

第五次实验报告 数组

韦璐 PB16000702

一. 数组是同类型数据元素的有序集合, 即数组由基本类型(整形, 实型, 字符, 指针等数据类型)组合而成。这种组合可以是多层的, 但最低一级的成员必须是基本类型的元素构成, 即数组元素必须是基本类型的量。

1. 一维数组的定义

数组必须先定义后使用。一维数组一般定义为: 类型标识符 数组名【常量表达式】,;

(1) **【】**: 为数组下标运算符。在此表示定义一个指定类型和名称的数组, 并为其开辟由常量表达式确定大小的连续存储空间。

(2) 数组名: 代表该数组存储空间的起始地址, 取名规则同变量。

(3) 常量表达式其值规定所定义数组的长度(大小)。定义数组时必须通过这个常量指定数组的大小, 不能包含变量, 且必须是正整型常量表达式。

(4) 数组的下标: 从零开始, 最后一个元素的下标是“数组的长度-1”。

(5) 一维数组在两种情况下可以省略数组长度的说明, 而用[]替代。

1) 一维数组名作为被调函数的参数, 其存储空间和大小由主调函数的实参数组传递确定。

2) 对一维数组赋初值意味着对全体数组元素赋初值, 此时数组的长度可以由初值个数确定。担当初值的个数不等于数组元素的个数时, 长度必须说明。

2. 一维数组的引用

(1) 一维数组元素可以被当做一个普通变量处理, 即数组元素可出现在变量能出现的地方。

数组元素表示形式为: 数组名【下标】。

(2) 数组名作为函数调用的参数。数组名代表该数组的起始地址, 数组名做参数, 其实质是以传地址方式传递数据, 以实现形参和实参共享存储空间, 具有带值返回功能。

(3) C 语言不检查数组边界, 当超越数组边界的时候, 系统编译时不会提示错误, 但运行的时候将导致运算结果错误或运行错误, 甚至可能导致其他变量或程序的破坏。

3. 一维数组的初始化

一般形式:

类型标识符 数组名【常量表达式】={元素值表列}

(1) 元素值表列可以是数组所有元素的初值, 也可以是前面部分的元素的值。对数值型数组, 初始化时未明确设定元素的初值, 则其值被自动设置为零。

(2) 对全体数组元素赋初值时, 元素个数可以忽略, 但“**【】**”不可以忽略。此时系统将根据数组初始化时大括号内值的个数, 决定该数组元素的个数。

(3) 数组初始化的赋值方式只能用于数组的定义中, 定义之后再赋值就只能一个元素一个元素地赋值。

4. 多维数组

多维数组的定义方式为:

类型标识符 数组名【常量表达式 1】【常量表达式 2】【常量表达式 3】.....;

(1) 多维数组的数组名, 每维元素个数的要求同一维数组。

(2) 多维数组中, 元素是按行存放的, 即按顺序先存放第一行的数组元素, 再存放第二行的数组元素, 以此类推。

(3) 由多维数组按行存储方式, 可以把二维数组理解为由低一维数组复合而成, 即构造而成。如二维数组 a[3][4]数组可以看成是由 a[0], a[1], a[2]三个元素组成, 而每个元素都是一个一维数组, 所以可以把他们看成是一维数组名, 各自代表一维数组的

起始地址。以此类推，对多维数组的元素如意三维数组 `int a[3][4][5]` 的元素 `a[i][j][k]` 为例，可以按照如下方式理解：`a[i][j][k]` 层 行列 是元素，为代表值 `a[i][j]` 一维数组名，代表一维数组的起始地址 `a[i]` 二维数组的名字，代表二维数组的起始地址 `a` 三维数组名，代表三维数组的起始地址

5. 多维数组的引用

(1) 多维数组元素同样可被当做一个普通变量处理，可以出现在任何同类型变量能出现的地方。多维数组元素表示形式。

(2) 数组的维数不同，引用数组元素的下标也就不同。

(3) 同一维数组一样，多维数组的下标可以是整型常量，整型变量，或者整型表达式。每一维下标都从 0 开始。

6. 多维数组的初始化

(1) 可按照行对多维数组初始化

```
Static int a[2][3]={1,2,3},{4,5,6};
```

(2) 可按照数组的存储顺序赋值

```
Static int a[2][3]={1, 2, 3, 4, 5, 6};
```

(3) 如果对多维数组全部赋初值，则最左边一维的元素个数可以省略，但其余各维数必须制定。

(4) 可以只对部分元素赋值。对数值型数组，如果进行初始化的时候未明确设定元素的初值，则数组元素值自动设置为 0。`static int a[2][3]={1,2},{4};` 那么其他没有设置的元素的值为零。

(5) 同一维数组一样，多维数组初始化的赋值方式只能用于数组的定义，定义之后再赋值就只能一个一个。

7. 数组元素做函数参数

数组元素可以作为函数的实参，此时与同类型的简单变量作为实参一样，属于单项的值传递。数组元素的值传递给形参后，形参的改变不影响作为数组元素的实参。

8. 数组名做函数参数

数组名作为函数参数，此时实参和形参都应该是数组名，或者是表示地址的指针变量；数据的传递方式属于传地址方式，传递的是整个数组。

(1) 数组名表示的是数组的起始地址，数组名做函数的实参的时候，传递的是数组的首地址，其实质是以传地址方式传递数据，实现形参和实参共享存储空间，所以具有带值返回的功能，即形参数组元素的值改变的时候，实参数组元素的值也改变了。

