第十章 指针

郎大鹏

第十章 指针

先看一个例子

```
#include <stdio.h>
void exchange1(int x, int y)
{
    int temp;
    temp=x;
    x=y;
    y=temp;
}
void main()
{
    int a=18,b=30;
    exchange1(a,b);
    printf("a=%d,b=%d\n",a,b);
}
```

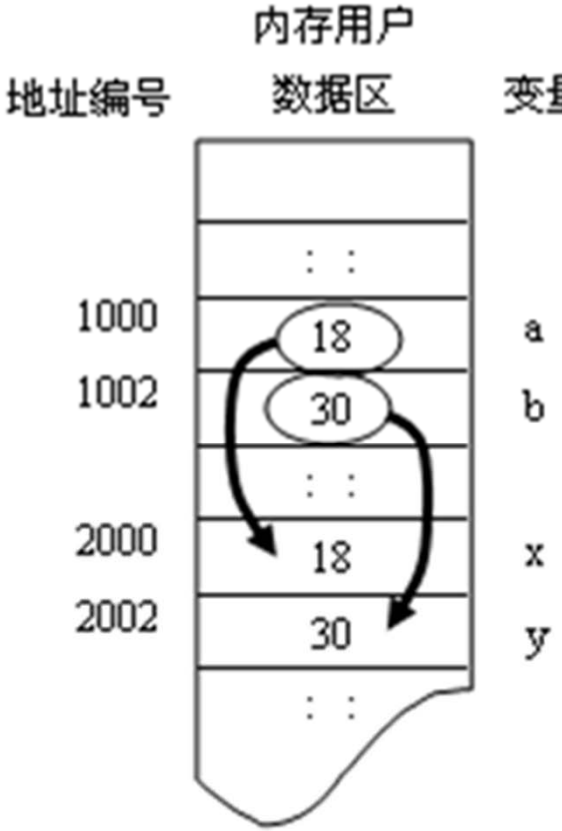


图 10.1　参数传递过程

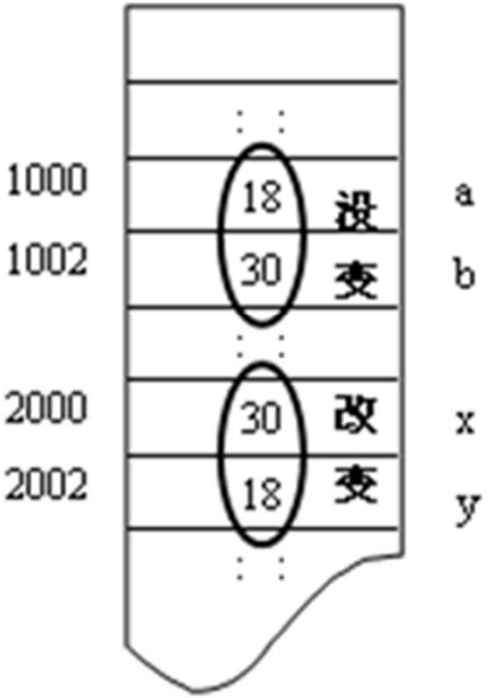


图 10.2　函数处理结果

10.1 内存数据的访问方式

访问内存中的数据的方式：

- ✓ **直接访问方式**

  通常，在程序中通过变量名来对存储单元进行数据的存取操作，这是“直接访问”方式。如：printf("a=%d,b=%d\n",a,b);

- ✓ **间接访问方式**

  假设定义了变量x、y，且已存放了变量a、b的地址（变量地址指该变量所占空间的首单元地址），要取出a变量的值，可以先找到变量x，从中取出a的地址1000，然后再到1000、1001单元取出a的值18。这种访问数据的方式为“间接访问”方式。

- ✓ **指针和指针变量**

  显然，可以通过地址找到相应的存储单元，访问其中的数据，在C语言中，把内存地址形象地称为“指针”，指针就是地址，地址就是指针。

10.2　指针变量的定义及其基本使用

10.2.1　指针变量的定义

定义指针变量的一般形式为：

基类型　* 指针变量名1，* 指针变量名2，…… ；

例如：

```
int  a，b，*p1;    /* p1是指向int类型变量的指针变量*/
char  c，*p2;      /* p2为指向char类型变量的指针变量*/
float  d，*p3;     /*p3为指向float类型变量的指针变量*/
```

