

**实验十三**

**指针+字符数组&二维数组**

学期：2024-2025 第一学期

编制日期：2024 年 11月 28 日

编制人：江家玮

学号：22281188

班级：计科2204

**实验十三**

**知识覆盖：指针与字符串、指针与二维数组**

1. **有一个包含n个字符的字符串，写一个函数将这个字符串中从第m个字符开始的全部字符复制成为另一个字符。并在主程序中调用该函数进行测试。**

函数原型分别为：void fun(char \*s, char \*c, int m, int n)。

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <stdio.h>

#include <string.h>

// 函数原型

void fun(char\* s, char\* c, int m, int n) {

// 检查m是否在有效范围内

if (m < 1 || m > n) {

printf("错误：m的值超出范围。\n");

c[0] = '\0'; // 将目标字符串置为空

return;

}

// 从第m个字符开始复制

strncpy(c, s + m - 1, n - m + 1);

c[n - m + 1] = '\0'; // 添加字符串结束符

}

int main() {

char s[100]; // 源字符串

char c[100]; // 目标字符串

int m, n; // 起始位置和字符串长度

// 输入源字符串

printf("请输入源字符串: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0'; // 去除fgets读取的换行符

// 输入起始位置m

printf("请输入起始位置m: ");

scanf("%d", &m);

// 获取字符串长度

n = strlen(s);

// 调用函数进行复制

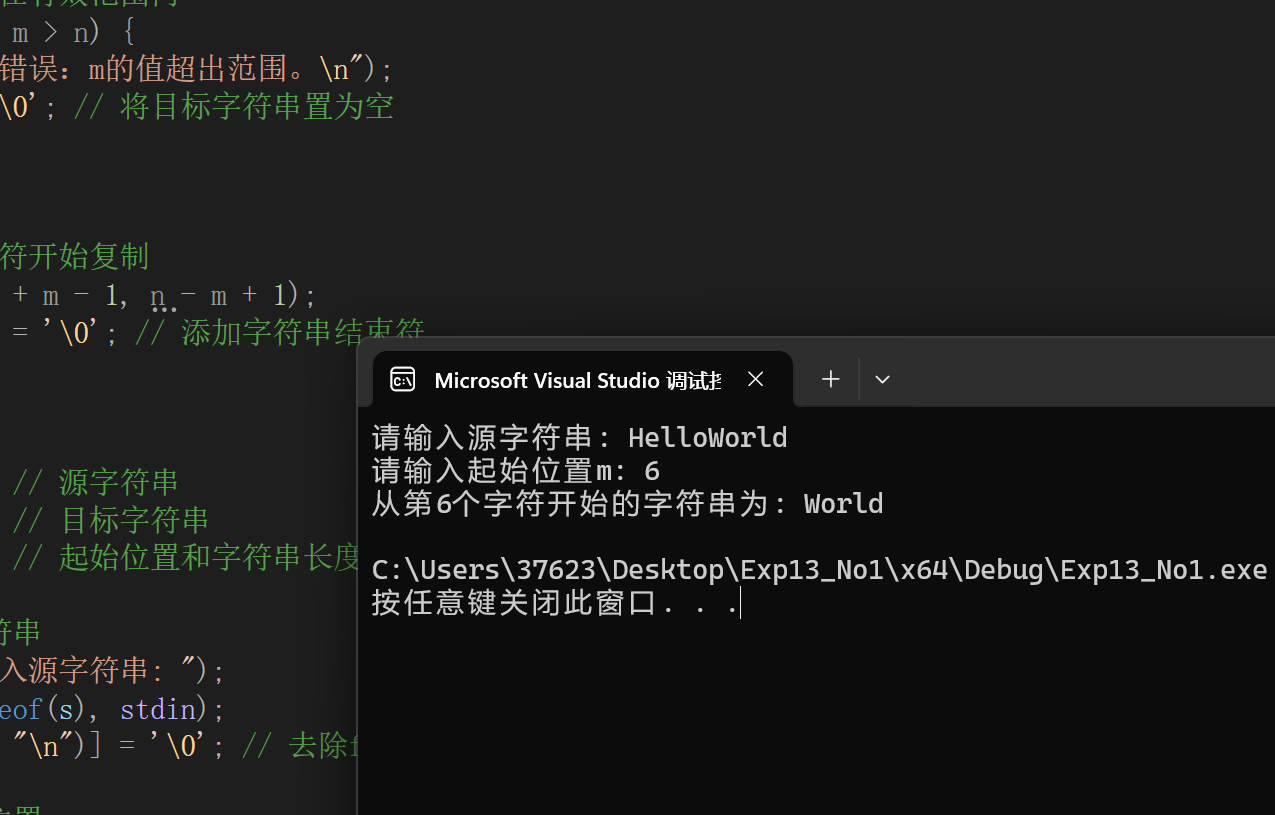
fun(s, c, m, n);

