

***指向函数的指针**

什么是函数的指针

如果在程序中定义了一个函数，在编译时会把函数的源代码转换为可执行代码并分配一段存储空间。这段内存空间有一个起始地址，也称为函数的入口地址。每次调用函数时都从该地址入口开始执行此段函数代码。

函数名就是函数的指针，它代表函数的起始地址。

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。

例如: `int (*p)(int,int);`

定义p是一个指向函数的指针变量，它可以指向函数类型为整型且有两个整型参数的函数。此时，指针变量p的类型用`int (*)(int,int)`表示。

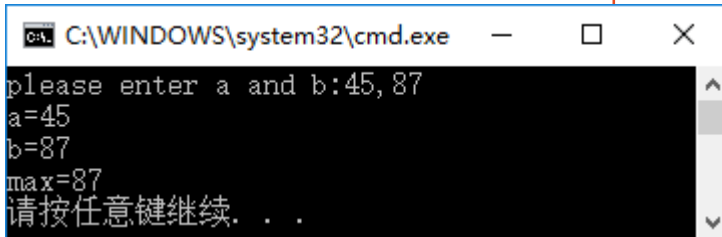
用函数指针变量调用函数

【例8.22】用函数求整数a和b中的大者。

(1)通过函数名调用函数

```
#include <stdio.h>
int main()
{
    int max(int,int);    //函数声明
    int a,b,c;
    printf("please enter a and b:");
    scanf("%d,%d",&a,&b);
    c=max(a,b);          //通过函数名调用max函数
    printf("a=%d\nb=%d\nmax=%d\n",a,b,c);
    return 0;
}

int max(int x,int y)    //定义max函数
{
    int z;
    if(x>y) z=x;
    else z=y;
    return(z);
}
```

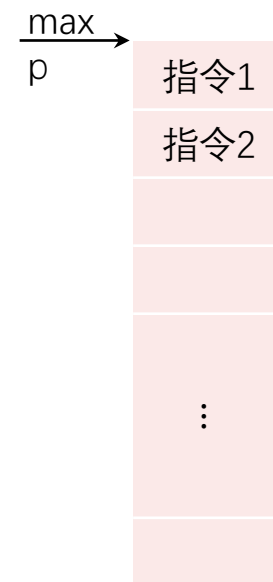


```
C:\WINDOWS\system32\cmd.exe
please enter a and b:45,87
a=45
b=87
max=87
请按任意键继续. . .
```

(2) 通过指针变量调用它所指向的函数

```
#include <stdio.h>
int main()
{
    int max(int,int);    //函数声明
    int (*p)(int,int);   //定义指向函数的指针变量p
    int a,b,c;
    p=max;               //使p指向max函数
    printf("please enter a and b:");
    scanf("%d,%d",&a,&b);
    c=(*p)(a,b);         //通过指针变量调用max函数
    printf("a=%d\nb=%d\nmax=%d\n",a,b,c);
    return 0;
}

int max(int x,int y)    //定义max函数
{
    int z;
    if(x>y)z=x;
    else z=y;
    return(z);
}
```



怎样定义和使用指向函数的指针变量

类型名 (*指针变量名)(函数参数表列)

```
int (*p)(int,int);
```

-
- (1) 定义指向函数的指针变量，并不意味着这个指针变量可以指向任何函数，它只能指向在定义时指定的类型的函数。
 - (2) 如果要用指针调用函数，必须先使指针变量指向该函数。
 - (3) 在给函数指针变量赋值时，只须给出函数名而不必给出参数。
 - (4) 用函数指针变量调用函数时，只须将(*p)代替函数名即可（p为指针变量名），在(*p)之后的括号中根据需要写上实参。
 - (5) 对指向函数的指针变量不能进行算术运算，如p+n,p++,p--等运算是无意义的。
 - (6) 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。
-

