

云南大学数学与统计学实验教学中心

《高级语言程序设计》实验报告

课程名称：程序设计和算法语言	学期：2016~2017 学年上学期	成绩：
指导教师：赵越	学生姓名：刘鹏	学生学号：20151910042
实验名称：数组程序设计		
实验编号：No.05	实验日期：2018 年 8 月 16 日	实验学时：2
学院：数学与统计学院	专业：信息与计算科学	年级：2015 级

一、实验目的

1. 掌握数组的概念和使用方法。
2. 掌握数组初始化的方法。
3. 学会字符数组和字符串的应用。
4. 学会用数组名作函数的参数。
5. 掌握一维数组与二维数组的定义及其元素的引用方法。
6. 深刻体会数组与循环的关系。
7. 掌握利用一维数组和二维数组实现一些常用算法的编程技巧。
8. 进一步掌握动态调试的基本技能。

二、实验环境

Windows10 Pro Workstation 17134.165;

Cygwin GCC 编译器。

三、实验内容

3.1 有关概念

1. 只有静态数组和外部数组才能初始化。
2. 引用数组时，编译器对下标是否越界不作检查。如定义 `int a[5];` 在引用时出现 `a[5];` 不给出错信息，而是引 `a[4]` 下面一个单元的值。
3. 字符串放在字符数组中，一个字符串以 `'\0'` 结束，有一些字符串函数如 `strcpy`, `strcmp`, `strlen` 等可以方便进行字符串运算。
4. 如有如下定义：`char *str = "I love china";` 表示 `str` 是一个字符型指针变量，它的值是一个字符数据的地址。不要认为 `str` 是字符串变量，在其中存放一个字串 `"I love china"`。
5. 用数组名作函数实参时，传到形参的是数组的首地址。

3.2 题 1

定义三个数组

```
int a[5];  
int b[2][2];
```

```
char c[10];
```

分别在函数体外和函数体内对它们进行初始化，然后输出它们的值。在程序中再加一语句，输出 `a[5]`，`b[2][2]`，分析结果。对 `c` 数组改为用赋值语句给各元素赋初值：`c[0]~c[9]` 各元素分别为：'I', ' ', 'a', 'm', ' ', 'b', 'o', 'y'。然后用 `printf("%s", c)` 输出字符串，分析结果。

3.2.1 程序代码

```
1  /*
2  * filename: 5.1 print test.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i, x, y;
10     static int a[5] = {1, 1, 1, 1, 1};
11     static int b[2][2]={2, 2}, {2, 2}};
12     static char c[10] = {'c', 'c', 'c', 'c', 'c', 'c', 'c', 'c', 'c', 'c'};
13     for(i = 0; i < 5; i++)
14         printf(" %5d ", a[i]);
15     printf("\n");
16
17     for(x = 0; x < 2; x++)
18         for(y = 0; y < 2; y++)
19             printf(" %5d ", b[x][y]);
20     printf("\n");
21
22     for(i = 0; i < 10; i++)
23         printf(" %3c ", c[i]);
24     printf("\n");
25
26     printf("a[5] = %d\n", a[5]);
27     printf("b[2][2] = %c\n", b[2][2]);
28
29     static char c2[10] = "I am a boy";
30     printf("%s\n", c2);
31
32     for(int i=0; i < 20; i++) {
33         printf("%d ", a[i]);
34         if((i+1) % 5 == 0) {
35             printf("\n");
36         }
37     }
38     return 0;
39 }
```

3.2.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.1\ print\ test.c -o 5.1.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.1.exe
  1   1   1   1   1
  2   2   2   2   2
 c   c   c   c   c   c   c   c   c   c
a[5] = 0
b[2][2] = c
I am a boy
1 1 1 1 1
0 0 0 2 2
2 2 1667457891 1667457891 25443
0 1835081801 1646289184 31087 0
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.3 搜索

有一数组，内放10个整数，要求找出最小的数和它的下标。然后把它和数组中最前面的元素对换位置。编写程序，上机运行，并记录下结果。提示：数组的10个元素可用输入函数 `scanf()` 通过键盘输入进去，找出数组中最小的元素可通过循环语句和条件语句来实现。

设 `min` 是存放数组中最小元素的变量，`array[k]` 为一个暂存单元。实现最前面的元素与最小元素对换可通过下面语句实现：

```

array[k]=array[0];
array[0]=min;