把a的地址存放在变量p1中，方法是：

```
p1=&a;
```

类似地，使得p2指向c、p3指向d的语句是：

```
p2=&c;
p3=&d;
```

10.2.1 指针变量的定义

- ✓ 必须区分开指针变量、指针所指变量；如p1=&a；中，p1是指针变量，a是指针p1所指变量；
- ✓ 指针变量的基类型限定了该指针指向的变量的类型。所以，下面的语句是错误的：

```
p1=&c;  /*p1的基类型是int，而c是char类型，类型不符*/
p2=&d;  /*p2的基类型是char，而d是float类型，类型不符*/
p3=&b;  /*p3的基类型是float，而b是int类型，类型不符*/
```

- ✓ C语言规定，指针变量只能存放本程序中已分配内存空间的地址.

```
p1=2008;  /*2008是一个整型数据，不能作为地址使用*/
p2=0xfff4;  /*0xfff4是一个十六进制数形式的整型数据，同样也不能作地址用*/
```

10.2.1　指针变量的定义

- ✓ 指针变量也是变量，它的值也可以改变，即可以随时改变它的指向。

如：

```
p1=&a;     /*p1指向a*/
p1=&b;     /*此时p1指向了变量b*/
int *p4=&a;
     /*再定义一个指针变量p4，并对其进行初始化，使其指向a */
p1=p4;
     /*p4中的地址赋值给了p1，意味着此时指针p1与p4指向了同一个
变量a */
```

10.2.1　指针变量的定义

- ✓ 指针变量也是变量，它的值也可以改变，即可以随时改变它的指向。

如：

```
p1=&a;     /*p1指向a*/
p1=&b;     /*此时p1指向了变量b*/
int *p4=&a;
     /*再定义一个指针变量p4，并对其进行初始化，使其指向a */
p1=p4;
     /*p4中的地址赋值给了p1，意味着此时指针p1与p4指向了同一个
变量a */
```

10.2.2　指针变量的基本使用方法

（1）取地址运算符“&”

“&”是一个单目运算符，其功能是返回其运算量的内存地址。它可以作用于普通变量，如前面提到的a、b、c等，也可以作用于数组元素、结构体变量。

（2）指针运算符“*”

对指针应用指针运算符“*”，可以访问到指针所指向的变量。例如，指针p已经指向了a变量，则对指针p应用“*”运算，即“*p”表示的就是p所指向的变量a，可以通过指针p对a进行赋值或引用

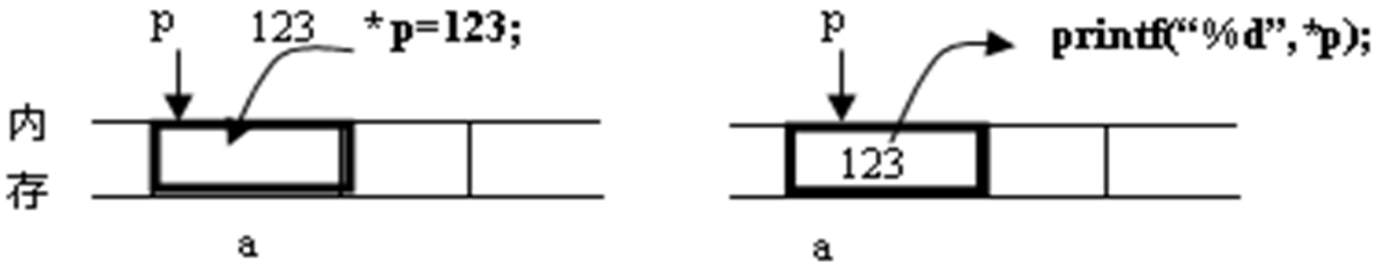
10.2.2　指针变量的基本使用方法

*p=123;

/*给p指向的变量a赋值123，此时a的值变为123 ，此语句等价于a=123;*/

printf(“a=%d\n”,*p);

/*输出p指向的变量a的值，即输出123，等价于printf(“a=%d\n”,a);*/



*p的含义

10.2.2　指针变量的基本使用方法

例10.2　应用指针间接访问变量a、b，使二者的数据互换。

```
#include <stdio.h>
void main()
{int a,b,temp;
 int *x=&a,*y=&b;
 scanf("%d%d",&a,&b);
 temp=*x;
 *x=*y;
 *y=temp;
 printf("a=%d,b=%d\n",a,b);
}
```

10.3 指针作函数的参数

修改**exchang1**函数……

```
void exchange2(int *x, int *y)
{ /* x、y中存放了两个整数（这里是a、b）的内存地址*/
  /*下面按照x、y中的地址找到相应内存单元，互换其中的数据*/
  int temp;
  temp=*x;
  *x=*y;
  *y=temp;
}
```

