第五章 指针

【教学目标】

掌握指针的概念；
掌握指针对变量、数组的引用方法；
掌握指针数组、行指针的应用；
了解二级指针的概念。

【教学内容】

变量的指针；
数组的指针；
指针数组、行指针；
指向指针的指针。

【教学重点和难点】

使用指针引用变量、数组；
指针数组、行指针；
数组名是地址常量；
通过对数组概念的理解，灵活利用指针引用各种数组的元素和字符串。

【问题的提出】

当需要引用一个具有很多元素的一个数组的值、或多维字符数组描述的各字符串或元素，能否有更简洁方便的方法；根据解决问题的实际需要，是否能动态的分配所需存储空间而不浪费；计算机所处理的信息都要通过内存进行交换，是否可以通过内存地址直接的对存放于内存的数据进行操作，以便提高处理效率；在计算机中，为了提高存储效率，充分利用有限的内存空间，是否可以将连续的数据分散保存而又能方便地描述出各存储数据的关系等。在 C 语言中，引入了指针概念的变量，可以方便的解决以上问题。

【教学要点】

1. 指针变量的运算

【例 5.1】指针变量的取值运算
利用指针变量，从键盘上输入两个整型值，将其进行交换并输出到屏幕。

```
#include"stdio.h"
main(    )
{ int a,b;
int *pa=&a,*pb=&b; /*  定义指针变量 pa 和 pb，分别指向变量 a 和 b，即 pa 和 pb 分别表示变量 a 和 b 的地址  */
printf("Please input A=");
scanf("%d",pa);          /*        给指针变量 pa 指向的变量 a 赋值                    */
printf("Please input B=");
```

```
    scanf("%d",pb);          /*        给指针变量 pb 指向的变量 b 赋值                    */
    *pa+=*pb;   /*   将地址 pa 和 pb 中的值累加，再存入地址 pa 中，即变量 a 的值是原 a、b
值之和        */
    *pb=*pa-*pb;/*  将地址 pa 中的值减去地址 pb 中的值并存入地址 pb 中，实现 b=(a+b)-b，
即 b 的值是原 a 的值  */
    *pa=*pa-*pb;/*  将地址 pa 中的值减去地址 pb 中的值并存入地址 pa 中，实现 a=(a+b)-a，
即 a 的值是原 b 的值  */
    printf("----------Result -----------\n");
    printf("A=%d,B=%d",*pa,*pb);   /*   输出交换后的变量 a 和 b 的值     */
    getch(    );
    }
```

【例 5.2】指针变量的算术运算-------地址移动或位置差

利用指针变量，将字符串在自身内部倒序交换后再输出。

```
#include"stdio.h"
#include"string.h"
main(    )
   { char a[128],*first,*end,ch; /*  定义数组 a、指针变量 first 和 end 及字符变量 ch      */
      int n,i;
      printf("Please input a character string:\n");
      gets(a);   /*     为数组 a 赋初值，即输入串的字符       */
      n=strlen(a);      /*   测量串 a 的长度       */
      first=a;          /*  为指针变量 first 赋初值为 a，即指针 first 指向字符串首地址     */
      end=&a[n-1]; /*指针变量 end  赋初值为&a[n-1]，即指针 end 指向字符串末尾元素*/
      for( ; first<end ; first++,end--)
   /*  将首、尾元素的值进行交换，再变换首、尾位置，直到首、尾相接时结束交换  */
          {
           ch=*first;
           *first=*end;
           *end=ch;
           }
printf("----------Result ----------\n");
      puts(a);   /*   输出交换后的字符串数组     */
      getch( );
}
```

说明：

测量字符串长度的目的是为确定字符串末尾的位置。

给指针变量 first 赋值字符数组名 a——数组首地址，就是使指针指向数组首地址，以便从开始位置引用数组的元素；给指针变量 end 赋&a[n-1]值，即数组末尾元素的地址，以便从末尾倒序引用数组的元素。

指针变量的算术运算，是移动指针指向的元素对象。

first<end 作为循环条件，即首部往后移、尾部往前移，分别对应的元素值在循环中通过 ch 字符变量作为桥梁进行交换，当首尾重叠时结束交换。

应该注意的是：当 first==end 时，元素个数 n 为奇数时，指针 first 和 end 指向同一个元素；元素个数 n 为偶数时，指针 first 和 end 分别指向前一次对方指向的元素，其值已经被交换了。当循环条件修改为 first<=end，元素个数 n 为奇数时，程序正确，元素个数 n 为偶数时，

