演示效果：

```
#include<stdio.h>
int main(){
    void exchange(int *p1,int *p2,int *p3);
    int a,b,c,*p1,*p2,*p3;
    printf("请输入三个整数: ");
    scanf("%d %d %d",&a,&b,&c);
    p1 = &a;
    p2 = &b;
    p3 = &c;
    exchange(p1, p2, p3);
    printf("由大到小的顺序为:%d,%d,%d\n",a,b,c);
    return 0;
}
void exchange(int *p1,int *p2,int *p3){
    void swap(int *q1,int *q2);
    if(*p1 < *p2){
        swap(p1, p2);
    };
    if (*p1 < *p3){
        swap(p1, p3);
    };
    if(*p2 < *p3){
        swap(p2, p3);
    }
}
```

```
请输入三个整数: 1 2 3
由大到小的顺序为:3,2,1
Program ended with exit code: 0
```

8.3 通过指针引用数组

8.3.1 数组元素的指针

所谓数组元素的指针就是数组元素的地址，在C语言中，数组名代表数组中首元素的地址，因此，存在以下等价语句：

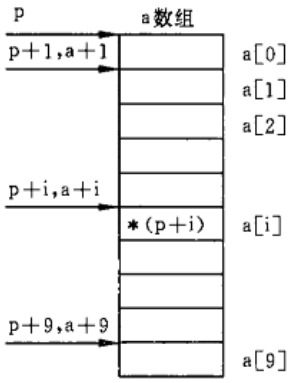
```
int *p = &a[0];
int *p = a[0];
```

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，可以对指针进行以下运算：

- 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；

以上四种指针运算，系统会根据p的基类型来移动指针的位置。



- 如果两个指针变量p1和p2都指向同一数组，如执行p1-p2，结果是p2-p1的值除以数组元素的长度。

8.3.3 通过指针引用数组元素

引用数组元素的两种方法：

- 下标法：如a[i]形式；
- 指针法：如*(a+i)或*(p+i)，其中a是数组名，p是指向数组元素的指针变量；

例8.5 输出数组中的全部元素。

- 下标法

```
#include<stdio.h>
int main(){
    int arr[10];
```

```
    printf("请输入数组元素: ");
    for (int i = 0; i < 10; i++) {
        scanf("%d",&arr[i]);
    }
    printf("数组元素分别为: ");
    for (int i = 0; i < 10; i++) {
        printf("%3d",arr[i]);
    }
    printf("\n");
    return 0;
}
```

请看演示效果:

```
#include<stdio.h>
int main(){
    int arr[10];
    printf("请输入数组元素: ");
    for (int i = 0; i < 10; i++) {
        scanf("%d",&arr[i]);
    }
    printf("数组元素分别为: ");
    for (int i = 0; i < 10; i++) {
        printf("%3d",arr[i]);
    }
    printf("\n");
    return 0;
}
```

```
请输入数组元素: 0 1 2 3 4 5 6 7 8 9
数组元素分别为:   0  1  2  3  4  5  6  7  8  9
Program ended with exit code: 0
```

- 通过数组名计算数组元素地址，找出元素的值

```
#include<stdio.h>
int main(){
    int arr[10];
    printf("请输入数组元素: ");
    for (int i = 0; i < 10; i++) {
        scanf("%d",arr+i);
    }
    printf("数组元素分别为: ");
    for (int i = 0; i < 10; i++) {
        printf("%3d",*(arr+i));
    }
    printf("\n");
    return 0;
}
```

请看演示效果：

```
8
9  #include<stdio.h>
10 int main(){
11     int arr[10];
12     printf("请输入数组元素：");
13     for (int i = 0; i < 10; i++) {
14         scanf("%d",arr+i);
15     }
16     printf("数组元素分别为：");
17     for (int i = 0; i < 10; i++) {
18         printf("%3d",*(arr+i));
19     }
20     printf("\n");
21     return 0;
22 }
23
```

```
请输入数组元素：0 1 2 3 4 5 6 7 8 9
数组元素分别为：  0  1  2  3  4  5  6  7  8  9
Program ended with exit code: 0
```

- 用指针变量指向数组元素

```
1  #include<stdio.h>
2  int main(){
3      int arr[10],*p;
4      printf("请输入数组元素：");
5      for (int i = 0; i < 10; i++) {
6          scanf("%d",arr+i);
7      }
8      printf("数组元素分别为：");
9      for (p = arr;p < (arr+10); p++) {
10         printf("%3d",*p);
11     }
12     printf("\n");
13     return 0;
14 }
```

请看演示效果：

```c
#include<stdio.h>
int main(){
    int arr[10],*p;
    printf("请输入数组元素: ");
    for (int i = 0; i < 10; i++) {
        scanf("%d",arr+i);
    }
    printf("数组元素分别为: ");
    for (p = arr;p < (arr+10); p++) {
        printf("%3d",*p);
    }
    printf("\n");
    return 0;
}
```

```
请输入数组元素: 0 1 2 3 4 5 6 7 8 9
数组元素分别为:   0  1  2  3  4  5  6  7  8  9
Program ended with exit code: 0
```