怎样定义和使用指向函数的指针变量

【例8.23】输入两个整数，然后让用户选择1或2，选1时调用max函数，输出二者中的大数，选2时调用min函数，输出二者中的小数。

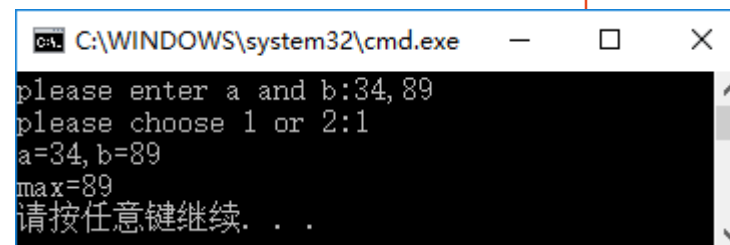
```
#include <stdio.h>
int main()
{
    int max(int,int);    //函数声明
    int min(int x,int y); //函数声明
    int (*p)(int,int);   //定义指向函数的指针变量
    int a,b,c,n;
    printf("please enter a and b:");
    scanf("%d,%d",&a,&b);
    printf("please choose 1 or 2:");
    scanf("%d",&n);    //输入1或2
    if(n==1) p=max;    //如输入1, 使p指向max函数
    else if (n==2) p=min; //如输入2, 使p指向min函数
    c=(*p)(a,b);       //调用p指向的函数
    printf("a=%d,b=%d\n",a,b);
    if(n==1) printf("max=%d\n",c);
    else printf("min=%d\n",c);
    return 0;
}
```

```
int max(int x,int y)
```

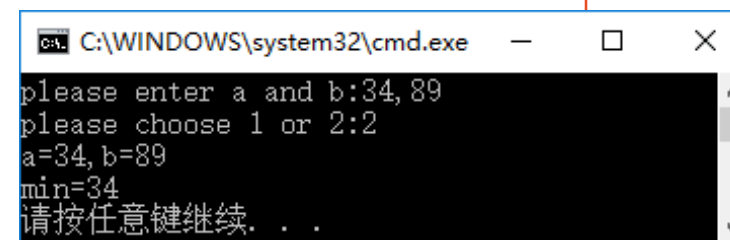
```
{
    int z;
    if(x>y) z=x;
    else z=y;
    return(z);
}
```

```
int min(int x,int y)
```

```
{
    int z;
    if(x<y) z=x;
    else z=y;
    return(z);
}
```



```
C:\WINDOWS\system32\cmd.exe
please enter a and b:34, 89
please choose 1 or 2:1
a=34, b=89
max=89
请按任意键继续. . .
```



```
C:\WINDOWS\system32\cmd.exe
please enter a and b:34, 89
please choose 1 or 2:2
a=34, b=89
min=34
请按任意键继续. . .
```

用指向函数的指针作函数参数

指向函数的指针变量的一个重要用途是把函数的入口地址作为参数传递到其他函数。

指向函数的指针可以作为函数参数，把函数的入口地址传递给形参，这样就能够和被调用的函数中使用实参函数。它的原理可以简述如下：有一个函数（假设函数名为fun），它有两个形参（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函数了。

实参函数名 f1

f2

```
void fun(int (*x1)(int), int(*x2) (int,int))  
{  
    int a,b,i=3,j=5;  
    a=(*x1)(i);  
    b=(*x2)(i,j);  
}
```

//定义fun函数，形参是指向函数的指针变量

//调用f1函数，i是实参

//调用f2函数，i,j是实参

用指向函数的指针作函数参数

【例8.24】有两个整数a和b，由用户输入1,2或3。如输入1，程序就给出a和b中的大者，输入2，就给出a和b中的小者，输入3，则求a与b之和。

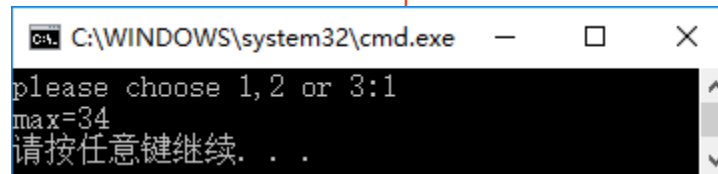
```
#include <stdio.h>
int main()
{
    int fun(int x,int y, int (*p)(int,int)); //fun函数声明
    int max(int,int);                       //max函数声明
    int min(int,int);                       //min函数声明
    int add(int,int);                      //add函数声明
    int a=34,b=-21,n;
    printf("please choose 1,2 or 3:");
    scanf("%d",&n);                       //输入1,2或3之一
    if(n==1) fun(a,b,max);                 //输入1时调用max函数
    else if(n==2) fun(a,b,min);            //输入2时调用min函数
    else if(n==3) fun(a,b,add);            //输入3时调用add函数
    return 0;
}

int fun(int x,int y,int (*p)(int,int)) //定义fun函数
{
    int result;
    result=(*p)(x,y);
    printf("%d\n",result);               //输出结果
}
```

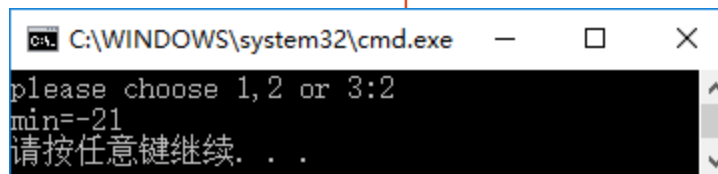
```
int max(int x,int y) //定义max函数
{
    int z;
    if(x>y) z=x;
    else z=y;
    printf("max=");
    return(z); //返回值是两数中的大者
}
```

```
int min(int x,int y) //定义min函数
{
    int z;
    if(x<y) z=x;
    else z=y;
    printf("min=");
    return(z); //返回值是两数中的小者
}
```