// 输出结果

printf("从第%d个字符开始的字符串为: %s\n", m, c);

return 0;

}



1. **分别用指针法写函数实现以下功能，并在主程序中进行测试。**
2. 写一个函数(自定义)，它检查两个字符串是否由同样的字符组成.
3. 写一个函数(自定义)，它判断一个字符串是否可以通过另一个字符串重排得到。

#include <stdio.h>

#include <string.h>

#include <ctype.h>

// 检查两个字符串是否由同样的字符组成

int areSameCharacters(char\* str1, char\* str2) {

int count1[256] = { 0 }; // 用于统计str1中每个字符的出现次数

int count2[256] = { 0 }; // 用于统计str2中每个字符的出现次数

// 统计str1中每个字符的出现次数

while (\*str1) {

count1[tolower(\*str1)]++; // 忽略大小写

str1++;

}

// 统计str2中每个字符的出现次数

while (\*str2) {

count2[tolower(\*str2)]++; // 忽略大小写

str2++;

}

// 比较两个字符串的字符统计结果

for (int i = 0; i < 256; i++) {

if (count1[i] != count2[i]) {

return 0; // 字符统计结果不同，返回0

}

}

return 1; // 字符统计结果相同，返回1

}

// 判断一个字符串是否可以通过另一个字符串重排得到

int canRearrange(char\* str1, char\* str2) {

return areSameCharacters(str1, str2); // 直接调用areSameCharacters函数

}

int main() {

char str1[100], str2[100];

// 输入第一个字符串

printf("请输入第一个字符串: ");

fgets(str1, sizeof(str1), stdin);

str1[strcspn(str1, "\n")] = '\0'; // 去除fgets读取的换行符

// 输入第二个字符串

printf("请输入第二个字符串: ");

fgets(str2, sizeof(str2), stdin);

str2[strcspn(str2, "\n")] = '\0'; // 去除fgets读取的换行符

// 测试areSameCharacters函数

if (areSameCharacters(str1, str2)) {

printf("两个字符串由同样的字符组成。\n");

}

else {

printf("两个字符串不包含同样的字符。\n");

}

// 测试canRearrange函数

if (canRearrange(str1, str2)) {

printf("第二个字符串可以通过第一个字符串重排得到。\n");