```

3.3.1 程序代码

```

1  /*
2  * filename: 5.2 search and insert.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i;
10     int array[10];
11     int min, k = 0;
12
13     printf("Please input 10 data\n");
14     for(i = 0; i < 10; i++)
15         scanf("%d", &array[i]);
16
17     min = array[0];
18     printf("\n");
19     for(i = 1; i < 10; i++)
20         if(min > array[i]) {
21             min = array[i];
22             k = i;

```

```

23     }
24     array[k] = array[0];
25     array[0] = min;
26     printf("After exchange:\n");
27     for(i = 0; i < 10; i++)
28         printf("%5d", array[i]);
29     printf("\n");
30     printf("k = %d\t min = %d\n", k + 1, min);
31     return 0;
32 }

```

3.3.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.2\ search\ and\ insert.c -o 5.2.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.2.exe
Please input 10 data
1 0 5 3 12 -9 8 5 2 1

After exchange:
-9  0  5  3 12  1  8  5  2  1
k = 6    min = -9

Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.4 有序插入

在一个已排好序的数列中（由小到大）再插入一个数，要求仍然有序。编程并上机运行。提示：编程时应考虑到插入的数的各种可能性（比原有所有的数大；比原有所有的数小；在最大数和最小数之间）。

3.4.1 程序代码

```

1  /*
2  * filename: 5.3 sort and insert.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  int main() {
8      int i, n;
9      float a, x[20], y[21];
10     printf("n = ");
11     scanf("%d", &n);
12     printf("sorted array: ");
13     for(i = 0; i < n; i++)
14         scanf("%f", &x[i]);
15     printf("Insert value = ");
16     scanf("%f", &a);
17     i = 0;
18     while(a > x[i] && i < n) {
19         y[i] = x[i];
20         i++;

```

```

21     }
22     y[i] = a;
23     for(i = i+1; i < n + 1; i++)
24         y[i] = x[i-1];
25     printf("\n");
26     for(i=0;i<n+1;i++) {
27         printf("%8.2f", y[i]);
28     }
29     return 0;
30 }

```

3.4.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.3\ sort\ and\ insert.c -o 5.3.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.3.exe
n = 6
sorted array: 1 2 3 4 5 6
Insert value = 3.5

    1.00    2.00    3.00    3.50    4.00    5.00    6.00
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.5 按姓名排序

编写一程序，一班级有 n 名学生要求按他们姓名的顺序排列（按汉语拼音的字母顺序从小到大），并按序输出。

3.5.1 程序代码

```

1  /*
2  * filename: 5.4 dictionary sort.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  #include <string.h>
8
9  void strup(char str[]) {
10     int i;
11     for(i=0; str[i] != '\0'; i++)
12         if(str[i] >= 'a' && str[i] <= 'z')
13             str[i] = str[i] + 'A' - 'a';
14 }
15
16 int main() {
17     char name[20][20];
18     int i, j, n;
19     printf("Number of students: ");
20     scanf("%d", &n);

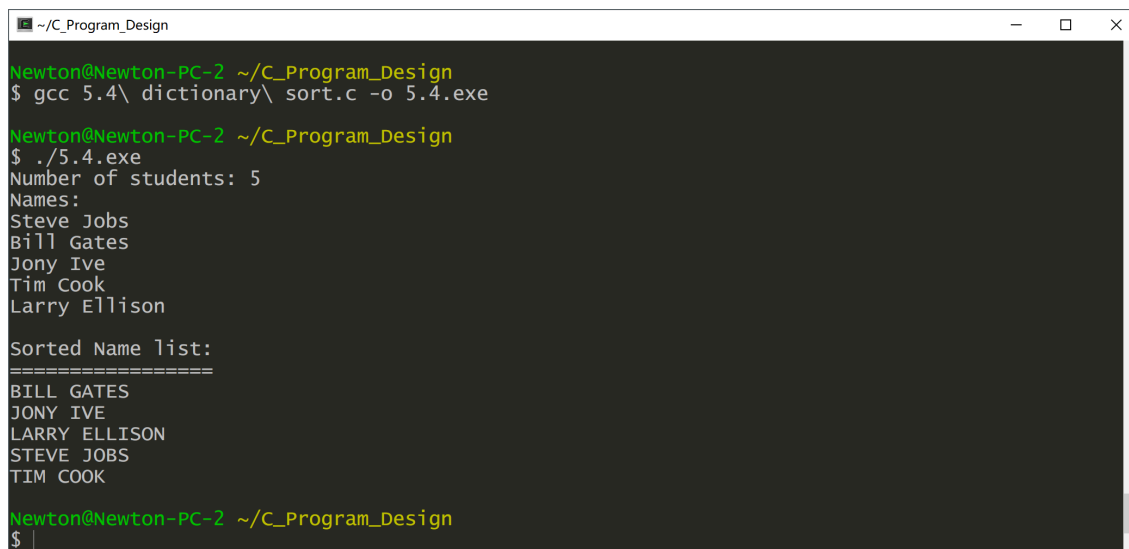
```

```

21     n += 1;
22     printf("Names:\n");
23     for(i = 0; i < n; i++) {
24         gets(name[i]);
25         strup(name[i]);
26     }
27
28     char tmp[20];
29
30     // small -> big, bubble sort
31     for (i = 0; i < n - 1; i++) {
32         for (j = i; j < n; j++) {
33             if(strcmp(name[i], name[j]) > 0) {
34                 strcpy(tmp, name[i]);
35                 strcpy(name[i], name[j]);
36                 strcpy(name[j], tmp);
37             }
38         }
39     }
40
41     printf("\nSorted Name list:\n=====");
42     for(i = 0; i < n; i++)
43         printf("%s\n", name[i]);
44     return 0;
45 }