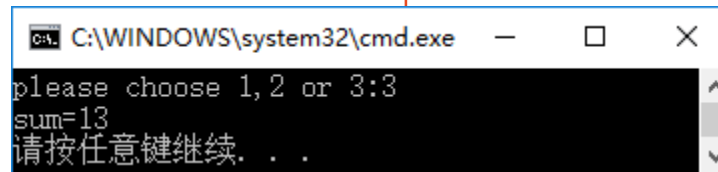
```
int add(int x,int y) //定义add函数
{
    int z;
    z=x+y;
    printf("sum=");
    return(z); //返回值是两数之和
}
```



```
C:\WINDOWS\system32\cmd.exe
please choose 1,2 or 3:1
max=34
请按任意键继续. . .
```



```
C:\WINDOWS\system32\cmd.exe
please choose 1,2 or 3:2
min=-21
请按任意键继续. . .
```



```
C:\WINDOWS\system32\cmd.exe
please choose 1,2 or 3:3
sum=13
请按任意键继续. . .
```

***返回指针的函数**

返回指针值的函数

类型名 *函数名(参数表列)

一个函数可以返回一个整型值、字符值、实型值等，也可以返回指针型的数据，即地址。其概念与以前类似，只是返回的值的类型是指针类型而已。

```
int *a(int x,int y);
```

a是函数名，调用它以后能得到一个int*型(指向整型数据)的指针，即整型数据的地址。x和y是函数a的形参，为整型。

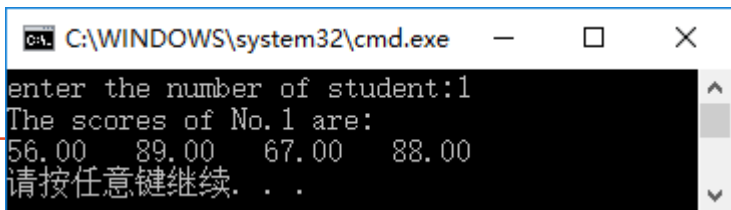
注意

在“*a”两侧没有括号，在a的两侧分别为*运算符和()运算符。而()优先级高于*，因此a先与()结合，显然这是函数形式。这个函数前面有一个*，表示此函数是指针型函数（函数值是指针）。最前面的int表示返回的指针指向整型变量。

返回指针值的函数

【例8.25】有a个学生，每个学生有b门课程的成绩。要求在用户输入学生序号以后，能输出该学生的全部成绩。用指针函数来实现。

```
#include <stdio.h>
int main()
{
    float score[][4]={{60,70,80,90},{56,89,67,88},{34,78,90,66}};
    //定义数组，存放成绩
    float *search(float (*pointer)[4],int n); //函数声明
    float *p;
    int i,k;
    printf("enter the number of student:");
    scanf("%d",&k);    //输入要找的学生的序号
    printf("The scores of No.%d are:\n",k);
    p=search(score,k); //调用search函数，返回score[k][0]的地址
    for(i=0;i<4;i++)
        printf("%5.2ft",*(p+i)); //输出score[k][0]~score[k][3]的值
    printf("\n");
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
enter the number of student:1
The scores of No.1 are:
56.00  89.00  67.00  88.00
请按任意键继续...
```

```
float *search(float (*pointer)[4],int n)
//形参pointer是指向一维数组的指针变量
{
    float *pt;
    pt=*(pointer+n); //pt的值是&score[k][0]
    return(pt);
}
```

pointer	score数组			
pointer	60	70	80	90
pointer+1	56	89	67	88
	34	78	90	66

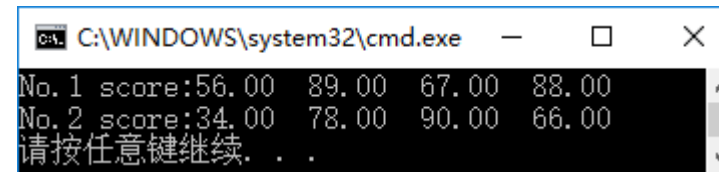
返回指针值的函数

【例8.26】对例8.25中的学生，找出其中有不及格的课程的学生及其学生号。

```
#include <stdio.h>
int main()
{
    float score[][4]={{60,70,80,90},{56,89,67,88},{34,78,90,66}};
    //定义数组，存放成绩
    float *search(float (*pointer)[4]); //函数声明
    float *p;
    int i,j;
    for(i=0;i<3;i++)                //循环3次
    {
        p=search(score+i);
        //调用search函数,如有不及格返回score[i][0]的地址,否则返回NULL
        if(p==*(score+i))
            //如果返回的是score[i][0]的地址，表示p的值不是NULL
        {
            printf("No.%d score:",i);
            for(j=0;j<4;j++)
                printf("%5.2f ",*(p+j));
            //输出score[i][0]~score[i][3]的值
        }
    }
}
```