`C`编译系统是将`arr[i]`转换为`*(arr+i)`处理的，即先计算元素地址，因此前两种方法相比第三种方法，较为耗时。

如果`p`当前指向`a`数组中第`i`个元素`a[i]`，则：

- `*(p--)`相当于`a[i--]`，即先对`p`进行`*`运算，在执行自减操作；
- `*(++p)`相当于`a[++i]`，即先对`p`执行自加操作，在进行`*`运算；

```c
#include<stdio.h>
int main(){
    int arr[10] = {0,1,2,3,4,5,6,7,8,9},*p;
    p = arr;
    printf("*(p++) = %d\n",*(p++));
    p = arr;
    printf("*(++p) = %d\n",*(++p));
    p = arr;
    printf("++(*p) = %d\n",++(*p));
    return 0;
}
```

请看运行效果：

```c

#include<stdio.h>
int main(){
    int arr[10] = {0,1,2,3,4,5,6,7,8,9},*p;
    p = arr;
    printf("*(p++) = %d\n",*(p++));
    p = arr;
    printf("*(++p) = %d\n",*(++p));
    p = arr;
    printf("++(*p) = %d\n",++(*p));
    return 0;
}

```

```
*(p++) = 0
*(++p) = 1
++(*p) = 1
Program ended with exit code: 0
```

8.3.4 用数组名作函数参数

例8.6 将数组中的元素逆序存放

```c
#include<stdio.h>
int main(){
    void inverse(int arr[],int n);
    int i,arr[10]  = {0,1,2,3,4,5,6,7,8,9};
    printf("原始数组为：\n");
    for (i = 0; i < 10; i++) {
        printf("%3d",arr[i]);
    }
    printf("\n");
    inverse(arr, 10);
    printf("逆序之后的数组为：\n");
    for (i = 0; i < 10; i++) {
        printf("%3d",arr[i]);
    }
    printf("\n");
    return 0;
}
void inverse(int *arr,int n){
    int *p,temp,*i,*j,m = (n-1)/2;
    i = arr;
    j = arr+n-1;
    p = arr+m;
    for (; i <= p; i++,j--) {
        temp = *i;
        *i = *j;
```

```
        *j = temp;
    }
}
```

请看运行效果：

```
#include<stdio.h>
int main(){
    void inverse(int arr[],int n);
    int i,arr[10]  = {0,1,2,3,4,5,6,7,8,9};
    printf("原始数组为：\n");
    for (i = 0; i < 10; i++) {
        printf("%3d",arr[i]);
    }
    printf("\n");
    inverse(arr, 10);
    printf("逆序之后的数组为：\n");
    for (i = 0; i < 10; i++) {
        printf("%3d",arr[i]);
    }
    printf("\n");
    return 0;
}
void inverse(int *arr,int n){
    int *p,temp,*i,*j,m = (n-1)/2;
    i = arr;
    j = arr+n-1;
    p = arr+m;
    for (; i <= p; i++,j--) {
        temp = *i;
        *i = *j;
        *j = temp;
    }
}
```

```
原始数组为：
  0  1  2  3  4  5  6  7  8  9
逆序之后的数组为：
  9  8  7  6  5  4  3  2  1  0
Program ended with exit code: 0
```

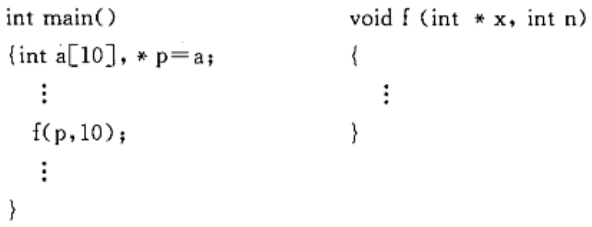
如果有一个实参数组，要想在函数中改变此数组中的元素的值，实参与形参的对应关系有以下4种情况：

- 形参和实参都用数组名；

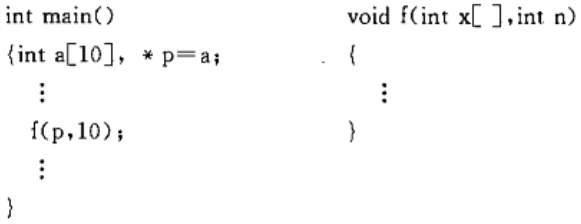
```
int main()
{int a[10];
   ⋮
 f(a,10);
   ⋮
}
```

```
int f(int x[], int n)
{
   ⋮
}
```

- 实参用数组名，形参用指针变量；
- 实参形参都用指针变量；



- 实参为指针变量，形参为数组名；



用指针变量作实参：

```
#include<stdio.h>
int main(){
    void inverse(int *arr,int n);
    int i,arr[10]  = {0,1,2,3,4,5,6,7,8,9},*p = arr;
    printf("原始数组为：\n");
    for (i = 0; i < 10; i++,p++) {
        printf("%3d",*p);
    }
    printf("\n");
    p = arr;
    inverse(p, 10);
    printf("逆序之后的数组为：\n");
    for (p = arr;p < arr+10; p++) {
        printf("%3d",*p);
    }
    printf("\n");

    return 0;
}
void inverse(int *arr,int n){
    int *p,m,temp,*i,*j;
    m = (n-1)/2;
    i = arr;
    j = arr+n-1;
    p  = arr+m;
    for (;i <= p ; i++,j--) {
        temp = *i;
        *i = *j;
        *j = temp;
    }
}
```