10.3　指针作函数的参数

例10.3 要求编写函数sort3对任意三个整数进行升序排序，然后在主函数中调用它。

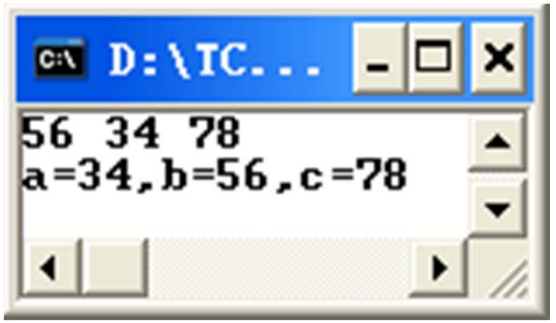
解题思路：

通常我们所说对a、b、c的排序是指，按一定的规则调整、交换变量中的数据，使得a中数据最小，c中数据最大，从而有序。

显然，要对主函数中的a、b、c排序，只能将它们的地址传递给函数sort3，由函数sort3对它们排序。

10.3 指针作函数的参数

```
#include <stdio.h>
void sort3(int *x,int *y,int *z)
{int temp;
 if (*x>*y)
    {temp=*x;*x=*y;*y=temp;}
 if (*y>*z)
    {temp=*y;*y=*z;*z=temp;}
 if (*x>*y)
   {temp=*x;*x=*y;*y=temp;}
}
void main()
{   int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    sort3(&a,&b,&c);
    printf("a=%d,b=%d,c=%d\n",a,b,c);
}
```



10.3　指针作函数的参数

例10.4 编写函数prime_maxmin求任意闭区间[a，b]内的所有素数的个数以及其中的最大素数、最小素数。

解题思路：

（1）要调用该函数必须指出闭区间的上限和下限，形参应包含 int a，int b ；

（2）函数处理的结果有3个：素数个数、最大素数、最小素数。这里我们都采用指针作形参来“返回”结果，因此，函数返回值类型为void。具体实现方法是，在主调函数中定义3个变量count、max、min，然后把它们的地址传递给函数prime_maxmin，函数把计算的结果直接存放在这些地址中。所以，该函数的形参还应包含：int *pcount，int *pmax，int *pmin 。

```
#include <stdio.h>
int prime(int x)  /*判断x是否为素数，是返回1，不是返回0 */
{int i;
 for(i=2;i<x;i++)
   if(x%i==0)return 0;
 return 1;
}
void prime_maxmin(int a,int b,int *pcount,int *pmax,int *pmin)
{int k;
for(k=a;k<=b;k++)   /*穷举[a，b]间的所有整数 */
   if(prime(k))   /*判断k是否为素数*/
   {*pcount=*pcount+1;
                /*若是素数，pcount指向的变量中的数据增1*/
*pmax=k;  /*素数放在pmax指向的变量中*/
if(*pcount==1)  *pmin=k;
                /*如果是第1个素数，一定是最小素数*/
   }
}
```

```
void main()
{int a1,b1,count,max,min;
 count=0;
 scanf("%d%d",&a1,&b1);
/*输入任意闭区间的下限a1和上限b1*/
 prime_maxmin(a1,b1,&count,&max,&min);
 /*调用函数*/
 printf("count=%d,max=%d,min=%d\n",count,max,min);
}
```

10.4 指针变量的各种应用

在C语言中，数组与指针密不可分。首先来观察几个数组：

char a[5]={ 'c', 'h', 'i', 'n', 'a'};

int b[5]={2, 6, 10, 7, 9 };

