二维数组与字符串数组

2022 年 8 月 16 日

1. 有一篇文章，共有若干行文字，每行有若干个字符。要求分别统计出其中英文大写字母、小写字母、数字、空格以及其它字符的个数。每行文字都以字符串的形态存储。文章的数据声明和定义如下。

```
const char PASSAGE[][1024] = {
        "Once upon a time, there was a wolf living on a grassland in west China.",
        "It loved eating sheep.",
        "It often drank water at a river.",
        "There were also 25 houses there."};
```

```
#include <stdio.h>
#include <string.h>
const char PASSAGE[][1024] = {
                    "Once upon a time, there was a wolf living on a grassland in
                        west China.",
                    "It loved eating sheep.",
                    "It often drank water at a river.",
                    "There were also 25 houses there."};

int main()
{
    // 一开始还未开始扫描字符串，全部初始化为0
    int upper_case_letter_num = 0, lower_case_letter_num = 0;
    int digit_num = 0, space_num = 0, other_num = 0;
    for(int i = 0; i < 4; i++)
    {
        // 以下PASSAGE[i]表示第i+1行
        for(int j = 0; j < strlen(PASSAGE[i]); j++)
```

```
        {
            char ch = PASSAGE[i][j];
            if(ch >= 'A' && ch <= 'Z') // 大写字母
            {
                upper_case_letter_num++;
            }
            else if(ch >= 'a' && ch <= 'z') // 小写字母
            {
                lower_case_letter_num++;
            }
            else if(ch >= '0' && ch <= '9') // 数字
            {
                digit_num++;
            }
            else if(ch == ' ') // 空格符
            {
                space_num++;
            }
            else
            {
                other_num++;
            }
        }
    }
    printf("big letter num: %d\n", upper_case_letter_num);
    printf("small letter num: %d\n", lower_case_letter_num);
    printf("digit num: %d\n", digit_num);
    printf("space num: %d\n", space_num);
    printf("other num: %d\n", other_num);
    return 0;
}
```

2. 输出杨辉三角前十行.

```
#include <stdio.h>

int main()
{
    int chinese_triangle[10][10];
```

```
6
7       chinese_triangle[0][0] = 1;
8       chinese_triangle[1][0] = 1;
9       chinese_triangle[1][1] = 1;
10
11      for(int i = 2; i < 10; i++)
12      {
13          chinese_triangle[i][0] = 1;
14          chinese_triangle[i][i] = 1;
15          for(int j = 1; j < i; j++)
16          {
17              // 等于肩上两个数相加
18              chinese_triangle[i][j] = chinese_triangle[i-1][j-1] +
                    chinese_triangle[i-1][j];
19          }
20      }
21
22      for(int i = 0; i < 10; i++)
23      {
24          for(int j = 0; j <= i; j++)
25          {
26              printf("%d\t", chinese_triangle[i][j]);
27          }
28          printf("\n");
29      }
30      return 0;
31  }
```

3. 定义两个 5×4 的二维矩阵 A 和 B，分别对这两个列表输入数据，求 $A+B$ 和 $A-B$ 的值。$A+B$ 就是把所有的 a_{ij} 和 b_{ij} 对应相加，并且把所得的和保存在另一矩阵的第 i 行第 j 列。减法是类似的。

```
1   #include <stdio.h>
2   int main()
3   {
4       int a[5][4]; // 定义矩阵A
5       int b[5][4]; // 定义矩阵B
6       int sum_mtx[5][4]; // 定义和矩阵
7       int diff_mtx[5][4]; // 定义差矩阵
```

```
8
9       printf("Input the first 5*4 matrix:\n");
10      for(int i = 0; i < 5; i++)
11      {
12          printf("Input the %d-th number:\n", i + 1);
13          for(int j = 0; j < 4; j++)
14          {
15              scanf("%d", &a[i][j]); // 输入a[i][j]的值
16          }
17      }
18
19      printf("Input the second 5*4 matrix:\n");
20      for(int i = 0; i < 5; i++)
21      {
22          printf("Input the %d-th number:\n", i + 1);
23          for(int j = 0; j < 4; j++)
24          {
25              scanf("%d", &b[i][j]); // 输入b[i][j]的值
26          }
27      }
28
29      printf("The first 5*4 matrix:\n");
30      for(int i = 0; i < 5; i++)
31      {
32          for(int j = 0; j < 4; j++)
33          {
34              printf("%d\t", a[i][j]);
35          }
36          printf("\n");
37      }
38
39      printf("The second 5*4 matrix:\n");
40      for(int i = 0; i < 5; i++)
41      {
42          for(int j = 0; j < 4; j++)
43          {
44              printf("%d\t", b[i][j]);
45          }
46          printf("\n");
```

```
47      }
48
49      // 求和
50      for(int i = 0; i < 5; i++)
51      {
52          for(int j = 0; j < 4; j++)
53          {
54              sum_mtx[i][j] = a[i][j] + b[i][j];
55          }
56      }
57
58      // 求差
59      for(int i = 0; i < 5; i++)
60      {
61          for(int j = 0; j < 4; j++)
62          {
63              diff_mtx[i][j] = a[i][j] - b[i][j];
64          }
65      }
66
67      printf("The sum matrix:\n");
68      for(int i = 0; i < 5; i++)
69      {
70          for(int j = 0; j < 4; j++)
71          {
72              printf("%d\t", sum_mtx[i][j]);
73          }
74          printf("\n");
75      }
76
77      printf("The diff matrix:\n");
78      for(int i = 0; i < 5; i++)
79      {
80          for(int j = 0; j < 4; j++)
81          {
82              printf("%d\t", diff_mtx[i][j]);
83          }
84          printf("\n");
85      }
```

```
    return 0;
}
```