运行效果如下所示：

```
#include<stdio.h>
int main(){
    void inverse(int *arr,int n);
    int i,arr[10]  = {0,1,2,3,4,5,6,7,8,9},*p = arr;
    printf("原始数组为：\n");
    for (i = 0; i < 10; i++,p++) {
        printf("%3d",*p);
    }
    printf("\n");
    p = arr;
    inverse(p, 10);
    printf("逆序之后的数组为：\n");
    for (p = arr;p < arr+10; p++) {
        printf("%3d",*p);
    }
    printf("\n");

    return 0;
}
void inverse(int *arr,int n){
    int *p,m,temp,*i,*j;
    m = (n-1)/2;
```

```
原始数组为：
  0  1  2  3  4  5  6  7  8  9
逆序之后的数组为：
  9  8  7  6  5  4  3  2  1  0
Program ended with exit code: 0
```

例8.7 用指针方法对10个整数按由大到小的顺序排序。

```
#include<stdio.h>
int main(){
    void sort(int arr[],int n);
    int *p,arr[10];
    p = arr;
    printf("请输入数组元素：");
    for (int i = 0; i < 10; i++) {
        scanf("%d",p++);
    }
    p = arr;
    sort(p, 10);
    printf("由大到小的顺序为：\n");
    for (p = arr; p < (arr+10); p++) {
        printf("%3d",*p);
    }
    printf("\n");
    return 0;
}
```

```
void sort(int *arr,int n){
    int i,j,k,t;
    for (i = 0; i < n-1; i++) {
        k = i;
        for (j = k+1; j < n; j++) {
            if(*(arr+j)>*(arr+k)){
                k = j;
            }
        }
        if(k!=i){
            t = arr[i];
            arr[i] = arr[k];
            arr[k] = t;
        }
    }
}
```

运行效果如下所示：

```c
#include<stdio.h>
int main(){
    void sort(int arr[],int n);
    int *p,arr[10];
    p = arr;
    printf("请输入数组元素：");
    for (int i = 0; i < 10; i++) {
        scanf("%d",p++);
    }
    p = arr;
    sort(p, 10);
    printf("由大到小的顺序为：\n");
    for (p = arr; p < (arr+10); p++) {
        printf("%3d",*p);
    }
    printf("\n");
    return 0;
}
void sort(int *arr,int n){
    int i,j,k,t;
    for (i = 0; i < n-1; i++) {
        k = i;
        for (j = k+1; j < n; j++) {
            if(*(arr+j)>*(arr+k)){
                k = j;
            }
        }
        if(k!=i){
```

```
请输入数组元素：3 1 5 4 3 2 7 5 0 8
由大到小的顺序为：
  8  7  5  5  4  3  3  2  1  0
Program ended with exit code: 0
```

8.3.5 通过指针运用多维数组

1. 多维数组元素的地址

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

从二维数组的角度来看，`arr`表示二维数组首元素的地址，现在的首元素不是一个简单的整型元素，而是由4个整型元素组成的一维数组，因此`arr`代表的是首行的首地址，`arr+1`代表序号为1的行的首地址，如果二维数组的首行的首地址为2000，一个整型数据占4个字节，则`a+1`的值应该是2000+4*4=2016，`arr+1`指向`arr[1]`，或者说`arr+1`的值是`arr[1]`的首地址，`arr+2`代表`arr[2]`的首地址。

`arr[0]`、`arr[1]`、`arr[2]`既然是一维数组名，而C语言又规定了数组名代表数组首元素地址，因此`arr[0]`代表一维数组`arr[0]`中第0列元素的地址，即`&a[0][0]`，也就是说`arr[1]`的值是`&arr[1][0]`，`arr[2]`的值是`&arr[2][0]`。

表 8.2　二维数组 a 的有关指针

表示形式	含　义	地　址
a	二维数组名，指向一维数组 a[0]，即 0 行首地址	2000
a[0]，*(a+0)，*a	0 行 0 列元素地址	2000
a+1，&a[1]	1 行首地址	2016
a[1]，*(a+1)	1 行 0 列元素 a[1][0]的地址	2016
a[1]+2，*(a+1)+2，&a[1][2]	1 行 2 列元素 a[1][2]的地址	2024
(a[1]+2)，(*(a+1)+2)，a[1][2]	1 行 2 列元素 a[1][2]的值	元素值为 13

例8.8 输出二维数组的有关数据。

```c
#include<stdio.h>
int main(){
    int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
    printf("%d %d\n",arr,*arr);
    printf("%d %d\n",arr[0],*(arr+0));
    printf("%d %d\n",&arr[0],&arr[0][0]);
    printf("%d %d\n",arr[1],arr+1);
    printf("%d %d\n",&arr[1][0],*(arr+1)+0);
    printf("%d %d\n",arr[2],*(arr+2));
    printf("%d %d\n",&arr[2],arr+2);
    printf("%d %d\n",arr[1][0],*(*(arr+1)+0));
    printf("%d %d\n",*arr[2],*(*(arr+2)+0));
    return 0;
}
```