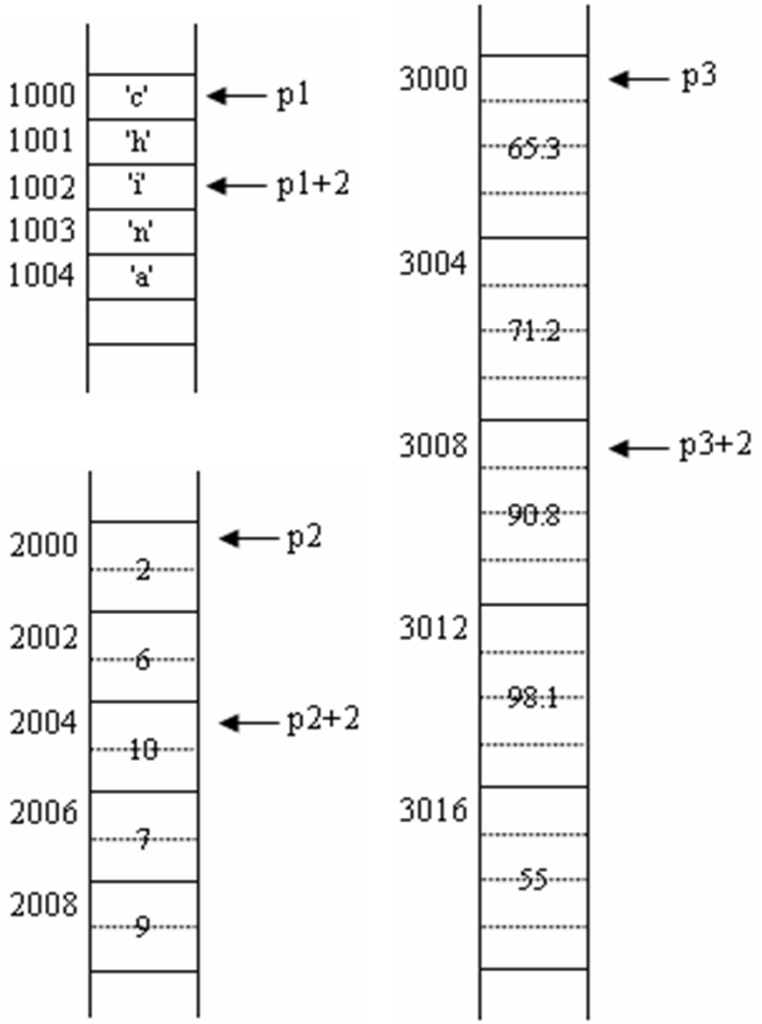
float c[5]={65.3, 71.2, 90.8,98.1, 55};

定义了3个指针并初始化：

char *p1=&a[0];

int *p2=&b[0];

float *p3=&c[0];



10.4 指针变量的各种应用

（1）指针的加、减运算规则;

（2）访问数组元素的几种方法

① 使用 *(p1+2) 访问a[2]。

```
*(p1+2)=30;  /*  首先计算出p1+2为a[2]的地址, 再对其进行指针运算“*”。*/
```

② 使用 *(a+2) 访问a[2]。

```
*(a+2)=30;
```

③ 使用p1[2]访问a[2]。

```
p1[2]=30;
```

既然 *(a+i) 等效于a[i]，那么形式相同的*(p1+i) 也可以认为等效于p1[i]，所以*(p1+2)等价于p1[2]。

④ 使指针p2不断后移，直到指向a[2]，然后给它赋值。

```
p2++;  /* p2指向了a[1] */
p2++;  /* p2指向了a[2] */
*p2=30;  /*对此时p2所指向的元素a[2]赋值*/
```

例10.5 输出数组的每个元素。

```
#include <stdio.h>
void main()
{int i,a[5],*p=a;
for(i=0;i<5;i++)  /*为数组输入数据*/
   scanf("%d",&a[i]);
 for(i=0;i<5;i++)  /*方法1：采用下标法输出各元素*/
   printf("a[%d]=%d ",i,a[i]);   /*这里的a[i]也可以写成*(a+i)，二者等价*/
 printf("\n");
for(i=0;i<5;i++)  /*方法2：使用指针法输出各元素*/
   printf("*(p+%d)=%d ", i ,*(p+i));  /*可以把*(p+i)替换为p[i]，二者等价*/
 printf("\n");
 for(p=a;p<(a+5);p++)  / *方法3：使用指针法访问各元素*/
   printf("*p=%d ",*p);  /*指针不断后移，依次输出当前指针指向的元素*/
 printf("\n");
}
```

10.4.2　指针与字符串

（1）用字符数组存放一个字符串，然后通过指针访问它。

观察下面的定义：

```
char a[10]={'c','h','i','n','a','\0'};
/*也可以写成char a[10]="china";  */
char *p=a;
```