(2) 形参中的数组不但要定义，而且要求与是参数组的类型一致。

(3) 多维数组作为函数参数在参数说明的时候与一维数组不同，一维数组可不指明长度，而多维数组除最左边一维可以不指明长度外，其余各个维度均要指明长度。

9. 字符数组

字符串可通过字符数组实现；字符串的处理是基于字符数组的，即具体操作通过字符数组的元素的运算实现。字符数组的定义为：`char 数组名【常量表达式】`；`char 数组名【常量表达式 1】【常量表达式 2】...`；

(1) 字符串常量是由双引号括起来的字符序列，在实际存储时系统自动在其尾部添加了一个字符串结束标志，`\0`，所以采用字符数组存放字符串的时候，应该包含字符串结束标志。

(2) 字符数组的元素个数应该大于或等于字符串长度+1，增加的一个字节用于存放字符串结束标志。

10. 字符数组的初始化和赋值

字符数组的初始化和赋值操作时要特别注意字符串的结束标志的处理。

- (1) 采用将字符逐个赋给数组中各个元素的初始化形式时，串尾应包含字符串结束标志。
`Char s[9]={ 'z','h','\0' }`
- (2) 直接用字符串常量给字符数组初始化时，系统自动在串尾加字符串结束标志。
- (3) 单个字符赋值时候，注意结束
- (4) 除初始化外，不能采用“字符数组=字符串常量”的赋值形式
- (5) 二维字符数组可以看作是这样一个一维数组，每个元素是一个字符串。
- (6) 二维字符数组的元素和名字都可以作为函数 参数，使用方法和二维数组的使用方法相同。

11. 字符数组的输入输出

设有：`char name[20];`

(1) 可用格式符“%c”输入输出某个元素。例如：

```
scanf("%c",&name[i]);
```

```
printf("%c",name[i]);
```

- (3) 用字符输入函数 `getchar()` 一个字符一个字符地输入赋给数组元素，最后用赋值语句补上字符串结束标志。
- (4) 用字符输出函数可以一个字符一个字符输出。
- (5) 可用“%s”输入输出一个字符串。利用 `scanf("%s",name);`

注意：

- 用“%s”输入串时，遇到空格结束，因此输入串的中间不能出现空格，否则将作为串输入结束处理。
- 数组名代表该数组存储单元的起始地址，故作为输入参数不用“&”取地址符。
- 系统自动在串尾加结束符。
- 正确区分字符串长度与数组长度这两个不同的单位。没看懂什么意思。

注意：

- ✓ 与“%s”对应的输出参数要求为字符数组名（地址常量）或字符串的起始地址，系统从第一个字符起依次输出串的内容，直到遇到结束符结束。
- ✓ 与“%s”对应的输出参数也可以是串中的某个字符的地址，还可以是指向字符串任意位置的指针变量，系统从指定位置起依次 输出串的内容，知道遇到结束符结束。
- ✓ 使用 `gets()`和 `puts()`输入输出字符串。只要在括号中加入字符串的起始地址的名字就好了。`Gets` 函数遇到回车键结束字符串输入，并自动加上字符串的结束标志。因此输入串中可以包含空格和 `tab` 在内的全部字符，用于字符串的输入比较方便。
- ✓ 可调用复制字符串函数 `strcpy` 对字符串整体赋值。例如：`char s[9];strcpy(s,"zhangsan");`

12. 常用字符串操作库函数与编程

C 语言提供了很多字符串操作函数，对应的头文件 `string`，用到这些函数时，需要在程序的开头加行预处理命令。

常用的字符串操作函数：

- (1) `strlen` (字符串)：返回字符串的长度。(注意：这个长度不包含字符串的结束符)
- (2) `strcpy` (字符数组 1, 字符串 2)：将字符串 2 (包括结束符)复制到字符数组 1 中。字符数组 1 要足够大。
- (3) `strcat` (字符数组 1, 字符串 2)：将 2 的内容复制到 1 中串的尾部，并返回字符数组 1 的地址。1 要足够大，连接后 1 中的结束符号被覆盖，新的串尾保留一个结束符。
- (4) `strcmp` (字符串 1, 字符串 2)：比较字符串 1 和 2 的大小。两个字符串从左到右逐个字符比较，按照 `ascii` 码的大小，直到字符不同或者遇见结束符为止。如果全部字

符都对应相同且串长相等，则返回值为 0。如果不相同，则返回两个字符串中第一个不相同字符的 ascii 码值的差值，即 1 大于 2 时函数值为正，否则为负。

- (5) 以上函数参数中出现的字符串可以是字符串常量，存放字符串的字符数组名，并且要包含结束符，或者要处理的字符串的第一个字符的地址（字符型指针）；同样字符数组可以是数组名或者要吃力的字符串的第一个字符的地址（字符型指针）。