4. 编制程序，将 $N \times N$ 的矩阵转置。矩阵 $A = (a_{ij})_{m \times n}$ 的转置就是把所有的 a_{ij} 和相应的 a_{ji} 进行对换。设 $N = 4$。

```c
#include <stdio.h>
int main()
{
    int a[4][4];

    printf("Input a 4*4 matrix:\n");
    for(int i = 0; i < 4; i++)
    {
        printf("Input the %d-th row:\n", i + 1);
        for(int j = 0; j < 4; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }

    printf("The input matrix:");
    printf("The transposed matrix is:\n");
    for(int i = 0; i < 4; i++)
    {
        for(int j = 0; j < 4; j++)
        {
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }

    // 转置
    int b[4][4];
    for(int i = 0; i < 4; i++)
    {
        for(int j = 0; j < 4; j++)
        {
            b[i][j] = a[j][i]; // 复制到恰当位置
```

```
        }
    }

    printf("The transposed matrix is:\n");
    for(int i = 0; i < 4; i++)
    {
        for(int j = 0; j < 4; j++)
        {
            printf("%d\t", b[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

5. 有一个用户输入的 5×4 的二维矩阵，找出其中最大和最小元素，并指出它们所在的行和列。若存在多个最大或者最小元素，则只返回第一个所在的行和列。先按行看，再按列看。

```
#include <stdio.h>
int main()
{
    int a[5][4];
    printf("Input elements for a 5*4 matrix:\n");
    for(int i = 0; i < 5; i++)
    {
        printf("Input the %d-th row:\n", i + 1);
        for(int j = 0; j < 4; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }

    printf("The input matrix is:\n");
    for(int i = 0; i < 5; i++)
    {
        for(int j = 0; j < 4; j++)
        {
            printf("%d\t", a[i][j]);
```

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

    // 先把最大和最小值的行标和列标初始化为0
    int row_for_maxi = 0, column_for_maxi = 0;
    int row_for_mini = 0, column_for_mini = 0;

    for(int i = 0; i < 5; i++)
    {
        for(int j = 0; j < 4; j++)
        {
            if(a[i][j] > a[row_for_maxi][column_for_maxi]) // 找到更大的
            {
                // 更新行标和列标
                row_for_maxi = i;
                column_for_maxi = j;
            }

            if(a[i][j] < a[row_for_mini][column_for_mini]) // 找到更小的
            {
                // 更新行标和列标
                row_for_mini = i;
                column_for_mini = j;
            }
        }
    }

    printf("The first maxi is at (%d, %d)\n", row_for_maxi, column_for_maxi);
    printf("The first mini is at (%d, %d)\n", row_for_mini, column_for_mini);
}
```

6. 求 M 行 N 列矩阵中各行最大值中最小的数。设 $M = 5$，$N = 4$.

```
#include <stdio.h>
int main()
{
    int a[5][4]; // 用于存储输入矩阵
    int maxi_for_each_row[5]; // 用于存储各行的最大值
```

```
6
7       printf("Please input elements for a 5*4 matrix:\n");
8       for(int i = 0; i < 5; i++) // 枚举每一行
9       {
10          printf("Input the %d-th row:\n", i + 1);
11          for(int j = 0; j < 4; j++) // 枚举每一列
12          {
13              scanf("%d", &a[i][j]);
14          }
15      }
16
17      printf("The input matrix is:\n");
18      for(int i = 0; i < 5; i++) // 枚举每一行
19      {
20          for(int j = 0; j < 4; j++) // 枚举每一列
21          {
22              printf("%d\t", a[i][j]);
23          }
24          printf("\n");
25      }
26
27      // 以下求每一行的最大值
28      for(int i = 0; i < 5; i++) // 枚举每一行
29      {
30          maxi_for_each_row[i] = a[i][0]; // 假定该行第一列的元素是最大的
31          for(int j = 1; j < 4; j++) // 枚举每一列
32          {
33              if(a[i][j] > maxi_for_each_row[i]) // 找到更大的
34              {
35                  maxi_for_each_row[i] = a[i][j]; // 更新最大值
36              }
37          }
38      }
39
40      // 先假定第一个元素最小
41      int min_max = maxi_for_each_row[0];
42      for(int i = 1; i < 5; i++)
43      {
44          if(min_max > maxi_for_each_row[i]) // 找到更小
```

```
45          {
46              min_max = maxi_for_each_row[i]; // 更新最小
47          }
48      }
49
50      printf("The minimum value of the maximum in each row is: %d\n", min_max);
51  }
```