a数组中存放了字符串"china"，指针p初始指向第一个字符'c'。常常说p指向了字符串"china"。

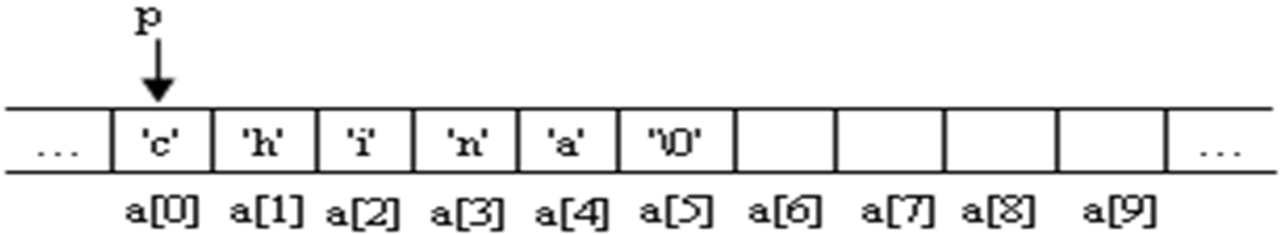


图　指针p指向字符串

10.4.2　指针与字符串

例10.6 求字符串的长度。

```
#include <stdio.h>
void main()
{
char a[30],*p;
int n=0;
 scanf("%s",a);
            /*输入字符串存放在a数组中*/
 for(p=a;*p!='\0';p++)
            n++;     /*计数字符个数*/
 printf("length=%d\n",n);
}
```

改写为

```
#include <stdio.h>
void main()
{char a[30];
 int i,n=0;
 scanf("%s",a);
 for(i=0;*(a+i)!='\0';i++)
            n++;
 printf("length=%d\n",n);
}
```

10.4.2　指针与字符串

（2）定义一个字符指针，使其直接指向指定的字符串。

```
char *p="Just do it!";
```

在编译程序时，系统会在内存中分配连续空间存放程序中出现的所有字符串常量，所以这里的字符串"Just do it!"就会占用一段连续的存储单元，并用其首字符所在单元的地址来初始化字符指针p，使p指向该字符串。

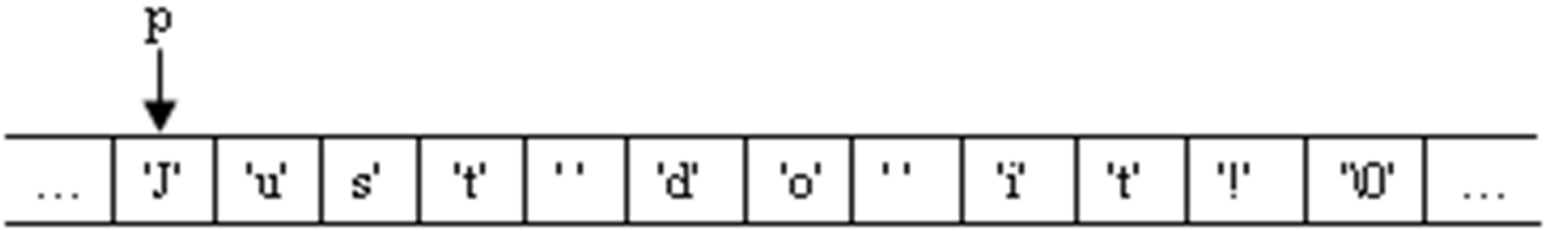


图　指针p指向指定字符串

10.4.2　指针与字符串

例10.7 使用上述指针指向字符串常量的方法求字符串的长度。

```
#include <stdio.h>
void main()
{char *p;
 int n=0;
p="Just do it!";  /*p指向了字符串的首字符*/
 for(  ; *p!='\0' ; p++)
                n++;
 printf("length=%d\n",n);
}
```

10.4.2　指针与字符串

（3）使用指针处理字符串过程中要注意的几点

① 指针法与下标式的互换性

② 注意*(p++)、*p++、*(++p)、*++p的使用

③ 随时注意指针的指向，如果一个指针定义时没有初始化，也没有使用赋值语句使其指向本程序合法数据空间，则该指针变量的值是无法预料的。

④ 注意区分数组名与字符指针在使用上的不同。

10.4.3 指针与结构体

（1）指针指向结构体变量

例如，定义了结构体类型student：

```
struct student
{ long num;    /*学号*/
 char name[20];  /*姓名*/
 float score1;    /*分数1*/
 float score2;    /*分数2*/
 float score3;    /*分数3*/
 float average;    /*平均分*/
 };
```

