实验 5　数组程序设计

一、实验目的

（1）加强对数组特点的了解，掌握一维数组的定义、初始化及其使用方法。

（2）掌握字符串的输入与输出方法，熟悉常用的字符串操作函数。

（3）掌握二维数组的定义、初始化及其使用方法。

（4）进一步熟悉数组应用程序的设计方法。

二、实验内容

1. 一维数组的输入与输出练习

修改例 5-1 的程序，使其在逆序输出数据时只输出其中的偶数。

附：例 5-1 源程序

```
#include<stdio.h>
int main()
{
    int i,a[10];                          /* 定义 a 数组 */
    printf("Input: ");
    for(i=0;i<10;i++)
       scanf("%d",&a[i]);                 /* 输入 a[0]、a[1]、a[2]、…、a[9] */
    printf("Output: ");
    for(i=9;i>=0;i--)
       printf("%d ",a[i]);                /* 输出 a[9]、a[8]、a[7]、…、a[0] */
    return 0;
}
```

2. 一维数组的初始化及应用练习

修改例 5-3 的程序，使 Fibonacci 数列的前 20 个数依次存储在 fib 数组的 fib[1]到 fib[20]这 20 个元素中。

附：例 5-3 源程序

```
#include<stdio.h>
int main()
{
    int fib[20]={1,1},i;                  /* fib 数组的初始化 */
    for(i=2;i<20;i++)
        fib[i]=fib[i-1]+fib[i-2];         /* 第 i 项为其紧邻的前两项之和 */
    for(i=0;i<20;i++)                     /* 控制输出每个数值 */
    {
        printf("%-10d",fib[i]);           /* 输出第 i 个数值 */
        if((i+1)%5==0)                    /* 每输出 5 个数之后换行 */
```

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


3. 字符串的输入与输出练习

修改例 5-4 的程序,将输出字符串的操作改为由 printf()函数实现。

附:例 5-4 源程序

```
#include <stdio.h>
#define N 100
int main()
{
    char str[N];                          /* 定义字符数组 str,用于存储字符串 */
    printf("String: ");
    gets(str);                            /* 输入字符串,存储到 str 数组中 */
    printf("Result: ");
    puts(str);                            /* 输出 str 数组中的字符串 */
    return 0;
}
```

4. 二维数组的练习

修改例 5-10 的程序,将数组的其他元素用 0 填充。

附:例 5-10 源程序

```
#include <stdio.h>
int main()
{
    int a[4][4] = {{1},{6,1},{8,7,1},{9,5,3,1}};  /* 部分元素的初始化 */
    int i,j;
    for(i = 0;i < 3;i++)
        for(j = i + 1;j < 4;j++)
            a[i][j] = a[j][i];                    /* 利用下三角元素填充上三角元素 */
    for(i = 0;i < 4;i++)                          /* 输出填充后的二维数组 a */
    {
        for(j = 0;j < 4;j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }
    return 0;
}
```

5. 一维数组的应用

修改例 5-11 的排序程序,实现对 N 个实数的降序排序。

附:例 5-11 源程序

```
#include <stdio.h>
#define N 10
int main()
{
    int a[N],i,j,temp;
    printf("Input data: ");
    for(i = 0;i < N;i++)                  /* 从键盘输入 N 个整数,存储在数组 a 中 */
        scanf("%d",&a[i]);
```

```
    for(i=1;i<N;i++)                    /* 进行趟数控制,共比较 N-1 趟 */
        for(j=0;j<N-i;j++)              /* 在每一趟中相邻数据两两比较 */
            if(a[j]>a[j+1])             /* 相邻元素前者大、后者小时将两元素值交换 */
            {temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;}
    printf("Result: ");
    for(i=0;i<N;i++)                    /* 输出排序结果 */
        printf("%d",a[i]);
    printf("\n");
    return 0;
}
```

6. 综合实验一

用一维数组作为存储结构,实现 Josephus 环报数游戏。Josephus 环报数游戏描述如下:有 n 个人围成一圈,从 1 开始顺序编号到 n,首先自 1 号开始顺时针从 1 报数,数到 m 者自动出列,然后自下一个人开始重新从 1 报数,数到 m 者仍然自动出列,以此类推,直到最后一个人出列为止。编写程序,输出这 n 个人出列的顺序。

7. 综合实验二

鞍点问题。在二维数组中若某一位置上的元素在该行中最大,而在该列中最小,则该元素即为该二维数组的一个鞍点。要求从键盘输入一个二维数组,当鞍点存在时把鞍点找出来。

三、实验指导

1. 一维数组的输入与输出练习

1) 程序分析

当输出存储在 a 数组中的数据时要对其每个元素加以判断,若是偶数,则将该元素输出。对任意元素 a[i],可用以下命令判断并输出。

```
if(a[i]%2==0) printf("%d",a[i])
```