}

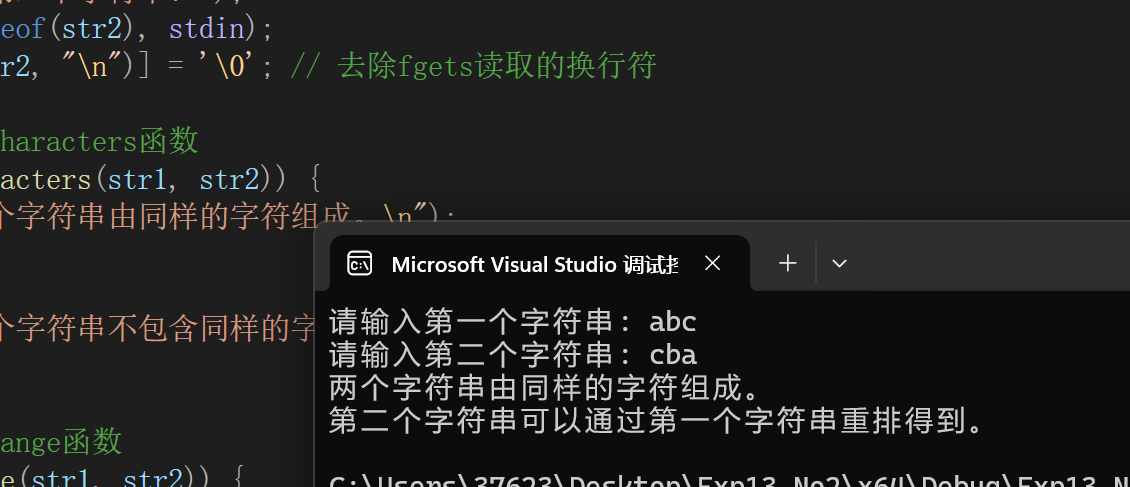
else {

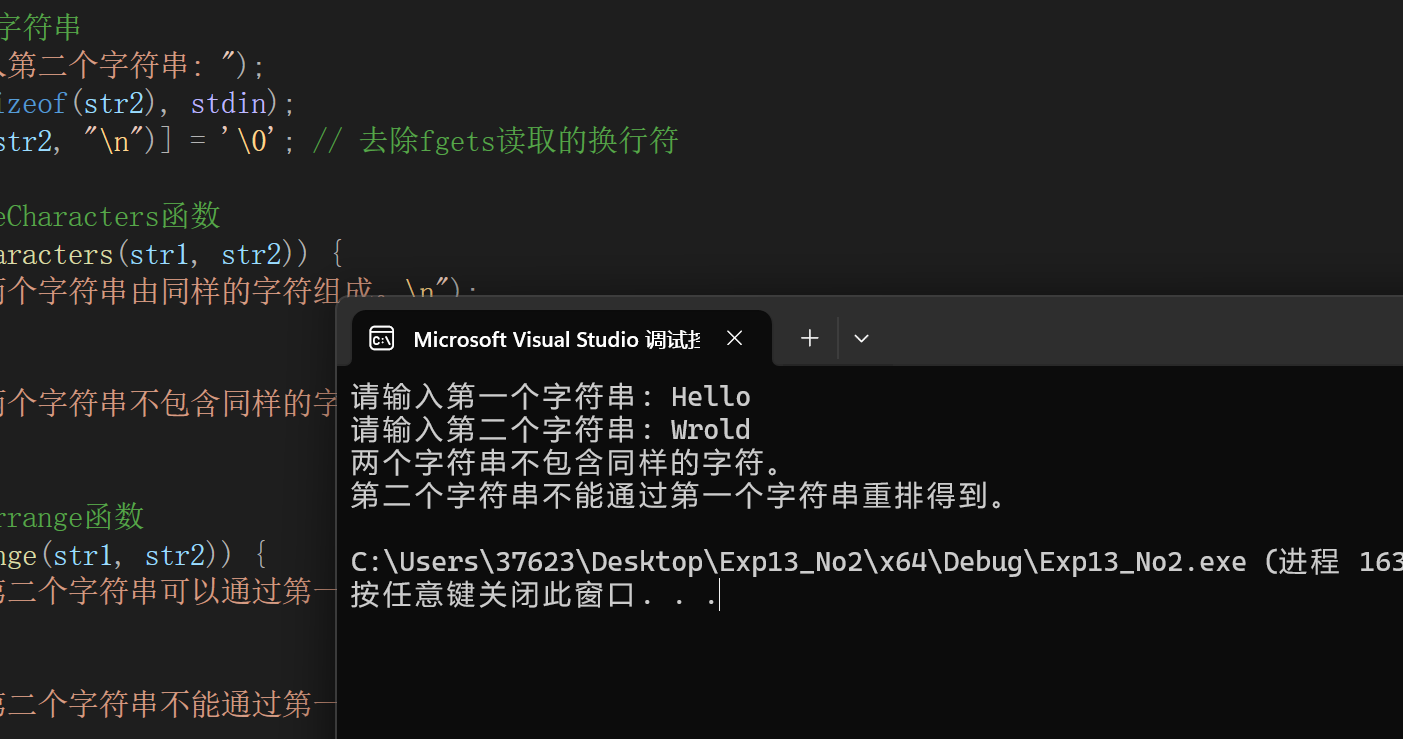
printf("第二个字符串不能通过第一个字符串重排得到。\n");