请看演示效果：

```
8
9  #include<stdio.h>
10 int main(){
11     int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
12     printf("%d %d\n",arr,*arr);                          2 ⚠ Format spe
13     printf("%d %d\n",arr[0],*(arr+0));                     ⚠ Format
14     printf("%d %d\n",&arr[0],&arr[0][0]);                2 ⚠ Format spe
15     printf("%d %d\n",arr[1],arr+1);                      2 ⚠ Format spe
16     printf("%d %d\n",&arr[1][0],*(arr+1)+0);               ⚠ Format
17     printf("%d %d\n",arr[2],*(arr+2));                     ⚠ Format
18     printf("%d %d\n",&arr[2],arr+2);                     ⚠ Format spe
19     printf("%d %d\n",arr[1][0],*(*(arr+1)+0));
20     printf("%d %d\n",*arr[2],*(*(arr+2)+0));
21     return 0;
22 }
23
24
```

```
-272632608 -272632608
-272632608 -272632608
-272632608 -272632608
-272632592 -272632592
-272632592 -272632592
-272632576 -272632576
-272632576 -272632576
5 5
9 9
Program ended with exit code: 0
```

2. 指向多维数组元素的指针变量

- 指向数组元素的指针变量

例8.9 使用指针元素输出二维数组元素值。

```
1  #include<stdio.h>
2  int main(){
3      int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
4      int *p;
5      for (p = arr[0]; p < arr[0]+12; p++) {
6          if((p-arr[0])%4==0){
7              printf("\n");
8          }
9          printf("%3d",*p);
10     }
11     printf("\n");
12     return 0;
13 }
```

请看演示效果：

```c
#include<stdio.h>
int main(){
    int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
    int *p;
    for (p = arr[0]; p < arr[0]+12; p++) {
        if((p-arr[0])%4==0){
            printf("\n");
        }
        printf("%3d",*p);
    }
    printf("\n");
    return 0;
}
```

```
  1  2  3  4
  5  6  7  8
  9 10 11 12
Program ended with exit code: 0
```

- 指向由`m`个元素组成的一维数组的指针变量

例8.10 输出二维数组特定行和列的元素值。

```c
#include<stdio.h>
int main(){
    int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
    //(*p)[4]表示定义p为一个指针变量，它指向包含4个整型元素的一维数组。
    int *pointer,(*p)[4],i,j;
    printf("原数组如下所示：\n");
    for (pointer = arr[0]; pointer < arr[0]+12; pointer++) {
        if((pointer-arr[0])%4==0){
            printf("\n");
        }
        printf("%3d",*pointer);
    }
    printf("\n");
    p = arr;
    printf("请输入行和列：");
    scanf("%d %d",&i,&j);
    printf("arr[%d][%d]=%d\n",i,j,*(*(p+i)+j));
    return 0;
}
```

请看演示效果：

```c
#include<stdio.h>
int main(){
    int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
    int *pointer,(*p)[4],i,j;
    printf("原数组如下所示：\n");
    for (pointer = arr[0]; pointer < arr[0]+12; pointer++) {
        if((pointer-arr[0])%4==0){
            printf("\n");
        }
        printf("%3d",*pointer);
    }
    printf("\n");
    p = arr;
    printf("请输入行和列：");
    scanf("%d %d",&i,&j);
    printf("arr[%d][%d]=%d\n",i,j,*(*(p+i)+j));
    return 0;
}
```

```
原数组如下所示：

  1  2  3  4
  5  6  7  8
  9 10 11 12
请输入行和列：1 2
arr[1][2]=7
Program ended with exit code: 0
```

3. 用指向数组的指针作函数参数

例8.11 计算3个学生（每个学生4门课）的总平均分以及第`n`个学生的成绩。

```c
#include<stdio.h>
int main(){
    void average(float *p,int n);
    void search(float (*p)[4],int n);
    float score[3][4] = {
        {78,67,89,78},
        {34,67,54,90},
        {89,90,67,89}
    };
    average(*score, 12);
    search(score, 2);
    return 0;
}

void average(float *p,int n){
    float sum = 0,aver;
    float *p_end = p+n-1;
    for (; p <= p_end; p++) {
        sum += (*p);
```

```
    }
    aver = sum / n;
    printf("aver=%5.2f\n",aver);
}
void search(float (*p)[4],int n){
    printf("第%d个学生成绩为:",n);
    for (int i = 0; i < 4; i++) {
        printf("%10.2f",*(*(p+n)+i));
    }
    printf("\n");
}
```

请看演示效果：

```
#include<stdio.h>
int main(){
    void average(float *p,int n);
    void search(float (*p)[4],int n);
    float score[3][4] = {
        {78,67,89,78},
        {34,67,54,90},
        {89,90,67,89}
    };
    average(*score, 12);
    search(score, 2);
    return 0;
}

void average(float *p,int n){
    float sum = 0,aver;
    float *p_end = p+n-1;
    for (; p <= p_end; p++) {
        sum += (*p);
    }
    aver = sum / n;
    printf("aver=%5.2f\n",aver);
}
void search(float (*p)[4],int n){
    printf("第%d个学生成绩为:",n);
    for (int i = 0; i < 4; i++) {
        printf("%10.2f",*(*(p+n)+i));
    }
    printf("\n");
```

```
aver=74.33
第2个学生成绩为:     89.00     90.00     67.00     89.00
Program ended with exit code: 0
```