程序不正确。

2. 利用指针引用变量

【例 5.3】使用指针间接引用变量的值，实现变量的交换

```
#include"stdio.h"
main( )
{float a=1,b=100,c;
 float *pa,*pb;     /*  定义指针 pa 和 pb          */
 pa=&a;pb=&b;   /*  使指针 pa 和 pb 分别指向变量 a 和 b        */
 clrscr( );
 printf("------Original data------\n");
 printf(" a=%6.2f,b=%6.2f\n\n\n",*pa,*pb);
  /*  输出由指针 pa 和 pb 间接引用的变量 a 和 b 的值  */
 c=*pa;     /*    由指针 pa 间接引用的变量 a 的值，赋给变量 c; 即实现 c=a     */
 *pa=*pb;
  /*  由指针 pb 间接引用的变量 b 的值，赋给由变量 pa 指向的变量 a，即实现 a=b */
 *pb=c;
  /*  直接引用变量 c 的值，赋给指针变量 pb 间接引用的变量 b，即实现 b=c    */
 printf(" ------Result------\n");
 printf("a=%6.2f,b=%6.2f\n",*pa,*pb);
  /*  输出由指针 pa 和 pb 间接引用的、已交换了值的变量 a 和 b 的值  */
 getch( );
 }
```

说明：

使用指针变量时，* 号出现在定义时，是指针类型变量的标志，其*号前有类型标示符，如 int *pa；float *p，char *ch；等；出现在引用指针所指向的变量值时，是取值运算符，前面无类型标示符，如程序中的*pa、*pb。

定义的指针当没有赋初始值，指针处于浮动状态，即没有确定的指向时，不能取指针所指向的对象的值，也不能给指针指向的地址赋值。

3. 利用指针引用数组元素

【例 5.4】利用指针变量找出一维数组中元素最大和最小的值及所在位置。

```
#include"stdio.h"
main( )
{int x[10],i;                /*       定义数组并赋初始值      */
 int *p,max,min,max_seat,min_seat;   /*  定义指针和保存最大最小值及其位置的变量    */
 clrscr( );
 p=x;                        /*    使指针指向数组首地址    */
 for(i=0;i<10;i++)           /*         为数组赋初始值      */
       scanf("%d", x+i);
/*    scanf("%d", p+i);    或  scanf("%d", &x[i]); 不能是 scanf("%d", x++); */
 max=min=x[0];               /*    给最大、最小值赋初始值——数组首元素    */
 max_seat=min_seat=0;    /*    给最大、最小值的位置赋初始值——数组首地址    */
 for(p++;p-x<10;p++)     /*    使数组依次从第二到第十个元素进行处理    */
     {
```

```
        if(*p>max)        /*   查找出最大值，用 max 保存值，其位置存入 max_seat 变量中   */
            {max=*p; max_seat=p-x;}
        if(*p<min)        /*   查找出最小值，用 min 保存值，其位置存入 min_seat 变量中   */
            {min=*p; min_seat=p-x;}
        }
  printf("Max=%d,x[%d]\n",max,max_seat);   /*     输出最大值及位置   */
  printf("Min=%d,x[%d]\n",min,min_seat);   /*     输出最小值及位置   */
  getch( );
  }
```