现在定义一个结构体变量s1，用来存放一名学生的信息：

```
struct  student  s1;
```

再定义一个指向struct student 结构体类型的指针p：

```
struct  student  *p;
```

然后使p指向结构体变量s1：

```
p=&s1;
```

10.4.3　指针与结构体

（2）指针指向结构体数组

如果一个班级有30人，就需要定义一个结构体数组存放全班学生的信息，例如：

```
struct  student  s[30];
```

每个数组元素都是一个结构体数据。可以定义指针p使其指向首元素：

```
struct  student  *p=s;
```

然后利用p++使指针不断后移，依次指向每个元素（每个结构体数据），访问每个学生的数据。具体实例在10.5.3小节和10.6节详述。

10.5 使用指针的算法分析和设计

10.5.1 使用指针处理数组

例10.8 对任意n(n<=20)个整数构成的序列，整个序列循环右移m位（m<n），末尾数据移动到序列的开始处。

例如，有10个数据：

1，2，3，4，5，6，7，8，9，10

将其循环右移3位之后，序列变为：

8，9，10，1，2，3，4，5，6，7

10.5.1　使用指针处理数组

10.5.1　使用指针处理数组

例10.8　对任意n(n<=20)个整数构成的序列，整个序列循环右移m位（m<n），末尾数据移动到序列的开始处。

例如，有10个数据：

1，2，3，4，5，6，7，8，9，10

将其循环右移3位之后，序列变为：

8，9，10，1，2，3，4，5，6，7

编写程序：

```
#include  <stdio.h>
void main()
{int a[20],n,m;
 int i,temp;
 int *p;
 printf("n,m=");
 scanf("%d %d",&n,&m);    /*输入数据个数n和右移位数m*/
 for(i=0;i<n;i++)     /*输入n个数据*/
   scanf("%d",&a[i]);
 for(i=1;i<=m;i++)  /*循环m次，实现循环右移m次（位）*/
 {p=a+n-1;  /*初始p指向末尾数据*/
  temp=*p;  /*暂存末尾数据*/
  while(p>a)  /*指针还没指向首元素a[0]*/
  {*p=*(p-1);  p--; }    /*数据后移，指针前移*/
  *p=temp;    /*末尾数据移动到序列首部*/
  }
 printf("\n");
 for(i=0;i<n;i++)    /*输出移位后的数据序列*/
   printf("%d ",a[i]);
}
```

10.5.1　使用指针处理数组

例10.9　一组n个整数中找出最小数与第1个数对调，找出最大数与最后数对调。

分析问题：

假设有5个数据存放在数组a中，如图1所示。最小元素为9，将其与首元素a[0]交换；最大元素为45，把它与最后元素a[4]交换，结果如图2所示

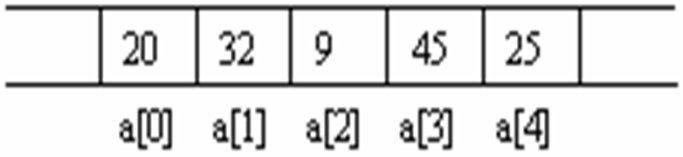


图1

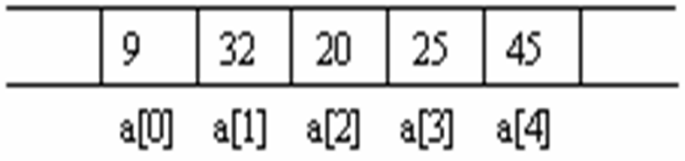


图2

```
#include<stdio.h>
#define n 5
void main()
{int a[n];
 int *p,*pmax,*pmin,t;
  printf("please enter %d integers:\n",n);
  for(p=a;p<a+n;p++)  /*输入n个数据*/
              scanf("%d",p);   /*用户输入数据存放在指针p所指向的元素中*/
  pmin=pmax=a;    /*假设首元素最大、最小*/
  for(p=a+1;p<a+n;p++)   /*指针p不断指向a[1]、a[2]……，依次比较找最大最小*/
              if(*p>*pmax)     /*当前最大元素与p指向元素比较*/
                pmax=p;
              else   if(*p<*pmin) pmin=p;
  t=*a;*a=*pmin;*pmin=t;    /*pmin所指最小元素与a[0]交换*/
  t=*(a+n-1);*(a+n-1)=*pmax;*pmax=t;  /*pmax所指最大元素与a[n-1]交换*/
  printf("The changed array is:\n");
  for(p=a;p<a+n;p++)/*输出交换后数组*/
              printf("%3d",*p);
}
```

10.5.2　使用指针处理字符串

例10.10　字符串逆置。输入一个字符串（最长不超过20个字符），如"Monday"，调整其中各字符的顺序，使之反序放置并输出它，结果为："yadnoM"。

分析问题：

针对要处理的长度不超过20个字符的字符串，需要使用一个字符数组来存放它，即：char s[21];

问题要求把该数组中构成字符串的所有字符调整存放的顺序，使之与原有顺序相反，即逆序存放。注意，并不是把反序的字符串存放在另一个字符数组中。

10.5.1　使用指针处理数组

编写程序：

```
#include <stdio.h>
void main()
{char s[21],*p,*q,ctemp;
 int n;
 scanf("%s",s);
 n=strlen(s);
 for(p=s,q=s+n-1;p<q;p++,q--)
   {ctemp=*p;*p=*q;*q=ctemp;}
 printf("%s\n",s);
}
```