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

float *search(float (*pointer)[4])
//定义函数，形参pointer是指向一维数组的指针变量
{
    int i=0;
    float *pt;
    pt=NULL;    //先使pt的值为NULL
    for(;i<4;i++)
        if(*(score+i)<60) pt=score+i;
        //如果有不及格课程，使pt指向score[i][0]
    return(pt);
}
```



```
C:\WINDOWS\system32\cmd.exe
No. 1 score:56.00 89.00 67.00 88.00
No. 2 score:34.00 78.00 90.00 66.00
请按任意键继续. . .
```

pointer	60	70	80	90
score+2	56	89	67	88
pointer	34	78	90	66

pt(当有不及格时)	pt (当无不及格时)
*pointer	NULL
↓	
56 89 67 88	

***指针数组和多重指针**

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];

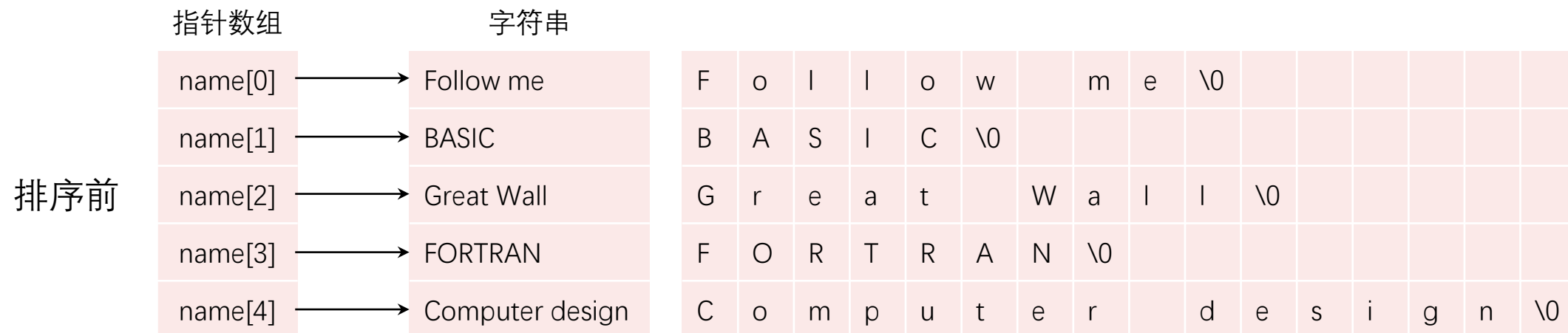
```
int *p[4];
```

一个数组，若其元素均为指针类型数据，称为**指针数组**，也就是说，指针数组中的每一个元素都存放一个地址，相当于一个指针变量。

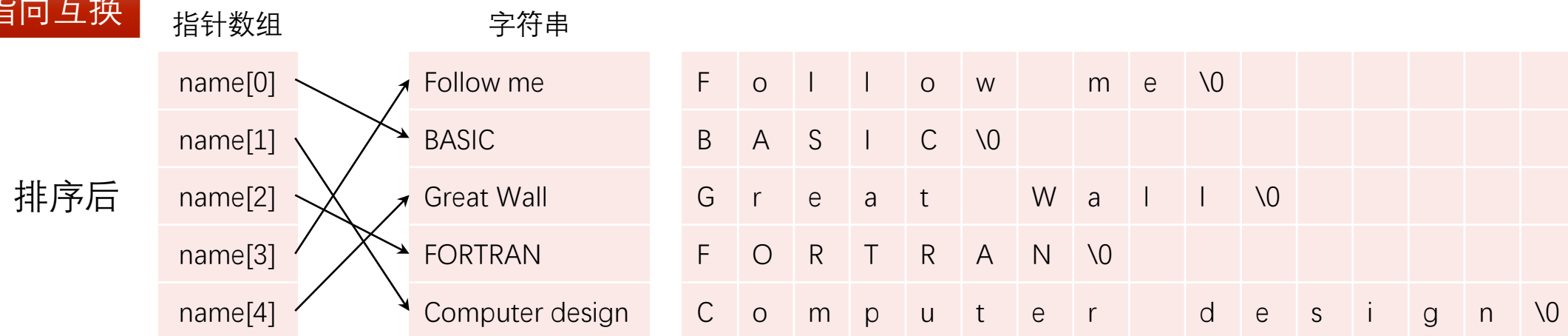
指针数组比较适合用来指向若干个字符串，使字符串处理更加方便灵活。

什么是指针数组

【例8.27】将若干字符串按字母顺序（由小到大）输出。



指向互换



什么是指针数组

【例8.27】将若干字符串按字母顺序（由小到大）输出。