13. 以数组为基础的常用算法

以数组为基础可以构建很多使用算法，常用的有：

- (1) 冒泡法，选择法和希尔法等排序算法。
- (2) 筛选法求 1 至指定书之间的素数。
- (3) 折半查找
- (4) 字符串连接，拷贝和比较等字符处理
- (5) 矩阵的相加和相乘等操作

二．实验操作

范例

题目一

实验代码：

```
#include<stdio.h>
main(){
    static int a[6]={10,20,30,40,50};
    static char b[6]="USTC";
    printf("\na: ");
    for(i=0;i<6;i++)
        printf("(%x)%-4d",&a[i],a[i]);//十六进制输出整数
    printf("\nb: ");
    for(i=0;i<6;i++)
        printf("(%x)%-4c",&b[i],b[i]);//十六进制的 ascii 码加上字符型的输出
    printf("\nb: ");
    for(i=0;i<6;i++)
        printf("(%x)%-4d",&b[i],b[i]);//十六进制的码加上整型的输出
    printf("\nString: %s\n",b);    //以字符串的形式进行输出  ? ? ? ? ? ? ?
}
```

实验结果：

```
a: (403010)10 (403014)20 (403018)30 (40301c)40 (403020)50 (403024)0
b: (403028)U (403029)S (40302a)T (40302b)C (40302c) (40302d)
b: (403028)85 (403029)83 (40302a)84 (40302b)67 (40302c)0 (40302d)0
String: USTC
```

结果分析：

1. 括号内是十六进制的地址值，不同计算机上，地址值可能是不同的。
2. 观察结果，可以看到同一数组各元素的地址是连续的，不同类型数组元素的存储单元的大小是不同的。
3. 使用“%d”输出字符数组各元素的 ASCII 码值，可以看到空字符的 ASCII 码值为零。
4. 使用“%s”输出存储于字符数组中的一个字符串。

题目二：

对一个四行四列的二维数组，完成以下操作：

- (1) 求对角线上的元素之和与积。

(2) 求所有靠边元素之和与积。

(3) 求所有不靠边的元素之和与积

分析：对于一个行列相同的二维数组，有两条对角线，一条称为主对角线，另一条称为副对角线，主对角线的特点是下标的行列值相等；副对角线的下标特点是行列下标之和等于行数-1，所有靠边的元素要么行为 0 要么为行数减 1 要么列下标为 0 要么为列数-1；所有不靠边的元素特点是满足上述四点的否定。

实验代码：

```
#include<stdio.h>
main(){
    int x[4][4],i,j;
    long djh=0,djj=1,kbh=0,bkbh=0,kbj=1,bkbj=1;
    for(i=0;i<4;i++){
        printf("请输入第%d 行: ",i+1);
        for(j=0;j<4;j++){
            scanf("%d",&x[i][j]);
        }
        for(i=0;i<4;i++){
            for(j=0;j<4;j++){
                if(i==j||i+j==3){
                    djh+=x[i][j];//求对角线元素之和
                    djj*=x[i][j];//求对角线元素之积
                }
                if(i==0||i==3||j==0||j==3){
                    kbh+=x[i][j];//求靠边元素之和
                    kbj*=x[i][j];//求靠边元素之积
                }
                if(i!=0&&i!=3&&j!=0&&j!=3){
                    bkbh+=x[i][j];
                    bkbj*=x[i][j];
                }
            }
        }
        printf("对角线元素之和=%ld,对角线元素之积=%ld\n",djh,djj);
        printf("靠边元素之和=%ld,靠边元素之积=%ld\n",kbh,kbj);
        printf("不靠边元素的和=%ld, 不靠边元素的积=%ld\n",bkbh,bkbj);
    }
}
```

实验结果：

请输入第1行: 1 2 3 4	请输入第1行: 1
请输入第2行: 5 6 7 8	2
请输入第3行: 9 10 11 12	3
请输入第4行: 13 14 15 16	4
对角线元素之和=68, 对角线元素之积=3843840	请输入第2行: 5
靠边元素之和=102, 靠边元素之积=233775104	6
不靠边元素的和=34, 不靠边元素的积=4620	7
-----	8
Process exited after 19.88 seconds with	请输入第3行: 9
请按任意键继续. . .	10
	11
	12
	请输入第4行: 13
	14
	15
	16
	对角线元素之和=68, 对角线元素之积=3843840
	靠边元素之和=102, 靠边元素之积=233775104
	不靠边元素的和=34, 不靠边元素的积=4620

结果分析:

我发现 scanf 很奇怪, 两种输入都可以, 也许是因为空格和回车都算是它的间隔符?

第三题

编程输出杨辉三角形的前十行:

题目分析: 这些数据的特点是对角线和第一列上的值为 1; 其余从第三行起, 每行自第二个数开始的每个数等于上一行同一列上的数之和。

实验代码:

```
#include<stdio.h>
main(){
    int a[10][10],i,j;
    for(i=0;i<10;i++){
        a[i][0]=1;
        a[i][i]=1;
    }
    for(i=2;i<10;i++){
        for(j=1;j<i;j++){
            a[i][j]=a[i-1][j]+a[i-1][j-1];
        }
        for(i=0;i<10;i++){
            for(j=0;j<=i;j++){
                printf("%5d",a[i][j]);
                printf("\n");
            }
        }
    }
}
```

实验结果:

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1

```

第四题

题目：设有两个矩阵，求它们的矩阵乘积，要求通过编写矩阵乘法的函数实现。根据矩阵乘法规则，会有计算公式。

实验代码：

```

#include<stdio.h>
#include<math.h>
matproduct(int a[][3],int b[][2],int c[][2],int m,int n,int p){
    int i,j,k,s;
    for(i=0;i<m;i++){
        for(j=0;j<p;j++){
            for(k=0;k<n;k++) s+=a[i][k]*b[k][j];
            c[i][j]=s;
        }
    }
}
main(){
    static int a[3][3]={1,1,1},{2,2,2},{3,3,3};
    static int b[3][2]={4,4},{5,5},{6,6};
    int c[3][2],i,j;
    matproduct(a,b,c,3,3,2);
    for(i=0;i<3;i++){
        for(j=0;j<2;j++){
            printf("%6d%c",c[i][j],((j+1)%2==0)?'\n':' ');//注意这里!! 在后面的“里面，一定要加空格，
            //不然编译不通过!
        }
    }
}

```

实验结果：

```

15 15
30 30
45 45

```

第五题

编制程序，读入一串正文行，打印出其最长行的内容及串长，然后再对最长行采用选择法按照从小到大排序输出。

题目代码：

```
#include<stdio.h>
#define MAXLINE 100
main( ){
    int getline(char s[],int lim);
    void copy (char s1[],char s2[]);
    void sort(char s[],int l);
    int len,max=0;char line[MAXLINE],save[MAXLINE];
    while((len=getline(line,MAXLINE))>0)
        if(len>max){
            max=len;
            copy(line,save);
        }
        if(max>0)
            printf("%d %s\n",max,save);
            sort(save,max);
            printf("%d %s\n",max,save);
}

getline(char s[],int lim){
    int c,i;
    for(i=0;i<lim-1&&(c=getchar())!=EOF&&c!='\n';i++)
        s[i]=c;
    if(c=='\n')
        s[i++]=c;
    s[i]='\0';
    return(i);
}

void copy(char s1[],char s2[]){
    int i=0;
    while((s2[i]=s1[i])!='\0')
        ++i;
}

void sort(char s[],int l){
    int i,j,k;
    char t;
    for(i=0;i<l-1;i++){
        k=i;
        for(j=i+1;j<l;j++)
            if(s[j]<s[k])k=j;
        t=s[k];s[k]=s[i];s[i]=t;
    }
}
```


实验结果：

```
test
the test
test3
^Z
9 the test
9
eehsttt
```

实验总结：

不知道为什么，我的结果和书本上的结果有一些区别，可能是输入的问题？

第六题：

题目：

在同一图纸上打印出以下的方程的图形：

$Y1=\sin x$ 用“*”在 ± 180 度的范围，间隔是15度

$Y2=x/3$ 用“¥”，在 ± 180 度里面

- (1) $y1=\sin x$ 用“*”打印， x 取值在 ± 180 度间所有15度的整数倍的点
- (2) $y2$
- (3) 曲线的中心轴用.表示，位置在第40列
- (4) 设计这个程序还需要考虑如下两个问题：
 1. 在第一行中有可能要打印三个函数的点，分别是两个函数和中心轴线。我们在这里采用的是根据函数值来确定打印字符的位置，并由此来置换字符串的值。我们设字符数组的内容为80个空格符，这样不在函数值的位置上的字符为空格，使图形清晰。
 2. 因为 $\sin x$ 的值在 ± 1 之间，因此如果不考虑振幅，其函数值密集，使得打印的图形无法正确显示出正弦曲线，因此程序中应该做放大处理。

实验代码：

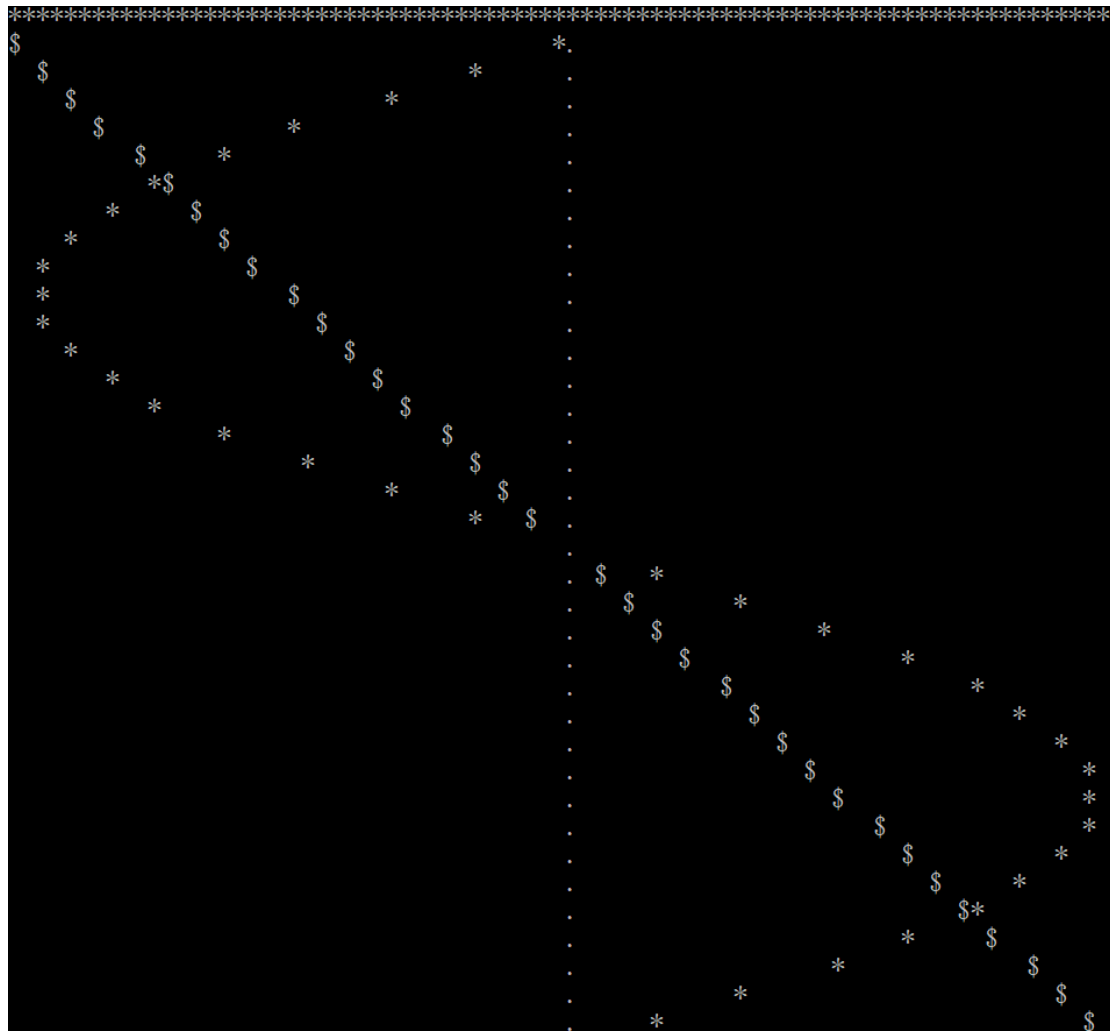
```
#include<stdio.h>
#include<math.h>
main() {
    int i,iy1,iy2;
    char a[80],b[80];
    double x,y1,y2;
    for(i=0;i<80;i++) {
        a[i]=' ',b[i]='*';
    }
    a[79]='\0';b[79]='\0';
    printf("%s\n",b);
    x=-3.1415926;
    while(x<3.1415926) {
        y1=sin(x);
```