例8.12 在例8.11的基础上找出有一门以上课程不及格的学生，并输出全部课程的成绩

```
#include<stdio.h>
int main(){
    void search(float (*p)[4],int n);
    float score[3][4] = {
        {78,67,89,78},
```

```
        {34,67,54,90},
        {89,90,67,89}
    };
    search(score, 3);
    return 0;
}

void search(float (*p)[4],int n){
    int i,j,flag;
    for (i = 0; i < n; i++) {
        flag = 0;
        for (j = 0; j < 4; j++) {
            if(*(*(p+i)+j) < 60){
                flag = 1;
            }
        }
        if(flag == 1){
            printf("第%d个学生成绩不合格,其各科成绩如下：\n",(i+1));
            for (j = 0; j < 4; j++) {
                printf("%10.2f",*(*(p+i)+j));
            }
            printf("\n");
        }
    }
}
```

请看演示效果：

```c
#include<stdio.h>
int main(){
    void search(float (*p)[4],int n);
    float score[3][4] = {
        {78,67,89,78},
        {34,67,54,90},
        {89,90,67,89}
    };
    search(score, 3);
    return 0;
}

void search(float (*p)[4],int n){
    int i,j,flag;
    for (i = 0; i < n; i++) {
        flag = 0;
        for (j = 0; j < 4; j++) {
            if(*(*(p+i)+j) < 60){
                flag = 1;
            }
        }
        if(flag == 1){
            printf("第%d个学生成绩不合格,其各科成绩如下: \n",(i+1));
            for (j = 0; j < 4; j++) {
                printf("%10.2f",*(*(p+i)+j));
```

```
第2个学生成绩不合格,其各科成绩如下:
     34.00     67.00     54.00     90.00
Program ended with exit code: 0
```

8.4 通过指针引用字符串

8.4.1 字符串的引用方式

在C程序中，要想引用一个字符串，可以有以下两种方法：

- 用字符数组存放一个字符串，可以通过数组名和下标引用字符串中一个字符，也可以通过数组名和格式声明`%s`输出该字符串；

```c
#include<stdio.h>
int main(){
    char string[] = "I am hahaCoder!";
    printf("%s\n",string);
    printf("%c\n",string[7]);
    return 0;
}
```

- 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；

```c
#include<stdio.h>
int main(){
    char *string = "I am hahaCoder!";
    printf("%s\n",string);
    return 0;
}
```

对字符指针变量`string`初始化，实际上是把字符串第1个元素的地址（即存放字符串的字符数组的首元素地址）赋给指针变量`string`，使`string`指向字符串的第1个字符，在输出项中给出字符指针变量名`string`，则系统会输出`string`所指向的字符串第1个字符，然后自动使`string`加1，使之指向下一个字符，在输出字符……如此直到遇到字符串结束标志`\0`为止。

「注」：通过字符数组名或字符指针变量可以输出一个字符串，而对一个数值型数组，是不能企图用数组名输出它的全部元素的。

例8.13 将字符串`a`复制为字符串`b`，然后输出字符串`b`。

- 地址法

```c
#include<stdio.h>
int main(){
    char arr[] = "I am hahaCoder!",arr_cpy[20];
    int i;
    for (i = 0; *(arr+i)!='\0'; i++) {
        *(arr_cpy+i) = *(arr+i);
    }
    *(arr_cpy+i) = '\0';
    printf("字符串arr为：%s\n",arr);
    printf("字符串arr_cpy为:%s\n",arr_cpy);
    return 0;
}

```

- 指针变量法

```c
#include<stdio.h>
int main(){
    char arr[] = "I am hahaCoder!",arr_cpy[20],*p_arr,*p_arrCpy;
    p_arr = arr;
    p_arrCpy = arr_cpy;
    for (; *p_arr!='\0'; p_arr++,p_arrCpy++) {
        *p_arrCpy = *p_arr;
    }
    *p_arrCpy = '\0';
    printf("字符串arr为：%s\n",arr);
    printf("字符串arr_cpy为:%s\n",arr_cpy);
    return 0;
}
```

8.4.2 字符指针作函数参数

例8.14 用函数调用实现字符串的复制。

- 用字符数组名作为函数参数

```c
#include<stdio.h>
int main(){
    void cpy_string(char source[],char dest[]);
    char arr_source[] = "I am hahaCoder!";
    char arr_dest[] = "www.shipudong.com!";
    printf("arr_source=%s\narr_dest=%s\n",arr_source,arr_dest);
    printf("将字符串arr_source复制给arr_dest:\n");
    cpy_string(arr_source, arr_dest);
    printf("arr_source=%s\narr_dest=%s\n",arr_source,arr_dest);
    return 0;
}
void cpy_string(char source[],char dest[]){
    int i = 0;
    while (source[i]!='\0') {
        dest[i] = source[i];
        i++;
    }
    dest[i] = '\0';
}
```