逆序输出 a 数组中所有偶数的控制命令如下。

```
for(i=9;i>=0;i--)
    if(a[i]%2==0) printf("%d",a[i]);
```

2) 程序调试

在调试程序时,所用测试数据应包含以下情况。

(1) 在输入数据时偶数、奇数随机排列。

(2) 输入的数据都不是偶数。

(3) 输入的数据都是偶数。

(4) 输入数据时首、尾数据是偶数。

(5) 输入数据时首、尾数据不是偶数。

为提高测试效率,在定义 a 数组时可以使用符号常量 N,在调试程序时将 N 定义为一个较小的值。例如,可以用以下宏命令将 N 定义为 5。

```
#define N 5
```

2. 一维数组的初始化及应用练习

1) 程序分析

(1) 与例 5-3 的程序相比较,本实验中的 fib 数组定义与初始化发生变化。在例 5-3 的

程序中,Fibonacci数列的前20个数依次存储在fib数组的fib[0]到fib[19]这20个元素中,fib[0]和fib[1]分别被初始化为数列的前两个值。现要求将数列的前两个值存储在fib[1]和fib[2]中,其他值在fib数组中依次存储,fib[0]闲置不用。以下语句可实现数组的定义与初始化:

```
int fib[21] = {0,1,1}
```

因为fib[0]并未使用,初始化为任何数值都不会影响数列在fib数组中的存储。

(2) 注意fib数组的长度变化。新的fib数组的长度是21,用于存储Fibonacci数列的元素为fib[1]到fib[20]。

2) 参考程序

程序如下:

```
#include <stdio.h>
int main()
{
    int fib[21] = {0,1,1},i;              /* fib数组的初始化 */
    for(i = 3;i <= 20;i++)
        fib[i] = fib[i - 1] + fib[i - 2]; /* 第i项为其紧邻的前两项之和 */
    for(i = 1;i <= 20;i++)                /* 控制输出每个数值 */
    {
        printf("%-10d",fib[i]);          /* 输出第i个数值 */
        if(i % 5 == 0)                   /* 每输出5个数之后换行 */
            printf("\n");
    }
    return 0;
}
```

3. 字符串的输入与输出练习

1) 程序分析

以下两种方法都可输出存储在字符数组str中的字符串。

(1) 使用专门的字符串输出函数puts(),一般调用形式为puts(str)。

(2) 使用printf()函数,一般调用形式为printf("%s",str)。

本实验只需将程序中的puts(str)替换为printf("%s",str)即可。

2) 程序调试

(1) 运行程序,输入不含空格的字符串,观察并分析程序的执行结果。

(2) 运行程序,输入中间含有空格的字符串,观察并分析程序的执行结果。

(3) 运行程序,输入以空格开头的字符串,观察并分析程序执行结果。

(4) 运行程序,输入一个空字符串(直接按Enter键),观察并分析程序的执行结果。

3) 问题思考

向字符数组str输入字符串的常用方法有两种,一是使用专门的字符串输入函数gets(),一般调用形式为gets(str);二是使用scanf()函数,一般调用形式为scanf("%s",str)。请读者修改程序,用scanf()函数实现字符串的输入,然后使用上面的一组数据调试、运行程序,观察并分析程序的执行结果。

4. 二维数组的练习

1) 程序分析

在例5-10的程序中使用以下代码填充上三角中的各元素值:

```
for(i = 0;i < 3;i++)
    for(j = i + 1;j < 4;j++)
        a[i][j] = a[j][i];              /* 利用下三角元素填充上三角元素 */
```

用 0 填充上三角的各元素值相对简单，只需将 a[i][j]＝a[j][i]修改为 a[i][j]＝0 即可。

2）问题思考

以下程序实现的功能与本实验的要求相同，请读者注意比较这两个程序的异同。对于程序中 static 的作用将在第 6 章介绍。

```
#include < stdio.h >
int main()
{
    static int a[4][4] = {{1},{6,1},{8,7,1},{9,5,3,1}};          /* 二维数组的初始化 */
    int i,j;
    for(i = 0;i < 4;i++)                /* 输出二维数组 a */
    {
        for(j = 0;j < 4;j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }
    return 0;
}
```

5. 一维数组的应用

1）程序分析

（1）修改数组定义，将 a 数组定义为 float 型（或 double 型），数组的输入与输出格式也相应修改。

（2）在例 5-11 的程序中由以下代码实现排序过程。

```
for(i = 1;i < N;i++)                    /* 进行趟数控制，共比较 N - 1 趟 */
    for(j = 0;j < N - i;j++)            /* 在每趟中控制相邻数据两两比较 */
        if(a[j]> a[j + 1])              /* 相邻元素前大后小时将两元素值交换 */
        { temp = a[j]; a[j] = a[j + 1]; a[j + 1] = temp; }
```