```

3.5.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.4\ dictionary\ sort.c -o 5.4.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.4.exe
Number of students: 5
Names:
Steve Jobs
Bill Gates
Jony Ive
Tim Cook
Larry Ellison

Sorted Name list:
=====
BILL GATES
JONY IVE
LARRY ELLISON
STEVE JOBS
TIM COOK
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.6 * 打印魔方阵

所谓魔方阵是指，它的每行每一列的和与对角线之和均相等。例如，三阶魔方阵为

8	1	6
3	5	7
4	9	2

要求打印由1到 n^2 的自然数构成的魔方阵。

提示：魔方阵中各数排列规律为：将“1”放在第一行中间一列；从“2”开始直到 $n \times n$ ，各数依次按下列规则存放：每一个数存放的行比前一个数的行数减1，列数加1；如果上一数的行数为1，则下一个数的行数应为 n （指最下一行）；当上一个数的列数为 n 时，下一个数的列数应为1，行数减1。

如果按上面的规则，确定的位置上已有数，或上一个数是第1行第 n 列时，则把下一个数放在上一个数的下面。

3.6.1 程序代码

```

1  /*
2  * filename: 5.5 print.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  #define N 10
9
10 int main() {
11     int n, i, j, a, b, num, u, v;
12     int c[10][10];
13     printf("input an odd number: ");
14     scanf("%d", &n);
15     num = n * n;
16     for(i = 0; i < n; i++)
17         for(j = 0; j < n; j++)
18             c[i][j] = 0;
19     c[0][n/2] = 1;
20     a = 0;
21     b = n/2;
22     for(i = 2; i <= num; i++) {
23         if(a)
24             u = a - 1;
25         else
26             u = n - 1;
27
28         v = (b + 1) % n;
29         if(!c[u][v])
30             c[u][v] = i;
31         else {
32             u = (a + 1) % n;
33             v = b;
34             c[u][v] = i;
35         }
36         a = u;
37         b = v;
38     }
39     for(i = 0; i < n; i++) {

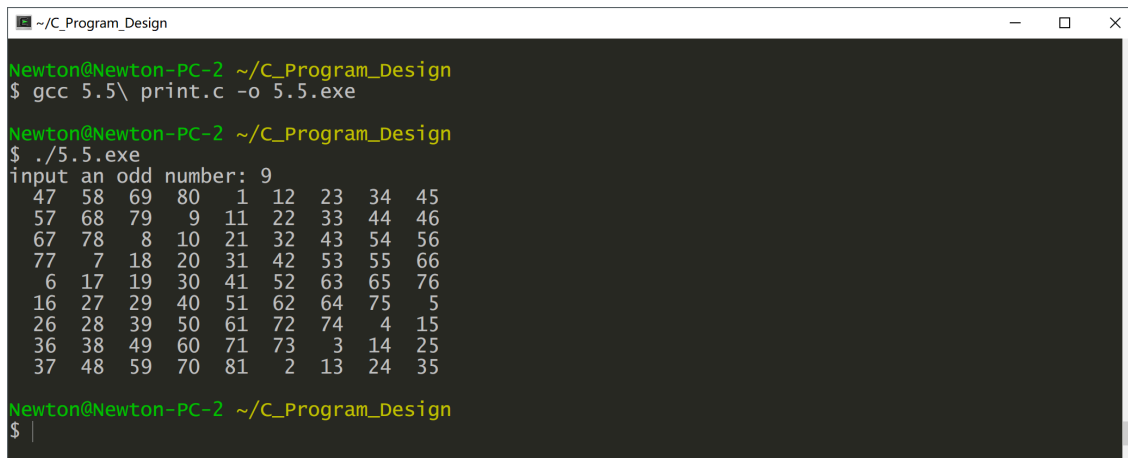
```

```

40     for(j = 0; j < n; j++)
41         printf("%4d", c[i][j]);
42         printf("\n");
43     }
44     return 0;
45 }