```
C:\WINDOWS\system32\cmd.exe
BASIC
Computer design
FORTRAN
Follow me
Great Wall
请按任意键继续. . .
```

```
#include <stdio.h>
#include <string.h>
int main()
{
    void sort(char *name[],int n);    //函数声明
    void print(char *name[],int n);    //函数声明
    char *name[]={"Follow me","BASIC",
    "Great Wall","FORTRAN","Computer design"};
    //定义指针数组，它的元素分别指向5个字符串
    int n=5;
    sort(name,n); //调用sort函数，对字符串排序
    print(name,n); //调用print函数，输出字符串
    return 0;
}

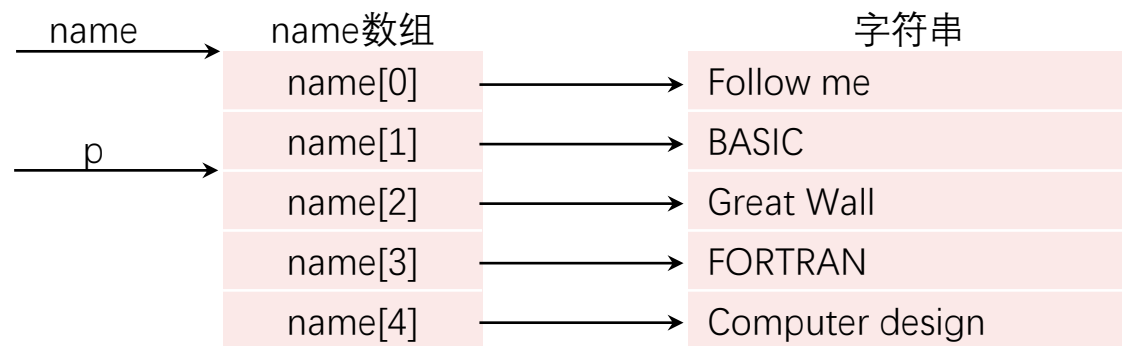
void sort(char *name[],int n)    //定义sort函数
{
    char *temp;
```

```
    int i,j,k;
    for(i=0;i<n-1;i++)    //用选择法排序
    {
        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 print(char *name[],int n)    //定义print函数
    {
        int i;
        for(i=0;i<n;i++)
            printf("%s\n",name[i]);
        //按指针数组元素的顺序输出它们所指向的字符串
    }
```

指向指针数据的指针变量

在了解了指针数组的基础上，需要了解**指向指针数据的指针变量**，简称为**指向指针的指针**。



name是一个指针数组，它的每一个元素是一个指针型的变量，其值为地址。name既然是一个数组，它的每一元素都应有相应的地址。数组名name代表该指针数组首元素的地址。name+i是name[i]的地址。name+i就是指向指针型数据的指针。还可以设置一个指针变量p，它指向指针数组的元素。p就是指向指针型数据的指针变量。

定义一个指向指针数据的指针变量：

```
char **p;
```

p的前面有两个*号。p指向一个字符指针变量（这个字符指针变量指向一个字符型数据）。如果引用*p，就得到p所指向的字符指针变量的值。

```
p=name+2;
```

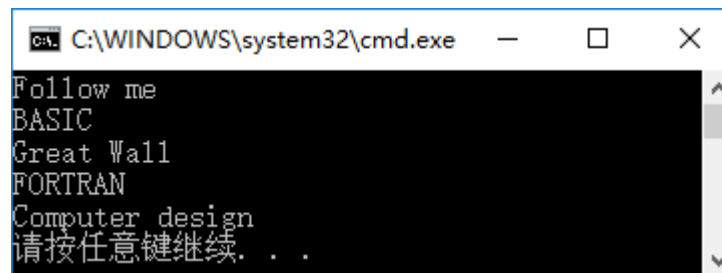
```
printf("%d\n",*p);    //name[2]的值（它是一个地址）
```

```
printf("%s\n",*p);    //以字符串形式(%s)输出字符串"Great Wall"
```


指向指针数据的指针变量

【例8.28】 使用指向指针数据的指针变量。