说明：

数组名即数组首地址，是一个常量，在定义数组时由系统分配，如 p=x。

程序中的 p-x 表示指针 p 移动后与首元素间的距离，即元素个数或者数组的下标位置。

p++，以 int 类型的 2 个字节为一次移动单位，而不是一次移动一个字节。

【例 5.5】利用一级指针处理二维数组，找出一维数组中元素最大和最小的值及所在位置。

```
#include"stdio.h"
#define M 3
#define N 4
main( )
   { float a[M][N],max,min;    /*定义数组及保存最大、最小值的变量     */
     int i,j,max_seat,min_seat;    /*   定义最大最小值位置的变量   */
   float *p;    /*定义指向数组的指针变量   */
   for(i=0;i<M;i++)          /*     执行程序时，从键盘上输入 M*N 个值为数组赋初值     */
        for(j=0;j<N;j++)
             scanf("%f",&a[i][j]);
   p=a[0];          /*     使指针指向数组的首行首地址     */
   clrscr( );
   for(;p-a[0]<M*N;p++)
 /*  从首行首地址开始，依次取出每一个元素，按每行 N 个输出   */
      { if((p-a[0])%N==0)   printf("\n");
        printf("%6.2f",*p);
       }
      puts("\n");
      p=a[0];
   /*由于输出后，指针 P 已经指向数组最后元素之后，重使指针指向数组的首行首地址，
以便对数组进行操作*/
      max_seat=min_seat=0; /*   为最大、最小值位置变量赋初值   */
      max=min=*p;                /*   为最大、最小值变量赋初值   */
      p++;
      for(i=1;i<M*N;i++,p++)
  /*  从第二个元素开始，依次比较每一个元素，找出大、最小值及位置   */
          if(*p>max)
               { max=*p; max_seat=i;}
          else if(*p<min)
               {min=*p;min_seat=i;}
      printf("The Max is %6.2f,seat is A[%d][%d]\n",max,max_seat/N,max_seat%N);
```

```
        /*   位置对行数的整数倍数即行号，位置对行数的余数减 1 即列号   */
    printf("The Min is %6.2f,seat is A[%d][%d]\n",min,min_seat/N,min_seat%N);
    }
```

说明：

多维数组在 c 语言中，各元素按行优先形式顺序保存，即第一行末尾存放第二行的元素值，依次类推。

多维数组是由一维数组叠加组合而成，因此二维数组可以按行拆分成每行一个一维数组，如 a[i]表示二维数组中的一行，相当于一个一维数组名，是该行的首地址，也是常量。

二维整型数组 a[M][N]中的 a[i][j]元素，在内存中离首地址的存储位置字节差计算方法是(i*N+j)*2。

对于二维数组元素 a[i][j]值的引用，可以通过数组名直接实现，如：*(a[i]+j)、*(*(a+i)+j)、*(a+i)[j]都是引用数组元素 a[i][j]值。

4. 利用指针引用字符串

【例 5.6】利用指向字符串的指针变量，连接两个字符串（不使用函数 strcat）

```
#include"stdio.h"
#include"string.h"
main( )
{ char *p1,*p2,*p;      /*  定义指针变量，其中 p 用于记忆串的开始位置     */
   clrscr( );
   p1="1234567890";   /*   给指针变量赋初始值     */
   p2="qwertyu";
   p=p1;                    /*   p 指向 p1 的位置，用于记忆串的开始位置        */
   while(*p1)   p1++;
   /*  通过对串结束标记的检查，使 p1 指向串“1234567890”的结束标记     */
   while(*p2)   *p1++=*p2++;
   /*  当 p2 所指向的串没到结束标记时，依次将其每个元素赋到当前 p1 指向的位置  */
   *p1='\0';                  /*   给 p1 指针所指串添加串结束标记，使字符串完整     */
   puts(p);      /*  由于 p 没移动，通过 p1 的移动在其后面连接了 p2 指针所描述的串，因此输出的是连接后的新串  */
   getch();
   }
```

说明：

用指针引用字符串过程中，指针当前的位置直接影响到字符串的值。

用指针描述字符串比使用字符数组方便。不用指定数组长度，同时还可以在定义后直接用赋值运算符赋值，不会浪费内存空间。

【例 5.7】利用指针数组变量，将二维字符数组中的各字符串在不改变原来顺序条件下排序输出，实现对数据的索引。

```
#define   N   5
#include<string.h>
#include<stdio.h>
main()
{ int i,j;
   char *str[N],*t;    /*   定义指针数组和用于指针交换的中间变量   */
```

```
    char p[ ][10]={"Monitor","Landscape","Paddle","Partition","Current"};
    for(i=0;i<N;i++)
   /*   使指针数组中的每一个指针元素依次指向二维字符数组每行的首地址   */
        str[i]=p[i];
    for(i=1;i<N;i++)   /* 用冒泡排序法，使指针数组中的指针元素依次指向由小到大的串，但数组 p 的值没有更改 */
      for(j=0;j<N-i;j++)
          if(strcmp(str[j],str[j+1])>0)
            {   t=str[j];
                str[j]=str[j+1];
                str[j+1]=t;
            }
      printf("------------------------Original String------------------------\n");
  for(i=0;i<N;i++)     /*   输出原始数组的值     */
        printf("%12s",p[i]);
  printf("\n------------------------Result String------------------------\n");
    for(i=0;i<N;i++)     /*     输出排序数组的值     */
        printf("%12s",str[i]);
    }
```