```

        y2=x/3.0;
        iy1=40+38*y1;//本行和下一行的 38 为调整放大振幅的调整系数，可以
适当改变数值观察图形的变化
        iy2=40+38*y2;
        a[iy1]='*';
        a[iy2]='$';
        a[40]='.';
        printf("%s\n",a);
        a[iy1]=' ';
        a[iy2]=' ';
        x=x+3.1415926/18;
    }
}

```

实验结果:



第七题

实验题目:

若矩阵某位置上的元素在该行上为最小，而在该列上为最大，则称该位置为矩阵的鞍点。编程给出矩阵中所有的鞍点。

具体要求：编制一个通用的输出给定矩阵中所有鞍点的函数 andian (int a[], int m, int n)。其中，a 所对应的实参为二维数组，且为 m 行 n 列的矩阵。

分析：

从矩阵的第 0 行开始直到最后一行为止，逐行寻找该行中值最小的所有元素，然后对其中的每一个元素所在的列寻找最大值，如果该列上的组大致与该行的最小值相等，则说明钙元素为鞍点，将它所在的行号和列号输出。

为了使函数 andian 中的一维数组与矩阵对应，需要进行下标变换。

函数的流程图大概：就是 andian 输入为一个数组的首地址和行列的大小，然后定义几个变量，定义一层循环，然后是另外两层循环。

实验代码：

```
#include<stdio.h>
#define M 10
#define N 10
zhaoandian (int a[],int m,int n){
    int i,j,k,kk,d,p,t;
    for(i=0;i<m;i++){
        d=a[i*N];
        for(j=k=0;j<n;j++){
            t=i*N+j;
            if(a[t]<d){
                d=a[t];k=j;
            }
        }
        for(j=k;j<n;j++){
            t=i*N+j;
            if(a[t]==d){
                p=a[t];
                for(kk=0;kk<m;kk++){
                    t=kk*N+j;
                    if(a[t]>p) p=a[t];
                }
                if(d==p) printf("andian:%5d%5d\n",i,j);
            }
        }
    }
}

main( ){
    int i,j,k,m,n,a[M][N];
    printf("enter m,n:");
    scanf("%d%d",&m,&n);
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
}
```

```

        zhaoandian (*a, m, n);
    }

```

实验结果：

```

enter m,n:3 4
1 2 3 4
9 8 10 8
3 5 6 7
andian:    1
andian:    1

```

实验题目：

第二题

题目：

输入 n 个整数，n 比二十小，存放在一个一维数组中。然后在第 m 个整数后插入一个输入的整数，后续的元素都后移一位，输出最终的数组。

实验代码：