```
#include <stdio.h>
int main()
{
    char *name[]={"Follow me","BASIC","Great Wall","FORTRAN","Computer design"};
    char **p;
    int i;
    for(i=0;i<5;i++)
    {
        p=name+i;
        printf("%s\n",*p);
    }
    return 0;
}
```



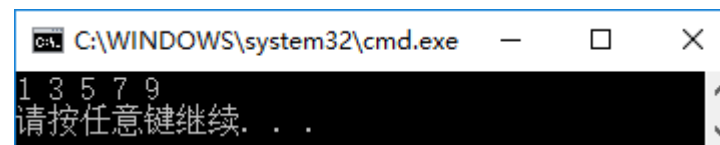
p是指向char*型数据的指针变量，即指向指针的指针。在第1次执行for循环体时，赋值语句“p=name+i;”使p指向name数组的0号元素name [0]，*p是name [0] 的值，即第1个字符串首字符的地址，用printf函数输出第1个字符串（格式符为%s）。执行5次循环体，依次输出5个字符串。

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。

指向指针数据的指针变量

【例8.29】有一个指针数组，其元素分别指向一个整型数组的元素，用指向指针数据的指针变量，输出整型数组各元素的值。

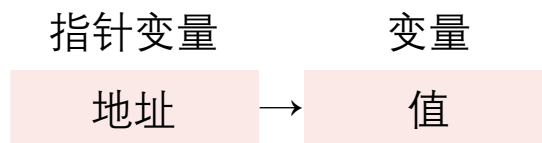
```
#include <stdio.h>
int main()
{
    int a[5]={1,3,5,7,9};
    int *num[5]={&a[0],&a[1],&a[2],&a[3],&a[4]};
    int **p,i;           //p是指向指针型数据的指针变量
    p=num;               //使p指向num[0]
    for(i=0;i<5;i++)
    {
        printf("%d ",**p);
        p++;
    }
    printf("\n");
    return 0;
}
```



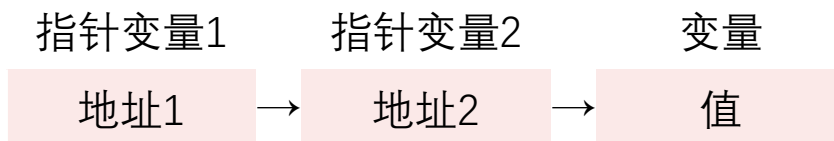
指向指针数据的指针变量

利用指针变量访问另一个变量就是“间接访问”。

如果在一个指针变量中存放一个目标变量的地址，这就是“单级间址”；



指向指针数据的指针用的是“二级间址”方法；



从理论上说，间址方法可以延伸到更多的级，即多重指针。



指针数组作main函数的形参

指针数组的一个重要应用是作为main函数的形参。在以往的程序中，main函数的第1行一般写成以下形式：`int main()` 或 `int main(void)`

括号中是空的或有“void”，表示main函数没有参数，调用main函数时不必给出实参。

在某些情况下，main函数可以有参数，即：`int main(int argc, char *argv[])`

其中，argc和argv就是main函数的形参，它们是程序的“命令行参数”。argc(argument count的缩写，意思是参数个数)，argv(argument vector缩写，意思是参数向量)，它是一个*char指针数组，数组中每一个元素(其值为指针)指向命令行中的一个字符串的首字符。

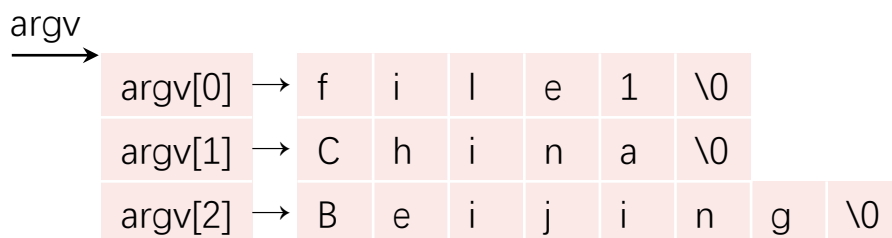
注意

如果用带参数的main函数，其第一个形参必须是int型，用来接收形参个数（文件名也算一个参数），第二个形参必须是字符指针数组，用来接收从操作系统命令行传来的字符串中首字符的地址。

main函数是操作系统调用的，实参只能由操作系统给出。在操作命令状态下，实参是和执行文件的命令一起给出的。

命令名 参数1 参数2...参数n

file1 China Beijing



指针数组作main函数的形参

```
int main(int argc, char *argv[])
{
    while(argc > 1)
    {
        ++argv;
        printf("%s\n", *argv);
        --argc;
    }
    return 0;
}
```



```
int main(int argc, char *argv[])
{
    while(argc-- > 1)
        printf("%s\n", *++argv);
    return 0;
}
```