```

3.6.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.5\ print.c -o 5.5.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.5.exe
input an odd number: 9
 47 58 69 80  1 12 23 34 45
 57 68 79  9 11 22 33 44 46
 67 78  8 10 21 32 43 54 56
 77  7 18 20 31 42 53 55 66
  6 17 19 30 41 52 63 65 76
16 27 29 40 51 62 64 75  5
26 28 39 50 61 72 74  4 15
36 38 49 60 71 73  3 14 25
37 48 59 70 81  2 13 24 35

Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.7 元素移位

用移位法将数组 a 中的最后一个数移到最前面，其余数依次往后移动一个位置。

请按以下步骤实习和思考：

- ①分析程序及其特性。
- ②上机运行程序，查看运行结果是否正确？
- ③用动态跟踪查找错误原因，以此分析并找出错误原因。
- ④改正错误后重新运行程序，直到结果正确为此。
- ⑤如果要用三次循环移位来实现将最后三个数移到前面，其余数依次往后移三个位置，则程序应该如何修改？

3.7.1 程序代码

```

1  /*
2  * filename: 5.6 move.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i, t;
10     int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
11     t = a[9];
12     for(i = 9; i > 0; i--)

```



```

13     a[i] = a[i-1];
14     a[0] = t;
15     printf("\n");
16     for(i = 0; i < 10; i++)
17         printf("%4d", a[i]);
18     return 0;
19 }

```

3.7.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/_C_Program_Design
$ gcc 5.6\ move.c -o 5.6.exe
Newton@Newton-PC-2 ~/_C_Program_Design
$ ./5.6.exe
 9  0  1  2  3  4  5  6  7  8
Newton@Newton-PC-2 ~/_C_Program_Design
$

```

3.8 成绩排序

输入 n 个学生的单科成绩，然后从高到低的顺序排序后输出。

- ①分析程序及其特性。
- ②上机编译程序，程序是否有语法错误？应如何修改？（数组 a 的长度可比 n 大些），改正错误后重新编译和运行程序，直到结果正确为此。
- ③你对选择排序算法的实现过程是否清楚了？若不清楚，请用动态跟踪的方法观察其实现过程，操作如下：首先将光标移至 `if` 语句行上，按 `F4`，接着输入数据，当绿条第停留在 `if` 语句行时，用 `Ctrl-F7` 操作将 a 数组的内容显示出来，不断按 `F4`，观察 a 数组值的变化情况，以此分析和领会算法的实现过程。
- ④输入冒泡排序程序，用动态跟踪观察其实现过程。
- ⑤如果要用三次循环移位来实现将最后三个数移到前面，其余数依次往后移三个位置，则程序应该如何修改？

3.8.1 程序代码

```

1  /*
2  * filename: 5.7 sort.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  #define N 10
9
10 int main() {
11     int i, j, t, n, a[N];
12     printf("n = ");
13     scanf("%d", &n);
14     printf("input n numbers: \n");
15     for (i = 0; i < n; i++)

```

```

16     scanf("%d",&a[i]);
17     for(i = 0; i < n - 1; i++)
18         for(j = i + 1; j < n; j++)
19             if(a[i] < a[j]) {
20                 t = a[i];
21                 a[i] = a[j];
22                 a[j] = t;
23             }
24     printf("the sorted numbers:\n");
25     for(i = 0; i < n; i++)
26         printf("%4d", a[i]);
27     return 0;
28 }

```

3.8.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.7\ sort.c -o 5.7.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.7.exe
n = 5
input n numbers:
5 88 96 255 1
the sorted numbers:
255 96 88 5 1
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.9 矩阵操作

将矩阵 A[4][5] 中值（行中所有数的和）为最大的那一行元素与首行元素对换。

具体要求如下：

- ① 矩阵 A 的数值从键盘输入。（可以采取固定内容实例）
- ② 以矩阵的形式输出对换后的矩阵。

3.9.1 程序代码