```

#include<stdio.h>
#include<math.h> //输入 n 个整数，n 比二十小，存放在一个一维数组中。然
后在第 m 个整数后插入一个输入的
//整数，后续的元素都后移一位，输出最终的数组。
main() {
    int a[21];
    int b[21];
    int n, p;
    for(int i=0; i<20; i++) {
        scanf("%d", &a[i]);
    }
    printf("你想放什么在第几个数的后面: ");
    scanf("%d%d", &p, &n);
    for(int i=0; i<n; i++)
        b[i]=a[i];
    b[n]=p;
    for(int i=n+1; i<21; i++)
        b[i]=a[i-1];
    for(int i=0; i<21; i++) {
        a[i]=b[i];
        printf("%d ", a[i]);
    }
}

```

实验结果：

```

1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 15 5 5
你想放什么在第几个数的后面67 18
2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 15 5 67

```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
你想放什么在第几个数的后面21 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
```

第三题

题目

输入 n 个整数，小于二十，存放在一维数组中，然后输入一个数 m，并将第 m 个数删除，补缺输出。

实验代码：

```
#include<stdio.h>
#include<math.h>
main() {
    int a[20];
    int b[21];
    int n,p;
    for(int i=0;i<20;i++){
        scanf("%d",&a[i]);
    }
    printf("你想删除第几个数");
    scanf("%d",&n);
    if(n==20)
        a[19]=0;
    else
        for(int i=n;i<20;i++)
            a[i-1]=a[i];
        a[19]=0;
        for(int i=0;i<20;i++)
            printf("%d ",a[i]);
}
```

实验结果：

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
你想删除第几个数20
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

C:\Users\韦璐\Desktop\计算机程序设计\实验五\未命名9.exe

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
你想删除第几个数4
1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 0

C:\Users\韦璐\Desktop\计算机程序设计\实验五\未命名9.exe

1 2 3 4 5 6 7 8 9 10 11 12 13 14 1 516 17 18 19 20
你想删除第几个数7
1 2 3 4 5 6 7 7 7 7 7 7 7 7 7 7 7 7 0
```

第四题

题目：

已知两个二十个元素的数组，其中 b 是升序的数组，编写程序将 a 数组中的各个元素插入 b 数组，并保证 b 仍未升序数组。

实验代码：

```
#include <stdio.h>

int* merge(int *a, int lena, int *b, int lenb, int *c) {

    int i = 0, j = 0, k = 0; // 分别代表数组 a , b , c 的索引

    while (i < lena && j < lenb) {

        if (a[i] < b[j])

            c[k++] = a[i++];

        else

            c[k++] = b[j++];

    }

    while (i < lena)

        c[k++] = a[i++];

    while (j < lenb)

        c[k++] = b[j++];

    return c;

}

main()

{

    int b[20] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};

    int a[20] = {7, 9, 6, 32, 9, 4, 2, 3, 1, 33, 4, 6, 55, 76, 84, 37, 8, 77, 46, 98};
```

```
int c[40];

int s, t;

for(int i=0; i<20; i++) /*访问排序前的数组*/
{
    s=i;

    for(int j=i+1; j<20; j++)

        if(a[s]>a[j])

            s=j;

    if(s!=i)

        /*把这数组中最小的数字放到前面*/

        t=a[s];

        a[s]=a[i];

        a[i]=t;

    }

}

for(int i=0; i<20; i++) {

    printf("%d ", a[i]);

}

printf("\n");

merge(b, 20, a, 20, c);

for(int i=0; i<40; i++) {

    printf("%d ", c[i]);

}
```

```
}  
  
}
```

实验结果：

```
1 2 3 4 4 6 6 7 8 9 9 32 33 37 46 55 76 77 84 98  
1 1 2 2 3 3 4 4 4 5 6 6 6 7 7 8 8 9 9 9 10 11 12 13 14 15 16 17 18 19 20 32 33 37 46 55 76 77 84 98
```

实验总结:printf 的时候后面不能用地址符号!!! 然后数组的名字直接就代表地址!! 很方便!! 还有定义指针函数还有函数里的地址参数都要加星号!! 然后就是, i++是一个拥有神奇的自己加的功能的东西!! 函数的功能有一个多余了, 因为这里是指定了数组的大小, 大小不确定的时候函数的功能就能派上用场了!!

第五题

题目：

从键盘输入十个学生的姓名和成绩，分别用冒泡排序法和选择排序法按成绩降序输出，要求姓名和成绩对应关系不变。

冒泡排序法实验代码：

```
#include <iostream>  
  
#include <string.h>  
  
using namespace std;  
  
int main()  
{  
  
    int i, j, temp2, a[10]; //定义一个长度为 10 的数组 a  
  
    string name[10], temp1; //定义长度为 10 的字符串数组 name  
  
    for(i=0; i<=9; i++) //循环输入学生名字  
  
        cin>>name[i];  
  
    for(i=0; i<=9; i++) //循环输入学生成绩  
  
        cin>>a[i];  
  
    for(i=0; i<9; i++) //使用双层循环，对名字进行排序  
  
    {  
  
        for(j=0; j<9-i; j++)
```



```

        {
            if(a[j]<a[j+1])//如果前者大于后者，两者进行交换，同时对应
            成绩也进行交换
            {
                temp1=name[j];

                name[j]=name[j+1];

                name[j+1]=temp1;

                temp2=a[j];

                a[j]=a[j+1];

                a[j+1]=temp2;