```
命令提示符
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\jin>cd desktop

C:\Users\jin\Desktop>file1 China Beijing
China
Beijing

C:\Users\jin\Desktop>
```

***动态内存分配与指向它的指针变量**



什么是内存的动态分配

全局变量是分配在内存中的静态存储区的，非静态的局部变量(包括形参)是分配在内存中的动态存储区的，这个存储区是一个称为**栈**(stack)的区域。除此以外，C语言还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**(heap)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。



怎样建立内存的动态分配

用malloc函数开辟动态存储区

函数原型为

void *malloc(unsigned int size);

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。形参size的类型定为无符号整型(不允许为负数)。此函数的值（即“返回值”）是所分配区域的第一个字节的地址，或者说，此函数是一个指针型函数，返回的指针指向该分配域的第一个字节。

```
malloc(100);    //开辟100字节的临时分配域，函数值为其第1个字节的地址
```

指针的基类型为void，即不指向任何类型的数据，只提供一个纯地址。如果此函数未能成功地执行(例如内存空间不足)，则返回空指针(NULL)。

怎样建立内存的动态分配

用calloc函数开辟动态存储区

函数原型为

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

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**。

用calloc函数可以为二维数组开辟动态存储空间，n为数组元素个数，每个元素长度为size。这就是动态数组。函数返回指向所分配域的第一个字节的指针；如果分配不成功，返回NULL。

```
p=calloc(50,4);      //开辟50×4个字节的临时分配域，把首地址赋给指针变量p
```

怎样建立内存的动态分配

用realloc函数重新分配动态存储区

函数原型为

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

如果已经通过malloc函数或calloc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。如果重分配不成功，返回NULL。

```
realloc(p,50);    //将p所指向的已分配的动态空间改为50字节
```

怎样建立内存的动态分配

用free函数释放动态存储区

函数原型为

void free(void *p);

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应是最近一次调用calloc或malloc函数时得到的函数返回值。

```
free(p);    //释放指针变量p所指向的已分配的动态空间
```

free函数无返回值。

以上4个函数的声明在stdlib.h头文件中，在用到这些函数时应当用“#include <stdlib.h>”指令把stdlib.h头文件包含到程序文件中。

void指针类型

C 99允许使用基类型为void的指针类型。可以定义一个基类型为void的指针变量(即void*型变量)，它不指向任何类型的数据。在将它的值赋给另一指针变量时由系统对它进行类型转换，使之适合于被赋值的变量的类型。

注意

不要把“指向void类型”理解为能指向“任何的类型”的数据，而应理解为“指向空类型”或“不指向确定的类型”的数据。

```
int *pt;
```

```
pt=(int *)malloc(100);
```

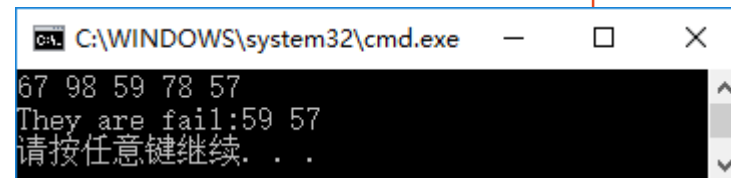
//malloc(100)是void *型，把它转换为int *型

void指针类型

【例8.30】建立动态数组，输入5个学生的成绩，另外用一个函数检查其中有无低于60分的，输出不合格的成绩。

```
#include <stdio.h>
#include <stdlib.h> //程序中用了malloc函数，应包含stdlib.h
int main()
{
    void check(int *); //函数声明
    int *p1,i; //p1是int型指针
    p1=(int *)malloc(5*sizeof(int)); //开辟动态内存区，将地址转换成int *型，然后放在p1中
    for(i=0;i<5;i++)
        scanf("%d",p1+i); //输入5个学生的成绩
    check(p1); //调用check函数
    return 0;
}

void check(int *p) //定义check函数，形参是int*指针
{
    int i;
    printf("They are fail:");
    for(i=0;i<5;i++)
        if(p[i]<60) printf("%d ",p[i]); //输出不合格的成绩
    printf("\n");
}
```