说明：

二维字符数组是由多个一维字符数组组合而成。

指针数组的定义形式是：char *str[N];其中每一个元素是一个指针变量，必须对每一个元素进行初始化，使其有明确的指向。

将二维数组看成一维数组，其中的每一个特殊元素如 p[i]是一个指针类型的常量值，相当于一个一维数组名，表示二维字符数组中的一行的首地址，用于引用一个字符串。

程序中的冒泡排序法是使指针数组中的各指针元素按字符串的升序顺序交换各指针的指向对象，但并不交换二维字符数组中各字符串原来位置。如有：int a=1,b=3;int *pa=&a,*pb=&b,*pc;做如下运算：pc=pa，pa=pb，pb=pc；实现了使指针 pa 指向了变量的地址，pb 指向了变量 a 的地址，但变量 a、b 的值并没有交换。

指针引用字符串时，从当前指针指向的位置开始，依次向后到字符串结束标记之间的字符，都是字符串中的元素。

当使用指针数组表示字符串时，字符串依次在内存中顺序保存，串间用结束标记分开，没有任何多余空间，能有效杜绝对存储空间的浪费；当采用二维数组保存时，每串所占空间最少是串中最长者所需要的空间，当串长度相差较大时，浪费存储空间越严重。

【例 5.8】利用行指针，将二维字符数组中的各字符串在不改变原来顺序条件下，按行倒序输出。

```
#define   N   5
#include<string.h>
#include<stdio.h>
main(   )
{ int i,j;
   char *t;
   char p[][10]={"Monitor","Landscape","Paddle_12","Partition","Current_7"};
   char (*str)[10];   /*     定义行指针，每行 10 个元素          */
   printf("\n----------------------Original String----------------------\n\n");
   for(i=0;i<N;i++)
```

```
        printf("%12s",p[i]);
    str=p;     /*        使行指针指向数组 p 的首行        */
    printf("\n\n------------------------Result String-----------------------\n\n");
    printf("          ");
    for(i=0;i<N;i++)
        { for(j=0;*(*(str+i)+j);j++);
  /*   通过循环，用变量 j 定位第 i 行的结束标记位置     */
          for(j--;j>=0;j--)      /*   从最后元素位置倒序输出       */
              printf("%c",*(*(str+i)+j));
          printf("      ");
        }
    printf("\n");
    }
```

说明：

行指针的定义格式是：char (*str)[N];专用于指向二维数组的一行，其中 N 表示一行中元素的个数；str 其本质是二级指针，其指向二维数组的方式是 str=p;其意思是指向二维数组中的第一行。

str+i 是表示二级指针移动，*(str+i)指第 i 行的首地址，可以用来引用第 i 行所表示的字符串，当然也可以用数组名的*(p+i)指第 i 行的首地址，即二维数组名相当于二级指针；*(str+i)+j 是第 i 行第 j 列元素的地址，*(*(str+i)+j)表示第 i 行第 j 列元素，即*(*(p+i)+j)。str[i]即*(str+i)，*(*(str+i)+j)即*(str[i]+j)或*(p[i]+j)，总之，当一个行指针指向二维数组后，可以用指针名替换数组名引用数组中的字符串或字符。

5. 利用指针动态分配数组长度

【例 5.9】从键盘上输入某门课程的学生人数并输入学生成绩，统计该门课程的平均成绩及最高和最低成绩。

```
#include"stdio.h"
#include"stdlib.h"
main( )
   { int n,i;
     int *p,max,min,sum;
     printf("Please input members of studen:");
     scanf("%d",&n);
     p=(int *)malloc(n*sizeof(int));
/*  为指针 p 分配由 n 个整型数所需的的连续存储空间，p 指向其首地址   */
     if(!p)                /*     内存空间分配失败，则提示并退出程序的执行   */
         {printf("Not Enough Memory!\n");
          exit(0);
          }
     printf("Please input %d school report card:",n);
     max=-32768;min=32767;sum=0;
     for(i=0;i<n;i++)   /*对 p 指向的存储内输入 n 个元素值，找出最大最小值，并累加     */
         {scanf("%d",p+i);
          if(max<*(p+i))    max=*(p+i);
          if(min>*(p+i))    min=*(p+i);
```

```
        sum+=*(p+i);
        }
    free(p);    /*    释放分配给指针 p 引用的存储空间      */
    printf("The Max=%d\nThe Min=%d\n",max,min);
    printf("The Average=%.2f\n",sum*1.0/n);
    }
```