            }

        }

    }

    cout<<endl;

    for(i=0;i<=9;i++)//循环输出排列好的名字和成绩

        cout<<name[i]<<" "<<a[i]<<endl;

    return 0;

}

```

实验结果：

```
qq w e d s x c f v g
1 2 3 4 5 6 7 8 9 10

g, 10
v, 9
f, 8
c, 7
x, 6
s, 5
d, 4
e, 3
w, 2
qq, 1
```

选择排序法实验代码：

```
#include <iostream>

#include <string.h>

using namespace std;

int main()
{
    int i, j, temp2, a[10]; // 定义一个长度为 10 的数组 a
    string name[10], temp1; // 定义长度为 10 的字符串数组 name
    for (i=0; i<=9; i++) // 循环输入学生名字
    {
        cin>>name[i];
    }
    for (i=0; i<=9; i++) // 循环输入学生成绩
    {
        cin>>a[i];
    }
    for (i=0; i<9; i++) // 使用双层循环，对名字进行排序
    {
```

```

        for(j=i+1;j<10;j++)

        {

                if(a[i]<a[j])//如果前者小于后者，两者进行交换，同时对应名字也进行交换，

                //这样就把最小的元素一个一个的放到最后并且放好以后之后的循环就不会涉及到它们了。

                {

                        temp1=name[j];

                        name[j]=name[i];

                        name[i]=temp1;

                        temp2=a[j];

                        a[j]=a[i];

                        a[i]=temp2;

                }

        }

}

cout<<endl;

for(i=0;i<=9;i++)//循环输出排列好的名字和成绩

        cout<<name[i]<<" "<<a[i]<<endl;

return 0;

}

```

实验结果：

```
q w e r t y u i o p
1 2 3 4 5 6 7 8 9 10

p, 10
o, 9
i, 8
u, 7
y, 6
t, 5
r, 4
e, 3
w, 2
q, 1
```

实验总结：

冒泡排序法和选择排序法的共性：

都是二层循环，都需要第三个变量来帮助排序。

冒泡排序法：内层循环中 i 是固定的，而 j 是进行遍历的。后面排好以后是固定的。

选择排序法：i 不固定，ij 进行比较。前面排好以后是固定的。

第六题

题目：

有 15 个数存放在一个数组中，输入一个数，要求用折半查找法找出该数是数组中第几个元素的值。如果该数不在数组中，则输出“无此数”。15 个数可采用数组赋初值的方法在程序中输出，要找的数用 scanf（）函数输入。

实验代码：

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

```
int main()
```

```
{
```

```
    int a[16] = {99, 97, 87, 76, 65, 63, 59, 58, 51, 43, 32, 31, 28, 15, 6, 1}; //输入数组，由于要求使用这般查找的方法，
```

```
    //而这个方法要求要有顺序，所以这个是升序的数组
```

int x, low, high, mid, n;//初始设定有几个未知数，mid 是中间用来交换的桥梁

n = 16;//对 n 进行初始化

low = 0;//对 low 进行初始化

high = n - 1;//对 high 进行初始化

scanf("%d", &x);//输入你要查找的数

```
while (low <= high) {  
  
    mid = (low + high) / 2;  
  
    if (x > a[mid])  
  
        high = mid - 1;  
  
    else if (x < a[mid])  
  
        low = mid + 1;  
  
    else if (x == a[mid]) {  
  
        printf("%d is %dth number!\n", x, mid+1);  
  
        break;  
  
    }  
  
}  
  
if (x != a[mid])  
  
    printf("No match!\n");  
  
return 0;  
  
}
```

实验结果：

```
32  
32 is 11th number!
```

实验总结：

这个方法对于有奇数个还是偶数个元素都**没有区分**，但是一定要求你的元素是按照一定的顺序排列的。

后面编写函数的时候还会使用下面的几个程序的主要部分，到那里再全部编成函数好了。

第七题

题目：

编制用户自定义函数，实现两个字符串连接（不要使用系统提供的 strcat 函数实现）

实验代码：

```
#include <stdio.h>

main()

{

char s1[20], s2[10]; //定义两个数组

int i=0, j=0; //定义两个用来循环的变量

gets (s1), gets (s2); //接收两个字符串

while (s1[i] != '\0') //找到当第一个字符串结束的时候的下标

{i++;}

while (s2[j] != '\0') //在第二个字符串没有结束之前，一直都往后遍历

{s1[i++] = s2[j++];}

s1[i] = '\0'; //读完之后给大的字符串加上结束的标志

puts (s1); //输出第一个大的字符串

}
```

实验结果：

```
sfneuvnev
dsvn
sfneuvnevdsvn
```

实验解读：

就是字符串的操作，注意结束标志的使用，然后由于是 `i++` 所以可以在每次都先进行操作之后再对 `i` 进行改变。

第八题

题目：

编制用户自定义函数实现按照字典顺序比较 `s`，`t` 两个字符串的大小，`s` 大于 `t` 返回正的值，`s` 等于 `t` 的时候返回 0，`s` 小于 `t` 的时候返回负数的值，正负值的大小正好是 `st` 中第一个所遇到的不相等的两个字符值之差。（不要使用系统提供的 `strcmp` 函数实现）

实验代码：