- 用字符指针变量作实参

```c
#include<stdio.h>
int main(){
    void cpy_string(char source[],char dest[]);
    char arr_source[] = "I am hahaCoder!";
    char arr_dest[] = "www.shipudong.com!";
    char *source = arr_source,*dest = arr_dest;
    printf("arr_source=%s\narr_dest=%s\n",arr_source,arr_dest);
    printf("将字符串arr_source复制给arr_dest:\n");
    cpy_string(source, dest);
    printf("arr_source=%s\narr_dest=%s\n",arr_source,arr_dest);
    return 0;
}
void cpy_string(char source[],char dest[]){
    int i = 0;
    while (source[i]!='\0') {
        dest[i] = source[i];
        i++;
    }
    dest[i] = '\0';
```

```
}
```

- 用字符指针变量作实参和形参

```
#include<stdio.h>
int main(){
    void cpy_string(char source[],char dest[]);
    char *arr_source = "I am hahaCoder!";
    char arr_dest[] = "www.shipudong.com!";
    char *dest = arr_dest;
    printf("arr_source=%s\narr_dest=%s\n",arr_source,arr_dest);
    printf("将字符串arr_source复制给arr_dest:\n");
    cpy_string(arr_source, dest);
    printf("arr_source=%s\narr_dest=%s\n",arr_source,arr_dest);
    return 0;
}
void cpy_string(char *source,char *dest){
    for (; *source!='\0'; source++,dest++) {
        *dest = *source;
    }
    *dest = '\0';
}
```

8.4.3 使用字符指针变量和字符数组的比较

- 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字符串第1个字符的地址；
- 可以对字符指针变量赋值，但不能对数组名赋值；
- 编译时为字符数组分配若干存储单元，以存储各元素的值，而对字符指针变量，只分配一个存储单元；
- 指针变量的值是可以改变的，而数组名代表一个固定的值，不能改变；

```
#include<stdio.h>
int main(){
    char *arr = "I love my motherland!";
    arr = arr+10;
    printf("%s\n",arr);
    return 0;
}
```

- 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；
- 用指针变量指向一个格式字符串，可以用它代替`printf`函数中的格式字符串；

- 指针变量

```c
#include<stdio.h>
int main(){
    int a = 10;
    float b = 10.02;
    char *format = "a=%d,b=%.2f\n";
    printf(format,a,b);
    return 0;
}
```

- 字符数组

```c
#include<stdio.h>
int main(){
    int a = 10;
    float b = 10.02;
    char format[] = "a=%d,b=%.2f\n";
    printf(format,a,b);
    return 0;
}
```

8.5 指向函数的指针

8.5.1 什么是函数指针

如果在程序中定义了一个函数，在编译时，编译系统为函数代码分配一段存储空间，这段存储空间的起始地址称为这个函数的指针，可以定义一个指向函数的指针变量，用来存放某一函数的起始地址，这就意味着此指针变量指向该函数，如`int (*p)(int,int)`定义`p`是一个指向函数的指针变量，它指向一个整型函数，且具有两个整型参数。

8.5.2 用函数指针变量调用函数

调用函数的两种方法：

- 函数名调用；

```c
#include<stdio.h>
int main(){
    int max(int x,int y);
    int a,b,c;
    printf("请输入a、b的值：");
    scanf("%d %d",&a,&b);
    c = max(a,b);
    printf("a和b中的较大者为:%d\n",c);
    return 0;
}
int max(int x,int y){
    return x>y?x:y;
}
```

- 指向函数的指针变量；

```c
#include<stdio.h>
int main(){
    int max(int x,int y);
    int (*p)(int ,int);
    int a,b,c;
    p = max;
    printf("请输入a、b的值：");
    scanf("%d %d",&a,&b);
    c = (*p)(a,b);
    printf("a和b中的较大者为:%d\n",c);
    return 0;
}

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

8.5.3 怎样定义和使用指向函数的指针变量

定义指向函数的指针变量的一般形式为：`类型名 (*指针变量名)(函数参数列表)`。

- 对指向函数的指针变量不能进行算术运算；
- 用函数名调用函数，只能调用所指定的一个函数，而通过指针变量调用函数比较灵活，可以根据不同情况先后调用不同的函数；

例8.15 用户根据数字编号选择不同的函数功能。

```c
#include<stdio.h>
int main(){
```

```
    int max(int x,int y);
    int min(int x,int y);
    int (*p)(int ,int);
    int a,b,num;
    printf("请输入a、b的值：");
    scanf("%d %d",&a,&b);
    printf("请选择max(编号1)函数或min(编号2)函数：");
    scanf("%d",&num);
    if(num==1){
        p = max;
    }else if (num==2){
        p = min;
    }
    c = (*p)(a,b);
    if(num==1){
        printf("a和b中的较大者为:%d\n",c);
    }else{
        printf("a和b中的较小者为:%d\n",c);
    }
    return 0;
}

int max(int x,int y){
    return x>y?x:y;
}
int min(int x,int y){
    return x<y?x:y;
}
```

8.5.4 用指向函数的指针做函数参数

函数`fun`有两个形参`x1`和`x2`，定义`x1`和`x2`为指向函数的指针变量，在调用函数`fun`时，实参为两个函数名`f1`和`f2`，给形参传递的是函数`f1`和`f2`的入口地址，这样在函数`fun`中就可以调用`f1`和`f2`了。

例8.16 输入两个整数，完成`max`、`min`、`add`三个功能。