10.5.2　使用指针处理字符串

例10.11 将任意一个十进制数a（$0 \le a \le 2^{32}-1$）转换为十六进制。如输入**2809**，输出相应的十六进制数**AF9** 。

分析问题：

（1）数据表示问题

十进制转换为十六进制，就是由一个整数经过处理得到一个十六进制字符串，如将整数2809进行处理后得到字符串"AF9"。

（2）进制转换问题

按照一定规则（整数a短除16取余）从低位到高位依次求出十六进制数的各位，如对2809，低到高得到字符'9'、'F'、'A'，将其构成字符串"9FA"。显然，正确的结果应该是"AF9"，所以还需要把该字符串逆置。

（3）字符串逆置

把对整数a处理后得到的十六进制字符串"9FA"进行逆序存放，得到"AF9"，然后输出结果。

编写程序：

```
#include <stdio.h>
void main()
{unsigned long a,r;
 char s[9],*p,*q,ctemp;
 int n;
 scanf("%ld",&a);  /*输入整数*/
 p=s;    /*指针p初始指向字符数组首元素*/
 do
 {r=a%16;    /*求余*/
  if(r<10)  *p=r+'0';   /*将余数转换成数字字符*/
  else  *p=r-10+'A';  /*将余数转换成相应字母字符*/
 p++;
  a=a/16;
 }while(a>0);  /*若整数商不为0则继续转换*/
 *p='\0';
 n=strlen(s);  /*求字符串s的长度*/
 for(p=s,q=s+n-1;p<q;p++,q--)
/*对称字符对调*/
 { ctemp=*p;*p=*q;*q=ctemp;}
 printf("%s\n",s);
}
```

10.5.3　使用指针处理结构体数据

例10.13　求两个复数的乘积。

分析问题：

数学上的复数包含两个部分：实部和虚部。两个复数*a*、*b*的乘积如：

$(a1+a2i)\times(b1+b2i)=(a1\times b1-a2\times b2)+(a1\times b2+a2\times b1)\ i$

可以定义一个结构体类型complex，来表示复数类型，同时定义变量a、b，表示两个复数：

```
struct complex
{ int real;
 int imaginary;
}a,b;
```

```
#include <stdio.h>
void main()
{struct complex
 { int real;
   int imaginary;
 }a,b,c;
 struct complex *pa=&a,*pb=&b,*pc=&c;
 printf("input a.real and a.imaginary:");
 scanf("%d%d",&pa->real,&pa->imaginary);
 printf("input b.real and b.imaginary:");
 scanf("%d%d",&pb->real,&pb->imaginary);
 pc->real=pa->real*pb->real-pa->imaginary*pb->imaginary;
 pc->imaginary=pa->real*pb->imaginary+pa->imaginary*pb->real;
printf("(%d+%di)x(%d+%di)=%d+%di\n",pa->real,pa->imaginary,
                pb->real,pb->imaginary,pc->real,pc->imaginary);
 }
```

10.5.3　使用指针处理结构体数据

例10.14　某班期末有4门课程成绩，要求统计每门课程的成绩分布情况，即优秀、中等、不及格的人数，同时输出优秀人数最多的课程名称及优秀人数。

分析问题：

每门课的信息包含：课程名称、优秀人数、中等人数、不及格人数，可以定义一个结构体类型coursegrade，来表示一门课程的成绩分布情况。

```
struct coursegrade
{char coursename[20];  /*课程名称*/
 int excellent;    /*优秀人数*/
 int medium;     /*中等人数*/
 int fail;    /*不及格人数*/
};
```

程序见课本.....

10.5.4　使用指针作参数传递一组数据

例10.15　对指定数组a（数据已经升序排好），给出一个数据key，若数组中存在该数据，则求出其位置（下标），否则将该数据插入数组中使之仍然有序。

分析问题：

假设有数组a，目前存放了n=10个元素，已经有序：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
3	6	8	10	16	20	28	42	50	72

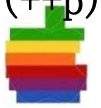
若给出数据key=28，在数组中查找key，a[6]与key相同，返回key出现的位置（下标）为6；

若给出数据key=29，在数组中没有与key相同的元素，此时把key插入该数组中，使之仍然有序。即：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
3	6	8	10	16	20	28	29	42	50	72.