```

1  /*
2  * filename: 5.8 matrix.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i, j;
10     int a[4][5] = {{1, 1, 1, 1, 1}, {2, 2, 2, 2, 2}, {3, 3, 3, 3, 3}, {4, 4, 4, 4, 4}};
11
12     int d[4] = {0};
13
14     for (i = 0; i < 4; i++) {
15         for (j = 0; j < 5; j++) {

```

```

16     d[i] += a[i][j];
17 }
18 }
19
20 int max_index;
21 max_index = (d[0] > d[1])? 1:2;
22 max_index = (max_index > d[2])? max_index:3;
23 max_index = (max_index > d[3])? max_index:4;
24
25 int tmp;
26
27 if (max_index != 1) {
28     for (i = 0; i < 5; i++) {
29         tmp = a[0][i];
30         a[0][i] = a[max_index-1][i];
31         a[max_index-1][i] = tmp;
32     }
33 }
34
35 for (i = 0; i < 4; i++) {
36     for (j = 0; j < 5; j++) {
37         printf("%4d", a[i][j]);
38     }
39     printf("\n");
40 }
41
42 return 0;
43 }

```

3.9.2 运行结果

```

~/_C_Program_Design
Newton@Newton-PC-2 ~/_C_Program_Design
$ gcc 5.8\ matrix.c -o 5.8.exe

Newton@Newton-PC-2 ~/_C_Program_Design
$ ./5.8.exe
 4  4  4  4  4
 2  2  2  2  2
 3  3  3  3  3
 1  1  1  1  1

Newton@Newton-PC-2 ~/_C_Program_Design
$

```

3.10

17 个人围坐一圈，顺序编号为 1, 2, 3, ..., 17。现在从第一个人开始数起，每数到 7 时，这个人就从圈里出来，再从下一个数重新开始数 1, 2, ..., 7，数到第 7 的这个人也从圈里出来，直到全部 17 个人从圈里出来为此。例如，前面站出来的 4 个人是 7, 14, 4 和 12。编程输出从圈里出来的人的顺序。

3.10.1 程序代码

```

1  /*
2  * filename: 5.9 josephus.c

```

```
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i, j = 0;
10     int a[17]; // 0~16 == 1~17
11     for (i = 0; i < 17; i++) {
12         a[i] = 1;
13     }
14
15     int order[17];
16
17     int count = 0;
18     int left = 16;
19     int next = 0;
20
21     while (left >= 0) {
22         while (count < 7) {
23             if (a[next] == 0) {
24                 next = (next + 1) % 17;
25             }
26             else {
27                 if (count < 6)
28                     next = (next + 1) % 17;
29                 count += 1;
30             }
31         }
32         order[j] = next;
33         a[next] = 0;
34         next = (next + 1) % 17;
35         count = 0;
36         left--;
37         j++;
38     }
39
40     for (i = 0; i < 17; i++) {
41         printf("%4d", order[i]+1);
42     }
43
44     return 0;
45 }
```

3.10.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.9\ josephus.c -o 5.9.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.9.exe
7 14 4 12 3 13 6 17 11 9 8 10 16 5 15 1 2

Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.11 程序排错

下列有关数组的说明或语句是否存在错误？

```

int a[3,4]; a[3]=a[1*2];           // 错误, 初始化错误
int a[3]={10,8,3,4}                // 错误, 超界初始化
float a[4];a[0]=15;a[4]=100;       // 错误, 超界
int I,j,k[3]={3};                  // 正确
char ch[]={};                       // 正确

```

3.12 输出杨辉三角

写一个输出 10 阶杨辉三角形的程序。

3.12.1 程序代码

```

1  /*
2  * filename: 5.10 pascal triangle.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  #define N 10
9
10 int main() {
11     int a[N][N], i, j;
12     for(i = 1; i < N; i++) {
13         a[i][1] = 1;
14         a[i][i] = 1;
15     }
16     for(i = 3; i < N; i++)
17         for(j = 2; j <= i - 1; j++)
18             a[i][j] = a[i-1][j-1] + a[i-1][j];
19     for(i = 1; i < N; i++) {
20         for(j = 1; j < 30 - 2 * i; j++)
21             printf(" ");
22         for(j = 1; j <= i; j++)
23             printf("%-4d",a[i][j]);
24         printf("\n");
25     }
26 }