}

return 0;

}





1. **编写函数判断n阶矩阵是否对称，对称时返回1，不对称时返回0。main函数中定义矩阵并调用该函数进行判断。请分别用数组写法和指针写法完成上述功能。**

#include <stdio.h>

#include <stdlib.h>

// 使用数组写法判断矩阵是否对称

int isSymmetricArray(int matrix[][10], int n) {

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

for (int j = 0; j < n; j++) {

if (matrix[i][j] != matrix[j][i]) {

return 0; // 不对称，返回0

}

}

}

return 1; // 对称，返回1

}

// 使用指针写法判断矩阵是否对称

int isSymmetricPointer(int\* matrix, int n) {

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

for (int j = 0; j < n; j++) {

if (\*(matrix + i \* n + j) != \*(matrix + j \* n + i)) {

return 0; // 不对称，返回0

}

}

}

return 1; // 对称，返回1

}

int main() {

int n; // 矩阵的阶数

int matrix[10][10]; // 定义矩阵，假设最大阶数为10

// 输入矩阵的阶数

printf("请输入矩阵的阶数n (n <= 10): ");

scanf\_s("%d", &n);

// 输入矩阵的元素

printf("请输入%d阶矩阵的元素:\n", n);

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

for (int j = 0; j < n; j++) {

scanf\_s("%d", &matrix[i][j]);

}

}

// 使用数组写法判断矩阵是否对称

if (isSymmetricArray(matrix, n)) {

printf("使用数组写法判断：矩阵是对称的。\n");

}

else {

printf("使用数组写法判断：矩阵不是对称的。\n");

}

// 使用指针写法判断矩阵是否对称

if (isSymmetricPointer(&matrix[0][0], n)) {

printf("使用指针写法判断：矩阵是对称的。\n");