```
#include<stdio.h>
int main(){
    void fun(int x,int y,int (*p)(int,int));
    int max(int x,int y);
    int min(int x,int y);
    int add(int x,int y);
    int a,b,num;
    printf("请输入a、b的值：");
    scanf("%d %d",&a,&b);
    printf("请选择max(编号1)函数、min(编号2)函数或add(编号3)函数：");
```

```
    scanf("%d",&num);
    if(num==1){
        fun(a, b, max);
    }else if (num==2){
        fun(a, b, min);
    }else if(num==3){
        fun(a, b, add);
    }
    return 0;
}
void fun(int x,int y,int (*p)(int,int)){
    int result;
    result = (*p)(x,y);
    printf("result=%d\n",result);
}

int max(int x,int y){
    return x>y?x:y;
}
int min(int x,int y){
    return x<y?x:y;
}
int add(int x,int y){
    return x+y;
}
```

8.6 返回指针的函数

定义返回指针值的函数的一般形式为：`类型名 *函数名(参数表列)`。

例8.17 有`a`个学生，每个学生有`b`门课程的成绩，要求输入学生序号后，能输出该学生的全部成绩。

```
#include<stdio.h>
int main(){
    float score[][4] = {
        {78,86,95,67},
        {76,85,92,69},
        {57,81,92,69}
    };
    float *search(float (*pointer)[4],int n);
    float *p;
    int num;
    printf("请输入学生编号：");
    scanf("%d",&num);
    p = search(score, num);
    for (int i = 0; i < 4; i++) {
        printf("%5.2f\t",*(p+i));
```

```
    }
    printf("\n");
    return 0;
}

float *search(float (*pointer)[4],int n){
    return *(pointer+n);
}
```

例8.18 找出例8.17中不及格的学生。

```
#include<stdio.h>
int main(){
    float score[][4] = {
        {78,86,95,67},
        {76,85,92,69},
        {57,81,92,69}
    };
    float *search(float (*pointer)[4]);
    float *p;
    for (int i = 0; i < 3; i++) {
        p = search(score+i);
        if(p==*(score+i)){
            printf("第%d个学生成绩不合格。\n",i+1);
            for (int j = 0; j < 4;j++) {
                printf("%5.2f\t",*(p+j));
            }
            printf("\n");
        }
    }
    printf("\n");
    return 0;
}

float *search(float (*pointer)[4]){
    float *pt;
    pt = NULL;
    for (int i = 0; i < 4; i++) {
        if(*(*pointer+i) < 60){
            pt = *pointer;
        }
    }
    return pt;
}
```

8.7 指针数组和多重指针

8.7.1 什么是指针数组

定义一维指针数组的一般形式为：`类型名 *数组名[数组长度]`，指针数组比较适合用来指向若干个字符串，使字符串处理更佳方便灵活，如需将多个字符串存入一个数组中，并对其进行排序和查询操作，按照一般方法，字符串本身就是一个字符数组，因此要设计一个二维的字符数组才能存放多个字符串，但在定义二维数组时，需要指定列数，也就是说二维数组中每一行包含的元素个数相等，而实际上各字符串擦回归难度一般是不相等的，如按最长的字符串来定义列数，则会浪费许多内存单元。

因此，可以分别定义一些字符串，然后用指针数组中的元素分别指向各字符串，即指针数组中的元素分别存放各字符串首字符的地址，如果相对字符串排序，不必改动字符串的位置，只须改动指针数组中各元素的指向。

例8.19 将若干字符串按字母顺序由小到大输出。

```
#include<stdio.h>
#include<string.h>
int main(){
    void sort(char *name[],int n);
    void printStr(char *name[],int n);
    char *name[] =
{"www.shipudong.com","hahaOCR","hahaCoder","hahaAI","hahaWebsite."};
    int n = 5;
    printf("原字符数组为:\n");
    printStr(name, n);
    printf("\n");
    sort(name, n);
    printf("排序之后的字符数组为:\n");
    printStr(name, n);
    printf("\n");
    return 0;
}
void sort(char *name[],int n){
    char *temp;
    int i,j,k;
    for (i = 0; i < n-1; i++) {
        k = i;
        for (j = i+1; j < n; j++) {
            if(strcmp(name[k], name[j])>0){
                k = j;
            }
        }
        if(k!=i){
            temp = name[i];
            name[i] = name[k];
            name[k] = temp;
        }
    }
}

```

```
void printStr(char *name[],int n){
    for (int i = 0; i < n; i++) {
        printf("%s\t",name[i]);
    }
}
```

8.7.2 指向指针数据的指针

可以设置一个指针变量`p`，使其指向指针数组，则变量`p`即为指向指针型数据的指针变量，定义形式为：`char **p;`等价于`char *(*p)`。

例8.20 使用指向指针数据的指针变量。