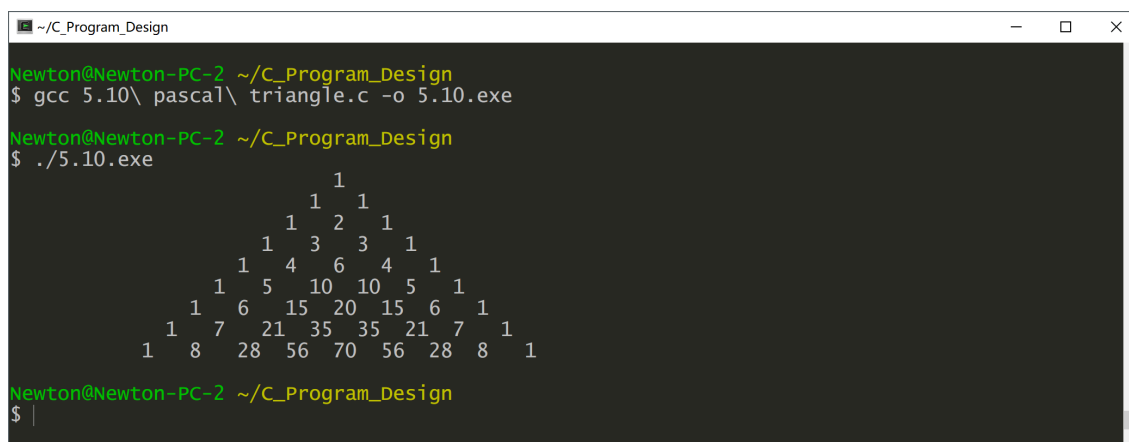
```

```

27     return 0;
28 }

```

3.12.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.10\ pascal\ triangle.c -o 5.10.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.10.exe
      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
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.13 矩阵转置

从键盘输入 $N \times N$ 的矩阵（ N 可以定义为 5），输出此矩阵和转置后的矩阵。

3.13.1 程序代码

```

1  /*
2  * filename: 5.11 matrix t.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  #define N 3
9
10 int main() {
11     int a[N][N], b[N][N], i, j;
12     for(i=0; i <= N-1; i++) {
13         for(j = 0; j <= N-1; j++)
14             scanf("%d", &a[i][j]);
15     }
16     printf("Matrix A is: \n");
17     for(i = 0; i <= N-1; i++) {
18         for(j = 0; j <= N-1; j++) {
19             printf("%5d", a[i][j]);
20             b[j][i] = a[i][j];
21         }
22         printf("\n");
23     }
24     printf("Matrix A^T is:\n");
25     for(j = 0; j <= N-1; j++) {
26         for(i = 0; i <= N-1; i++)
27             printf("%5d", b[j][i]);