（3）在上述代码中，加阴影的代码决定了排序结果为升序排序。若将其修改为 a[j]＜a[j＋1]，则实现降序排序。

2）程序调试

（1）将 N 定义为一个较小的值，以提高程序的调试效率。

（2）执行程序，输入一组无序的数值，观察程序执行结果。

（3）执行程序，输入一组由小到大排列的数值，观察程序的执行结果。

（4）执行程序，输入一组由大到小排列的数值，观察程序的执行结果。

6. 综合实验一（Josephus 环报数游戏）

1）编程分析

（1）将 n 个队员由 1 到 n 编号，并将编号存储到一维数组中。

（2）从首元素开始依次报数，当某元素报数到 m 时输出该元素值（队员出列），同时该元素值置为 0，然后从下一个元素开始重新报数；当报数到末元素后，再返回到数组头继续依次报数；在报数过程中，凡遇到元素值为 0 者（队员已出列）则跳过该元素。如此循环往

复,直到数组中的所有元素均为0时(所有队员已出列)报数过程结束。

(3) 设置flag标志,检测报数过程是否结束。在数组每趟报数之前先将flag置为0;在数组由前到后的报数过程中,凡有报数者(元素值不为0)则执行一次flag=1操作;若一趟报数结束后flag的值为0,则表示数组中的所有队员均已出列,报数过程结束,否则继续进行报数过程。

(4) 设置报数变量counter,其初值置为0。每检测到一个非0元素(队列中的一个队员报数)即执行counter++操作,当counter的值为m时队员出列(输出该元素值,该元素值为队员编号),然后counter复位为0,准备下一轮报数。

以下是n=10、m=3时报数过程的数组状态图示。图1-11所示为报数之前的数组状态;图1-12所示为第1趟报数后的数组状态,在该趟报数过程中3号、6号、9号出列,对应的数组元素被置为0;图1-13所示为第2趟报数后的数组状态,在该趟报数过程中2号、7号出列,对应的数组元素被置为0(在图中用阴影标识)。

1	2	3	4	5	6	7	8	9	10

图1-11 报数之前的数组状态

1	2	0	4	5	0	7	8	0	10

图1-12 第1趟报数后的数组状态

1	0	0	4	5	0	0	8	0	10

图1-13 第2趟报数后的数组状态

2) 参考程序

程序如下:

```
#include<stdio.h>
#define N 100
int main()
{
    int a[N],i,n,m,counter=0;
    int flag=1;                          /* 是否全部出列的标志 */
    scanf("%d%d",&n,&m);                 /* n个队员,报数到m者出列 */
    for(i=0;i<n;i++)                     /* 将队列的编号存储到一维数组 */
        a[i]=i+1;
    while(flag)                          /* 队列非空,则不断报数 */
    {
        flag=0;                          /* 数组每趟扫描之前置flag为0 */
        for(i=0;i<n;i++)
        {
            if(a[i]!=0)                  /* 遇到非0元素时进行以下处理 */
            {
                flag=1;                  /* 置队列非空标志为1 */
                counter++;               /* 报数 */
                if(counter==m)           /* 队列元素出列时进行以下处理 */
                {
                    printf("%d ",a[i]);  /* 出列 */
                    a[i]=0;              /* 出列后元素值置为0 */
                    counter=0;           /* 报数器置为0 */
```

```
                }
            }
        }
    }
    return 0;
}
```