```
#include<stdio.h>

int sub (char *s,char *t) //定义函数的名字，然后这里是输入地址作为函数的参数

{for (;*s==*t;s++, t++)/*是用来取地址里面存放的数值，然后地址的首字母++就代表地址进行了移动

if (*s=='\0') return 0;//如果第一个地址已经结束了，就直接返回 0，因为前面几个字母都是相同的

return (*s-*t);//否则就返回不相等的两个字符的差值，而且是第一个地址的数值减掉第二个地址里的数值

}

main()

{

    char a[]="abcd";

    char b[]="abde";
```

```
printf("%d\n", sub(a, b)); //选择调用函数并且输出返回值

}
```

实验结果：

这里数组分别是 1234 1256 和上面程序中所选择的字符，而且返回来的差值是 ascii 码。

-2



-1

实验总结：

这个实验中第一要注意到 printf 函数可以用函数的返回值作为输出；第二就是函数用指针作为参数；第三就是函数的形式是整形是因为返回值；第四就是函数取出地址的数值之后只会取出 ascii 码；第五就是字符串是指针存放的；第六就是指针++就代表我这个名字的指向的地址产生了变化；第七就是字符串的结束标志很重要；第七个就是函数可以在任意位置进行返回，一旦返回了函数就结束了。

第九题

题目：编制用户自定义函数，实现将 t 所代表的字符串拷贝到 s 中，即字符串拷贝函数。（no strcpy）

实验代码：

```
#include <stdio.h>

void mystrcpy(char *a, char *b)

{

    while(*a != '\0' )

        *b++=*a++;

    *b='\0';

}

int main()
```



```

{

    char *a="test string",b[100];

    mystrcpy(a,b);

    printf("b is %s\n",b);

}

```

实验结果：

```
b is test string
```

实验总结：

不知道为什么，在函数中的命令放在主函数中就是不能编译通过，但是放在其他函数中就可以。

第 10 题

题目：

产生 50 个随机数，用插入排序法按照非递减顺序进行排序。具体要求如下：

- (1) 在产生随机数的过程中，每产生一个随机数就插入到前面已有序的数组中。
- (2) 输出时要求每行输出五个数据，并上下对齐。
- (3) 产生随机数的函数已经有了。也可以用库函数产生，但是要调用库函数 `stdlib.h`，为了拥有更好的随机性，在使用时间种子的时候需要使用 `time.h`，这个的大体的函数也已经给出了。
- (4) 已经给出了插入排序的大致的方法。

实验代码：

```

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void insertsort(int *r,int i,int x){

    while(i!=0&& r[i-1]>=x){

        r[i]=r[i-1];

        i--;
    }
}

```

```

    }

    r[i-1]=x;
}

main() {

    int i=0, x;

    int r[101];

    for(i=0; i<100; i++) {

        r[i]=2*i;

    }

    for(i=0; i<100; i++) {

        printf("%d ", r[i]);

    }

    srand((unsigned int)time(NULL));

    for(i=0; i<50; i++) {

        x=(int) (rand()%100+1);

        printf(" ! %d", x);

        insertsort(r, 100, x);

    }

    for(i=0; i<101; i++) {

        printf("%d\n", r[i]);

    }

}


```

实验结果:

```

C:\Users\韦璐\Desktop\计算机程序设计\实验五\未命名18.exe
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68
70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100 102 104 106 108 110 112 114 116 118 120 122 124 126
128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176
178 180 182 184 186 188 190 192 194 196 198 ! 89 ! 81 ! 9 ! 30 ! 72 ! 33 ! 66 ! 82 ! 13 ! 68 ! 63
! 53 ! 46 ! 49 ! 81 ! 8 ! 52 ! 100 ! 68 ! 80 ! 64 ! 34 ! 67 ! 19 ! 99 ! 34 ! 22 ! 70 ! 96 ! 11 ! 75
! 34 ! 65 ! 95 ! 42 ! 84 ! 27 ! 54 ! 66 ! 25 ! 86 ! 34 ! 63 ! 25 ! 100 ! 53 ! 45 ! 38 ! 79 ! 190
2
4
8
9
9
10
11
13
13
14
14
19
19
19
20
22
22
25
25
25
27
27
30
34

```



C:\Users\韦璐\Desktop\计算机程序设计\实验五\未命名18.exe

```

27
30
34
34
34
34
34
34
34
34
38
38
38
42
42
45
46
46
46
46
49
50
53
53
53
54
54
54

```

```
C:\Users\韦璐\Desktop\计算机程序设计\实验五\未命名18.exe
54
54
54
54
56
58
63
66
66
66
66
66
66
67
68
68
68
68
70
72
72
72
72
79
80
80
80
80
81
81
```

```
C:\Users\韦璐\Desktop\计算机程序设计\实验五\未命名18.exe
80
81
81
82
82
82
82
84
84
86
86
86
86
89
90
90
95
96
96
99
100
100
100
100
100
100
100
-----
Process exited after 0.6016 seconds with return value 0
请按任意键继续. . .
```

实验总结：

本次实验基本上框架已经有了，可能给出的函数里面有一点小小的陷阱所以需要自己进行调试。