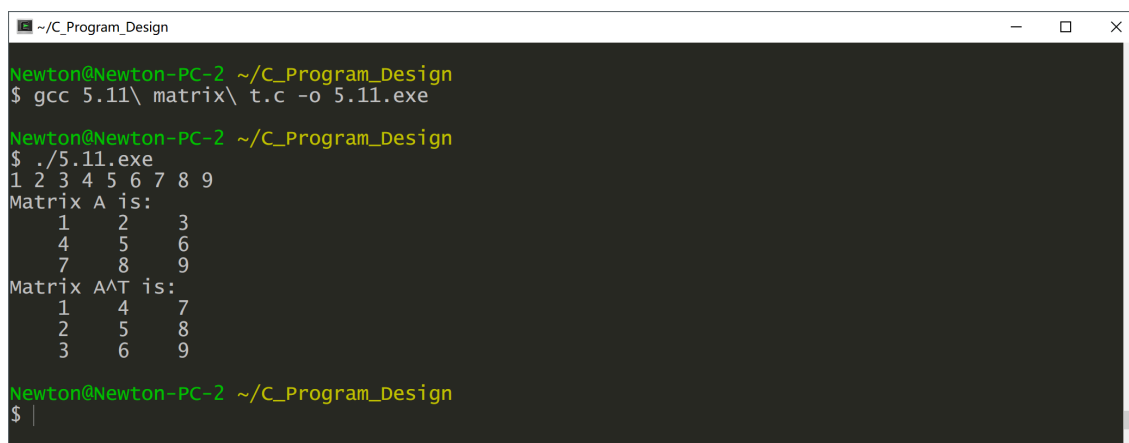
```

```

28     printf("\n");
29 }
30 return 0;
31 }

```

3.13.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.11\ matrix\ t.c -o 5.11.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.11.exe
1 2 3 4 5 6 7 8 9
Matrix A is:
  1  2  3
  4  5  6
  7  8  9
Matrix A^T is:
  1  4  7
  2  5  8
  3  6  9

Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.14 矩阵鞍点

从键盘输入 $m \times n$ 行列式，并输出此行列式；然后求所有的鞍点（某元素若是本行元素中的最大者，同时又是本列元素中最小者，则此元素称为鞍点）。最后输出这些鞍点及其对应坐标值。（若无鞍点，则显示无鞍点信息）。

3.14.1 程序代码

```

1  /*
2  * filename: 5.12 saddle point.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int a[3][4] = {
10         {51, 299, 1024, 33},
11         {2888, 10, 10000, 0},
12         {52, 10, 30000, 0}
13     };
14
15
16     int i, j, k;
17     for (i = 0; i < 3; i++) {
18         for (j = 0; j < 4; j++) {
19             printf("%6d", a[i][j]);
20         }
21         printf("\n");
22     }
23     printf("\n");

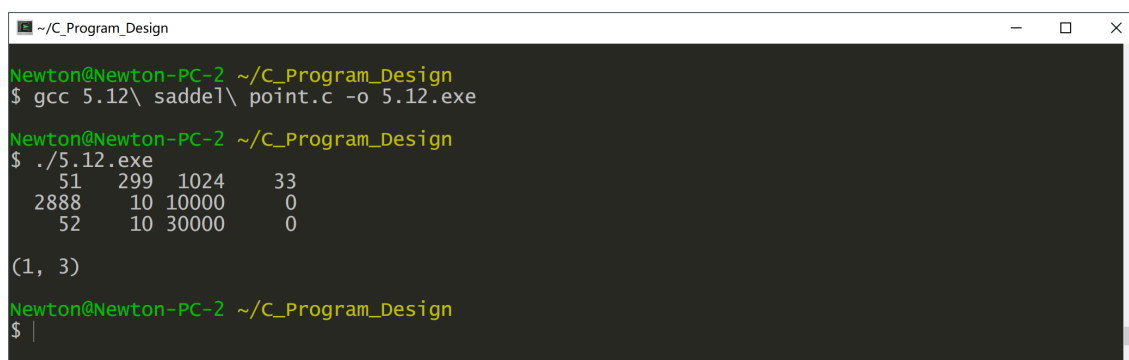
```

```

24
25     int g1 = 1;
26     int g2 = 1;
27
28     for (i = 0; i < 3; i++) {
29         for (j = 0; j < 4; j++) {
30             for (k = 0; k < 3; k++) {
31                 if (a[k][j] < a[i][j]) {
32                     g1 = 0;
33                 }
34             }
35             for (k = 0; k < 4; k++) {
36                 if (a[i][k] > a[i][j]) {
37                     g2 = 0;
38                 }
39             }
40             if(g1 * g2 == 1)
41                 printf("(%d, %d)\n", i+1, j+1);
42             g1 = g2 = 1;
43         }
44     }
45
46     return 0;
47 }

```

3.14.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.12\ saddle\ point.c -o 5.12.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.12.exe
51    299    1024    33
2888   10  10000    0
52    10  30000    0

(1, 3)

Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.15 回文字符串判断

从键盘上输入一字符串，并判断是否形成回文（即正序和逆序一样，如“abcd dcba”）。

3.15.1 程序代码

```

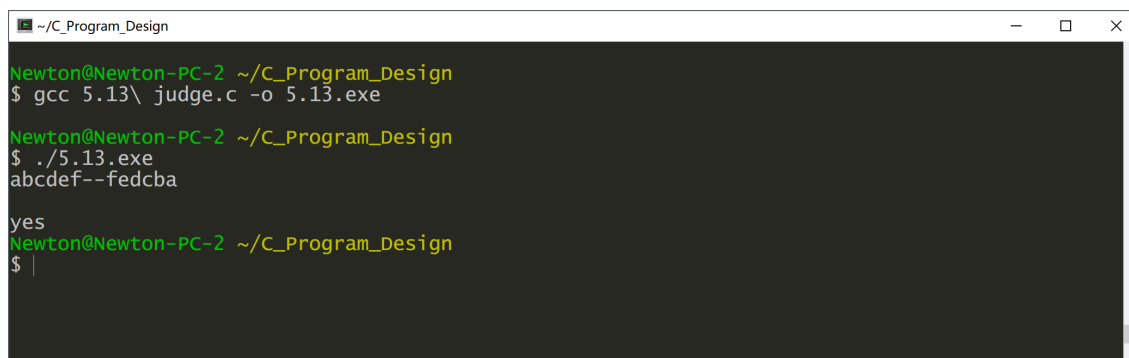
1  /*
2  * filename: 5.13 judge.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  #include <string.h>
8

```