这里涉及两部分功能，一个是查找，一个是插入排序。可以编写两个函数分别实现。

```
#include <stdio.h>
void bio_search(int *a, int n, int key,
int *position)
{int low=0,high=n-1,mid;
 mid=(low+high)/2;
 while(key!=a[mid]&&low<=high)
 {if (key<a[mid]) high=mid-1;
  else low=mid+1;
  mid=(low+high)/2;
 }
 if(low>high) *position=-1;
/*未找到返回-1*/
 else *position=mid;
}
void insert_sort(int *a, int n, int key)
{int *p;
 p=a+n-1;
 while(key<=*p && p>=a)
 {*(p+1)=*p;  /*p所指元素后移一位*/
  p--;
 }
 *(++p)=key; /*把key放入p所指元素后*/
}
```



```
void main()
{int
b[20]={3,6,8,10,16,20,28,42,50,72},x,n=10,
xp,i; /*xp存放x在数组b中位置*/
 printf("b array:\n");
 for(i=0;i<n;i++)
/*输出查找（插入）前的b数组数据*/
   printf("%d\n ",b[i]);
printf("input x:");
 scanf("%d",&x);
/*输入要查找（插入）的数据x*/
 bio_search(b,n,x,&xp);
/*实参b是数组名，即数组b的首地址*/
 if(xp==-1)  /*如果b数组中不存在x，则插入x
，否则输出找到的位置*/
 {insert_sort(b,n,x);
/*实参b是数组名，即数组b的首地址*/
  n++;  /*数组中数据个数增1*/
  for(i=0;i<n;i++)
        printf("%d ",b[i]);
 }
 else printf("found %d at b[%d]!\n",x,xp);
/*输出x所在位置（下标）*/
}
```

10.5.4　使用指针作参数传递一组数据

例10.16　编写函数substr，把给定字符串s1从m开始的全部字符复制成为另一个字符串s2，显然后者是前者的一个子串。

解题思路：

在函数substr中，首先让指针a指向s1串的第m个字符，此时，b已指向了s2字符数组的首元素。判断a指向字符是否到字符串的结尾，如果没有，则将a所指字符复制到b所指元素中，同时a、b指针均后移一个元素，继续判断下一个字符是否到串结尾；否则，a指向s1字符串结束标志时，复制结束。最后将串结束标志’\0’加到当前b所指元素中，构成子串。

编写程序：

```
#include <stdio.h>
void substr(char *a,char *b,int m)
{a=a+m-1;            /* 指向第 m 个字符 */
 while(*a!='\0')     /*a 所指没有到串结束标志*/
    *b++=*a++;       /*复制一个字符，指针后移*/
 *b='\0';
}
void main()
{char s1[30],s2[30];
 int m;
 gets(s1);
 scanf("%d",&m);
 substr(s1,s2,m);
 puts(s2);
}
```

图 10.32　例 10.16 运行结果

10.6　编程举例

例10.17　对一个班级的学生信息进行处理。每个学生的信息包括：学号、姓名、三门课程的分数。要求实现下述功能：

（1）计算并输出每门课程的平均分；

（2）查找3门分数都在90分以上的学生，输出其名单；

（3）将学生信息按照姓名字典顺序排序并输出。

分别编写3个函数来实现各个功能。

解题思路：

这里假设班级中有5位学生，每位学生信息可定义结构体类型来表示：

```
struct student
   {long num;   /*学号*/
    char name[20];  /*姓名*/
    float score1;    /*分数1*/
    float score2;    /*分数2*/
    float score3;    /*分数3*/
    float average;   /*平均分*/     };
```

定义结构体数组stu用于存放全班学生信息，这里同时进行了初始化：

```
const int n=5;  /*定义整型常量n，表示学生人数*/
struct student stu[n]={1001,"zhaoli",89,90,98,0,1002,"lili",90,98,97,0,1003,
"anda",87,89,67,0,1004,"beiping",92,94,95,0,1005,"qiuhui",67,78,76,0};
```

主函数调用这里的每个函数时，都需要传入全班学生的信息（整个结构体数组）和班级人数，即结构体数组的首地址和数组元素个数，因此，每个函数均需要包含两个形参：指向结构体类型的指针、整型变量n。

解题思路：

实现（1）功能的函数可以定义为：

```
void courseaverage(struct student *s,
int n)
{
 …… ……
}
```

实现（2）功能的函数可以定义为：

```
void excellentstu(struct student
*s,int n)
{
 …… ……
}
```

功能（3）要求将结构体数组各元素按照姓名的字典顺序升序排序，函数定义如下：

```
void stusort(struct student *s,int n)
{
 …… ……
}
```

程序见课本……