```
C:\WINDOWS\system32\cmd.exe
67 98 59 78 57
They are fail:59 57
请按任意键继续. . .
```

» 有关指针的小结

(1) 首先要准确理解指针的含义。“指针”是C语言中一个形象化的名词，形象地表示“指向”的关系，其在物理上的实现是通过地址来完成的。

- `&a`是变量`a`的地址，也可称为变量`a`的指针。
- 指针变量是存放地址的变量，也可以说，指针变量是存放指针的变量。
- 指针变量的值是一个地址，也可以说，指针变量的值是一个指针。
- 指针变量也可称为地址变量，它的值是地址。
- `&`是取地址运算符，`&a`是`a`的地址，也可以说，`&`是取指针运算符。`&a`是变量`a`的指针（即指向变量`a`的指针）。
- 数组名是一个地址，是数组首元素的地址，也可以说，数组名是一个指针，是数组首元素的指针。
- 函数名是一个指针(指向函数代码区的首字节)，也可以说函数名是一个地址(函数代码区首字节的地址)。
- 函数的实参如果是数组名，传递给形参的是一个地址，也可以说，传递给形参的是一个指针。

» 有关指针的小结

(2) 一个地址型的数据实际上包含3个信息：

- ① 表示内存编号的纯地址。
- ② 它本身的类型，即指针类型。
- ③ 以它为标识的存储单元中存放的是什么类型的数据，即基类型。

```
int a;
```

```
/* &a为a的地址，它就包括以上3个信息，它代表的是一个整型数据的地址，int是&a的基类型(即它指向的是int型的存储单元)。&a就是“指向整型数据的指针类型”或“基类型为整型的指针类型”，其类型可以表示为“int *”型。*/
```

» 有关指针的小结

(3) 要区别指针和指针变量。指针就是地址，而指针变量是用来存放地址的变量。

(4) 什么叫“指向”？地址就意味着指向，因为通过地址能找到具有该地址的对象。对于指针变量来说，把谁的地址存放在指针变量中，就说此指针变量指向谁。

注意

并不是任何类型数据的地址都可以存放在同一个指针变量中的，只有与指针变量的基类型相同的数据的地址才能存放在相应的指针变量中。

```
int a,*p;    //p是int*型的指针变量，基类型是int型
float b;
p=&a;        //a是int型，合法
p=&b;        //b是float型，类型不匹配
```

void *指针是一种特殊的指针，不指向任何类型的数据。如果需要用此地址指向某类型的数据，应先对地址进行类型转换。

» 有关指针的小结

(5) 要深入掌握在对数组的操作中正确地使用指针，搞清楚指针的指向。

```
int *p, a[10];    //p是指向int型类型的指针变量  
p=a;             //p指向a数组的首元素
```

» 有关指针的小结

(6) 有关指针变量的归纳比较

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
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

» 有关指针的小结

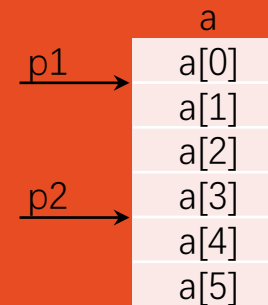
(7) 指针运算

- ① 指针变量加（减）一个整数。

```
p++; //将该指针变量的原值(是一个地址)和它指向的变量所占用的存储单元的字节数相加
```

- ② 指针变量赋值。将一个变量地址赋给一个指针变量。 不应把一个整数赋给指针变量。

```
p=&a; //将变量a的地址赋给p
p=array; //将数组array首元素地址赋给p
p=&array[i]; //将数组array第i个元素的地址赋给p
p=max; //max为已定义的函数，将max的入口地址赋给p
p1=p2; //p1和p2是基类型相同指针变量，将p2的值赋给p1
```



- ③ 两个指针变量可以相减。如果两个指针变量都指向同一个数组中的元素，则两个指针变量值之差是两个指针之间的元素个数。
- ④ 两个指针变量比较。若两个指针指向同一个数组的元素，则可以进行比较。指向前面的元素的指针变量“小于”指向后面元素的指针变量。如果p1和p2不指向同一数组则比较无意义。

» 有关指针的小结

(8) 指针变量可以有空值，即该指针变量不指向任何变量。

```
p=NULL;
```

NULL是一个符号常量，代表整数0。在stdio.h头文件中对NULL进行了定义：`#define NULL 0`。它使p指向地址为0的单元。系统保证使该单元不作它用（不存放有效数据）。

注意

p的值为NULL与未对p赋值是两个不同的概念。前者是有值的（值为0），不指向任何变量，后者虽未对p赋值但并不等于p无值，只是它的值是一个无法预料的值，也就是p可能指向一个事先未指定的单元。

任何指针变量或地址都可以与NULL作相等或不相等的比较。

```
if(p==NULL)
```

» 有关指针的小结

指针的优点：

- ① 提高程序效率；
- ② 在调用函数时当指针指向的变量的值改变时，这些值能够为主调函数使用，即可以从函数调用得到多个可改变的值；
- ③ 可以实现动态存储分配。

如果使用指针不当，会出现隐蔽的、难以发现和排除的故障。因此，使用指针要十分小心谨慎。