```
#include<stdio.h>
int main(){
    char *name[] =
{"www.shipudong.com","hahaOCR","hahaCoder","hahaAI","hahaWebsite."};
    char **p;
    for (int i = 0; i < 5;i++) {
        p = name+i;
        printf("%s->%d\n",*p,*p);
    }
    return 0;
}
```

例8.21 使用指向指针数据的指针变量输出整型数组各元素的值。

```
#include<stdio.h>
int main(){
    int a[5] = {1,3,4,5,6};
    int *num[5] = {&a[0],&a[1],&a[2],&a[3],&a[4]};
    int **p;
    p = num;
    for (int i = 0; i < 5; i++,p++) {
        printf("地址:%d->元素值:%d\n",*p,**p);
    }
    return 0;
}
```

说明：指针数组的元素也可以不指向字符串，而指向整型数据或实型数据等，例如：

```
int a[5]={1,3,5,7,9};
int *num[5],i;
int **p;              //p 是指向 int * 型数据的指针变量
for (i=0; i<5; i++)
    num[i]=&a[i];
```

为了得到 a[2]中的数据 5，可以先使 p=num+2，然后输出**p。注意 * p 是 num[2]的值，而 num[2]的值是 a[2]的地址，因此**p 是 a[2]的值 5，见图 8.39。

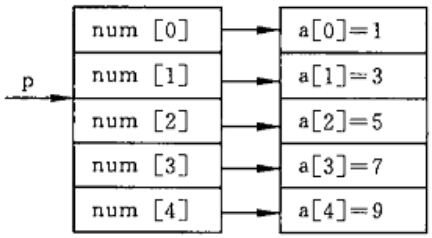


图 8.39

8.7.3 指针数组作`main`函数的形参

`main`函数的原型为：`int main(int argc,char *argv[])`，其中`argc`即`argument count`表示参数个数，`argv`即`argument vector`表示`* char`指针数组，数组中每一个元素指向命令行中的一个字符串。

8.8 动态内存分配与指向它的指针变量

8.8.1 什么是内存的动态分配

`C`语言允许建立内存动态分配区域，以存放一些临时用的数据，这些数据不必在程序的声明部分定义，也不必等到函数结束时才释放，而是需要时随时开辟，不需要时随时释放，这些数据是临时存放在一个特别的自由存储区，称为**堆**区，可以根据需要，向系统申请所需大小的空间，由于未在声明部分定义它们为变量或数组，因此不能通过变量名或数组名去引用这些数据，只能通过指针来引用。

8.8.2 怎样建立内存的动态分配

- `void *malloc(unsigned int size);`

其作用是在内存的动态存储区中分配一个长度为`size`的连续空间，此函数是一个指针型函数，返回的指针指向该分配域的开头位置，如果此函数未能成功地执行，则返回空指针`NULL`。

- `void *calloc(unsigned n,unsigned size);`

其作用是在内存的动态存储区中分配`n`个长度为`size`的连续空间，这个空间一般比较大，足以保存一个数组，函数返回指向所分配域的起始位置的指针，如果分配不成功，返回`NULL`。

- `void free(void *p);`

其作用是释放指针变量`p`所指向的动态空间，使这部分空间能重新被其他变量使用，`p`应该是最近一次调用`calloc`或`malloc`函数时得到的函数返回值，`free`函数无返回值。

- `void *realloc(void *p,unsigned int size);`

如果已经通过`calloc`或`malloc`函数获得了动态空间，想改变其大小，可以用`realloc`函数重新分配，用`realloc`函数将`p`所指向的动态空间的大小改变为`size`，`p`的值不变，如果重新分配不成功，返回`NULL`。

上述四个函数的声明均在`stdlib.h`头文件中。

8.8.3 `void`指针类型

例8.22 建立动态数组，存入学生成绩，并输出不合格的成绩。

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    void check(int *p);
    int *p1 = (int *)malloc(5*sizeof(int));
    for (int i = 0; i < 5; i++) {
        printf("请输入第%d个学生成绩: ",i+1);
        scanf("%d",p1+i);
    }
    printf("以下成绩不合格: \n");
    check(p1);
    return 0;
}
void check(int *p){
    for (int i = 0;i < 5; i++) {
        if(*(p+i)<60){
            printf("第%d个学生成绩为%d\n",i+1,*(p+i));
        }
    }
    printf("\n");
}
```

8.9 小节

表 8.4　指针变量的类型及含义

变量定义	类型表示	含　义
int i;	int	定义整型变量 i
int * p;	int *	定义 p 为指向整型数据的指针变量
int a[5]	int [5]	定义整型数组 a,它有 5 个元素
int * p[4];	int * [4]	定义指针数组 p,它由 4 个指向整型数据的指针元素组成
int (* p)[4];	int(*)[4]	p 为指向包含 4 个元素的一维数组的指针变量
int f();	int ()	f 为返回整型函数值的函数

续表

变量定义	类型表示	含　义
int *p();	int *()	p为返回一个指针的函数，该指针指向整型数据
int (*p)();	int (*)()	p为指向函数的指针，该函数返回一个整型值
int **p;	int **	p是一个指针变量，它指向一个指向整型数据的指针变量
void *p;	void *	p是一个指针变量，基类型为void(空类型)，不指向具体的对象

课后题：1、2、3、8、9、15、18、19