```
9  int main() {
10     char a[100];
11     gets(a);
12     if (strlen(a) % 2 == 1) {
13         printf("not 1");
14         return 0;
15     }
16
17     int n = strlen(a) / 2;
18     int i, j = 0;
19     char b[50] = "";
20     puts(b);
21     for (i = 2*n - 1; i >= n; i--) {
22         b[j++] = a[i];
23     }
24
25     a[n] = '\0';
26
27     if (strcmp(a, b) == 0) {
28         printf("yes");
29     }
30     return 0;
31 }
32 }
```

3.15.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/_C_Program_Design
$ gcc 5.13\ judge.c -o 5.13.exe
Newton@Newton-PC-2 ~/_C_Program_Design
$ ./5.13.exe
abcdef--fedcba
yes
Newton@Newton-PC-2 ~/_C_Program_Design
$ |
```

3.16 字符串修改

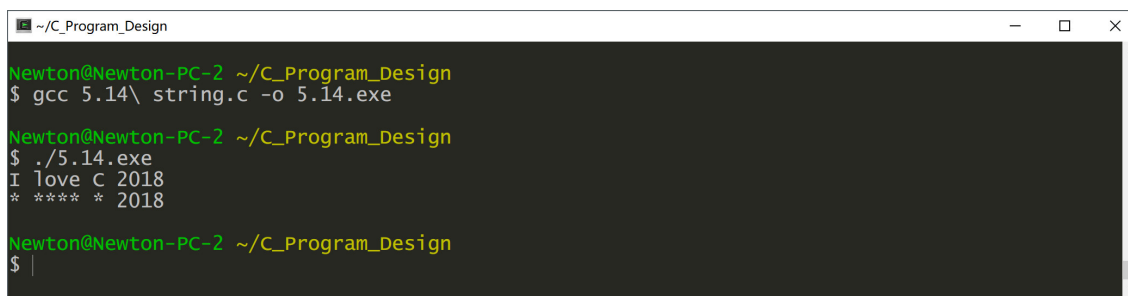
按字符数组输入字符，将其中的英文字母都改成‘*’，然后按字符串形式输出。

3.16.1 程序代码

```
1  /*
2   * filename: 5.14 string.c
3   * property: exercise
4   */
5
6  #include <stdio.h>
7  #include <string.h>
8
```

```
9  int main() {
10     char a[100];
11     gets(a);
12
13     int i;
14     for (i = 0; i < strlen(a); i++) {
15         if (a[i] > 'a' && a[i] < 'z' || a[i] > 'A' && a[i] < 'Z') {
16             a[i] = '*';
17         }
18     }
19     puts(a);
20 }
```

3.16.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 5.14\ string.c -o 5.14.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./5.14.exe
I love C 2018
*  *  *  *  * 2018
Newton@Newton-PC-2 ~/C_Program_Design
$
```

四、实验总结

数组作为指针的前身，本身具有很多优越性，比如大小固定，而且申请释放不需要自己管理。大小固定既是一种优点又是一种缺点，在后来的指针程序设计中，很多语句来得不如用数组轻松，但是在编写灵活性强的结构时，数组就力所不及了。本实验中的数组关注 `string.h` 给出的几个标准函数。

本次实验，集中主要精力，在以前版本的基础上，对文档结构进行了重整，看起来自然了很多，目录也规范了很多。有关编程的规范性问题，参考林锐高质量 C/C++ 编程指南的第一版[3]。

五、参考文献

1. Stevens, W.R. and S.A. Rago, *UNIX 环境高级编程*. 2nd ed. 2005, 北京: 人民邮电出版社.
2. Hahn, H., *Harley Hahn's Guide to Unix and Linux*. 2009, New York: McGraw-Hill.
3. 林锐, *高质量 C++/C 编程指南*. 1.0 ed. 2001.

六、教师评语