3）程序调试

在调试程序时要注意所输入数据的含义与数据的输入格式。

（1）执行程序，输入一组满足 n≤0 的数据，观察并分析程序结果。

（2）执行程序，输入一组满足 m≤0 的数据，观察并分析程序结果。

（3）执行程序，输入一组满足 m＝1 的数据，观察并分析程序结果。

（4）执行程序，输入一组满足 m＝n 的数据，观察并分析程序结果。

（5）执行程序，输入一组满足 m 在[2，n－1]范围的数据，观察并分析程序结果。

（6）执行程序，输入一组满足 m＞n 的数据，观察并分析程序结果。

（7）执行程序，输入一组满足 n＞100 的数据，观察并分析程序结果。

7. 综合实验二（鞍点问题）

1）编程分析

（1）使用逐行扫描的方法在二维数组中查找鞍点。

（2）对于每一行，首先找出该行的最大值元素，然后看该元素在其所在的列上是否为最小值，若是，则找到一个鞍点。

（3）找到鞍点后输出元素值及其所在的行、列值。

2）参考程序

程序如下：

```
#include<stdio.h>
#define M 3
#define N 4
int main()
{
    int a[M][N],i,j,k;
    printf("请输入二维数组的数据:\n");
    for(i=0;i<M;i++)                    /* 为二维数组输入源数据 */
       for(j=0;j<N;j++)
          scanf("%d",&a[i][j]);
    for(i=0;i<M;i++)                    /* 按行扫描 */
    {
       k=0;                             /* k记录当前行中的最大值所在列号,初值为0 */
       for(j=1;j<N;j++)                 /* 在当前行(i行)中找最大值 */
          if(a[i][j]>a[i][k])
             k=j;                       /* k记录当前的较大值所在列号 */
       for(j=0;j<M;j++)                 /* 判断a[i][k]是否为k列的最小值 */
          if(a[j][k]<a[i][k])
             break;                     /* 若找到更小值,则a[i][k]不是鞍点 */
       if(j==M)                         /* 条件成立,则a[i][k]是鞍点 */
          printf("%d %d,%d\n",a[i][k],i,k) ;      /* 输出鞍点值及其所在的行、列号 */
    }
    return 0;
}
```

3) 程序调试

(1) 输入有鞍点的一组数据,观察并分析程序的运行结果。例如:

```
  9    80  215  40
 60   -60   89   1
210    -3  101  89
```

(2) 输入没有鞍点的一组数据,观察并分析程序的运行结果。例如:

```
  9    80  215  40
 60   -60  189   1
210    -3  101  89
```

4) 问题思考

上述参考程序查找鞍点的算法并不完善,例如当一行上有两个相同的最大值时,本程序只对第1个最大值进行鞍点判断,而忽略了对第2个最大值的鞍点判断。试完善程序,找出数组中的每个鞍点。

实验 6　函数程序设计

一、实验目的

(1) 通过本实验加深对函数的理解。

(2) 掌握简单变量作函数参数时函数的定义和调用方法。

(3) 掌握一维数组作函数参数时函数的定义和调用方法。

二、实验内容

1. 无返回值函数的定义及调用练习

利用 p_star()函数输出图案。在例 6-2 中定义了连续输出 n 个 * 字符的函数 p_star()。试编写一个程序,调用该函数输出如图 1-14 所示的 * 字符图案。

```
*
**
***
****
*****
****
***
**
*
```

图 1-14　“*”图案

附：p_star()函数源代码

```
void p_star(int n)
{
    int i;
    for(i = 1;i <= n;i++)
        putchar('*');
}
```

2. 有返回值函数的定义及调用练习

修改例 6-7 的程序,使其能够利用求两个数中较大数的函数 max()求得 4 个数的最大数。

附：例 6-7 源程序

```
#include <stdio.h>
int main()
{
    float max(float,float);                    /* 函数声明 */
    float a,b,c;
    printf("a,b,c: ");
    scanf("%f,%f,%f",&a,&b,&c);
    printf("max = %f\n",max(max(a,b),c));      /* 函数调用 */
    return 0;
}
float max(float x,float y)                     /* 定义求两个数中较大数的函数 */
{
    float m;
    m = x > y?x:y;
    return m;
}
```

3. 递归函数练习

参考例6-10,编写用递归方法求累加和的函数sum(),并调用它计算10!。

附:例6-10计算累加和$\sum_{i=1}^{n} i$的递归函数源代码

```
long sum(int n)
{
    if(n<0)
        return -1;
    else if(n==0)
        return 0;
    else if(n==1)
        return 1;
    else
        return sum(n-1)+n;
}
```

4. 数组元素作函数参数练习

按以下要求修改例6-12的程序。

(1) 修改判断素数函数prime(),使得k为素数时函数返回值为0,k不是素数时函数返回值为1。

(2) 在主函数中调用修改后的prime(),求一个自然数数组中的素数。

附:例6-12源程序

```
#define N 10
#include<stdio.h>
#include<math.h>
int main()
{
    int prime(int);                          /* 判断素数函数 */
    int i,natural[N];                        /* 定义整数数组 */
    printf("Data: ");
    for(i=0;i<N;i++)                         /* 建立整数数组 */
        scanf("%d",&natural[i]);
    printf("Result: ");
    for(i=0;i<N;i++)
        if(prime(natural[i]))                /* 调用prime()判断素数 */
            printf("%d ",natural[i]);        /* 输出素数 */
    printf("\n");
    return 0;
}
int prime(int k)                             /* 定义判断素数的函数 */
{
    int sk,i;
    sk=(int)sqrt(k);
    for(i=2;i<=sk;i++)
        if(k%i==0)return 0;                  /* 非素数时返回0 */
    return 1;                                /* 素数时返回1 */
}
```

5. 一维数组作函数参数练习

按以下要求修改例6-13的程序。