}

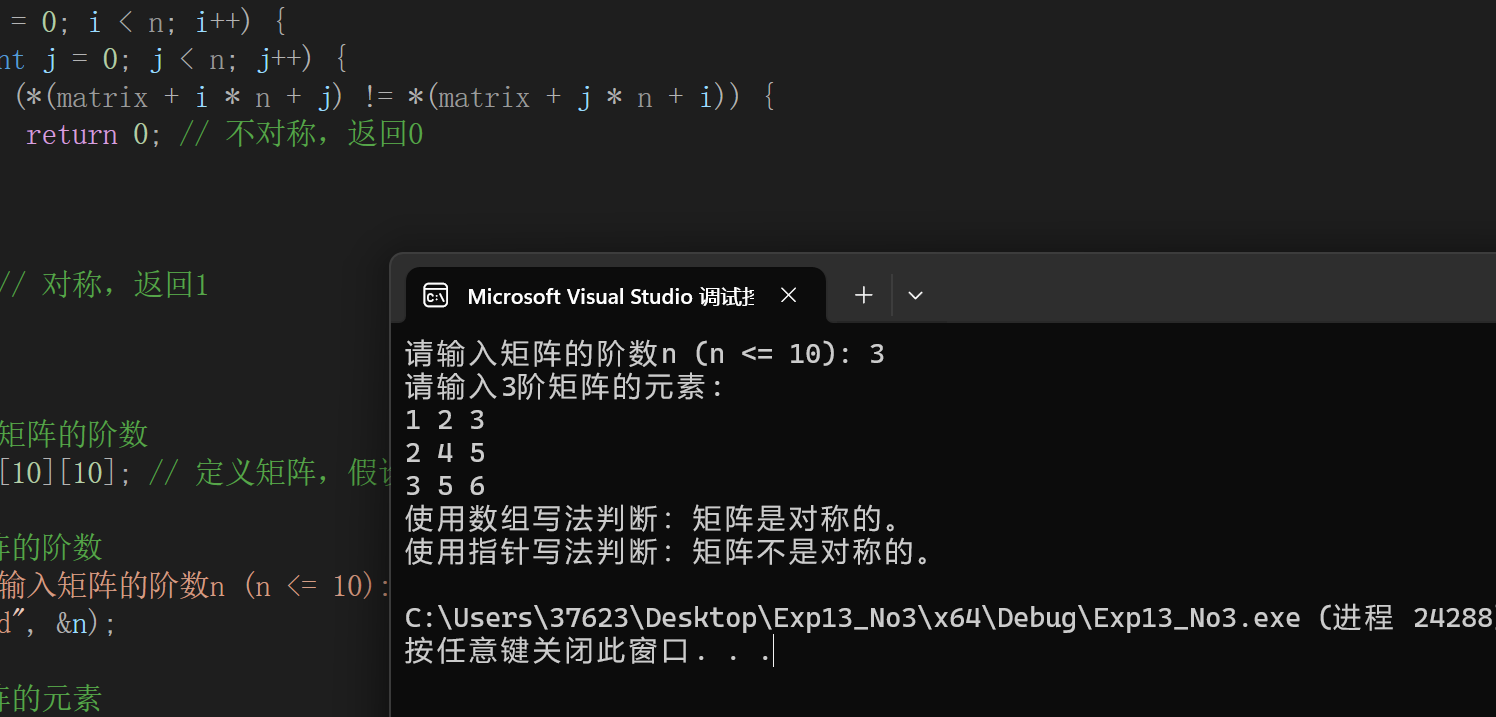
else {

printf("使用指针写法判断：矩阵不是对称的。\n");

}

return 0;

}





**4. 借助二维数组打印杨辉三角**

1

1　1

1　2　1

1　3　3　1

1　4　6　4　1

1　5 10 10　5　1

3...

#include <stdio.h>

// 打印杨辉三角

void printPascalTriangle(int n) {

int triangle[100][100]; // 定义二维数组存储杨辉三角

// 初始化杨辉三角的第一列和对角线为1

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

triangle[i][0] = 1; // 第一列全为1

triangle[i][i] = 1; // 对角线全为1

}

// 计算杨辉三角的其他元素

for (int i = 2; i < n; i++) {

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

triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];

}

}

// 打印杨辉三角

printf("杨辉三角（%d行）:\n", n);

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

// 打印每行的元素

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

printf("%d ", triangle[i][j]);

}

printf("\n");

}

}

int main() {

int n; // 杨辉三角的行数

// 输入杨辉三角的行数

printf("请输入杨辉三角的行数: ");

scanf\_s("%d", &n);

// 打印杨辉三角

printPascalTriangle(n);

return 0;

}



**5. 自定义函数找出二维矩阵的鞍点，如果没有鞍点打印相应信息**。

鞍点：该元素在矩阵所在的行中最大，所在的列中最小

#include <stdio.h>

// 查找二维矩阵的鞍点

void findSaddlePoint(int matrix[][10], int rows, int cols) {

int found = 0; // 标记是否找到鞍点

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

int maxRow = matrix[i][0]; // 假设当前行的第一个元素为最大值

int colIndex = 0; // 最大值所在的列索引

// 找出当前行的最大值及其列索引

for (int j = 1; j < cols; j++) {

if (matrix[i][j] > maxRow) {

maxRow = matrix[i][j];

colIndex = j;

}

}

// 检查最大值是否为所在列的最小值

int isSaddlePoint = 1; // 假设是鞍点

for (int k = 0; k < rows; k++) {

if (matrix[k][colIndex] < maxRow) {

isSaddlePoint = 0; // 不是鞍点

break;

}

}

// 如果找到鞍点，打印并标记

if (isSaddlePoint) {

printf("鞍点位于第 %d 行，第 %d 列，值为 %d\n", i + 1, colIndex + 1, maxRow);

found = 1;

}

}

// 如果没有找到鞍点，打印相应信息

if (!found) {

printf("矩阵中没有鞍点。\n");

}

}

int main() {

int rows, cols;

int matrix[10][10]; // 定义二维矩阵，假设最大维度为10x10

// 输入矩阵的行数和列数

printf("请输入矩阵的行数和列数（最大为10x10）: ");

scanf\_s("%d %d", &rows, &cols);

// 输入矩阵的元素

printf("请输入矩阵的元素:\n");

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

for (int j = 0; j < cols; j++) {

scanf\_s("%d", &matrix[i][j]);

}

}

// 查找鞍点

findSaddlePoint(matrix, rows, cols);

return 0;

}