说明：

为解决在编写程序过程中对未知数据量用数组存储导致浪费存储空间或分配空间不足的实际问题，C 语言提供动态内存分配的处理函数，malloc( )和 calloc( )。

malloc( )和 calloc( )的常用方法是：(类型标示符 *) malloc (无符号整数)；(类型标示符 *)calloc(元素个数，每个元素所需存储空间值)，其中常用 malloc（ ）函数。 malloc( )分配连续指定长度（以字节为单位）的存储空间，calloc( ) 分配多块连续的容量为（元素个数*每个元素所需存储空间值）的存储空间。

两个函数的返回值都是被分配存储空间的首地址。

malloc( )和 calloc( )的常用于在内存中申请临时存储空间，用于保存需受保护的临时数据，使用后应该释放被分配的存储空间，以便系统可以重新分配。当然，当程序运行结束后，被分配的空间会自动释放。释放函数是 free( )，使用格式是：free(指向被分配地址段首地址的指针变量)。

6. 利用指针引用指针变量

【例 5.10】使用二级指针引用指针数组中的各字符串。

```
#define N 5
  main( )
     {
     int i,j;
     char **p;    /*定义二级指针——指向指针的指针变量    */
     char *str[ ]={"monitor",     /*     定义指针数组并赋值          */
                   "landscape",
                   "paddle",
                   "partition",
                   "current"
    };
        clrscr();
        p=str;          /*    使指针变量 p 指向指针数组 str       */
    for(i=0;i<N;i++)
   {
    printf("\nP Add:%-4x",p+i);          /*    输出存二级指针的地址       */
    printf("*P Add:%-4d",*(p+i));
   /* 输出二级指针移动 i 位置时对应存储单元中的地址值，即串的首地址*/
    printf("Character string:%-10s ",*(p+i));      /*          输出字符串          */
    printf("str[i]   Add   %d",str[i]);          /*       输出串的首地址       */
    }
}
```

输出结果：

P Add：ffd2　*P Add：414 Character string：monitor　　str[0] Add　414
P Add：ffd4　*P Add：422 Character string：landscape　str[1] Add　422
P Add：ffd6　*P Add：432 Character string：paddle　　str[2] Add　432
P Add：ffd8　*P Add：439 Character string：partition　　str[3] Add　439
P Add：ffda　*P Add：449 Character string：current　　str[4] Add　449

说明：

指针数组与二级指针属同级指针类型。p=str 表示指针 p 指向 str 首地址，变量 p 存储单元内的值是字符串”monitor”的首地址。p+i 存第 i 个串的首地址。

*(p+i)是取二级指针 p 移动 i 位置时地址中存的变量值——第 i 个字符串的首地址。当用整型格式输出时，表示地址；用字符串格式输出时，表示从该地址开始到结束标记间的字符串。

当用二级指针或指针数组表示字符串时，对内存利用率最高。

【小结】

指针变量存储的值是被指向对象变量的地址，即存储单元的内存编号；变量的地址存放于一个指向该变量的指针变量存储单元中。利用指针可以引用被指向变量的值。指针变量定义后，必须赋值，即使其有明确的被指对象，否则指针处于浮动状态，不能对指针做取值运算，也不能给指向的对象赋值；如：int　*p;*p=10;或 int *p,i; i=*p;。

数组名是指针概念。一维数组名是一级指针，指向数组首地址，并且是常量；二维数组名是二级指针概念，其中的一个特殊元素如 a[i]，是一级指针，指向第 i 行首地址。

字符型指针变量用于引导字符串，比使用数组存储字符串更简单和适用。用字符指针描述字符串时，定义指针后，可以用赋值运算符单独赋值；而用数组存储字符串，定义以后只能用输入函数或 strcpy 函数以串形式赋值；共同点是在定义的同时都可以用串格式赋初值。

指针数组和行指针及二维数组名是二级指针概念，即指向指针的指针类型，其值是一个指针。如有：char　a[M][N], *b[M] ,(*c)[N],**p,i;有如下运算方式：for(i=0;i<M;i++) b[i]=a[i]; c=a;　p=a; 其中 a[i]，b[i]，p[i]，c[i]，*(c+i) ，*(a+i)，*(b+i),*(p+i)都表示第 i 行的首地址；而 a[i][j]，b[i][j]，p[i][j]，c[i][j]，*(*(c+i)+j)，*(c[i]+j)，*(*(a+i)+j)，*(a[i]+j)，*(*(p+i)+j)，*(p[i]+j)，*(*(c+i)+j)，*(c[i]+j)都表示第 i 行 j 列元素的值。在二级指针使用过程中，特别注意赋值和引用格式。