

**实验十四**

**综合练习1**

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

**实验十四**

**知识覆盖：**循环、函数、字符串、指针、结构体

1. 输入一个实数x，计算并输出下式的值，直到最后一项的绝对值小于10-5（保留两位小数）。
   1. 要求：计算公式在函数double sumFunc(double x)中实现，在主程序中进行测试。

#include <stdio.h>

#include <math.h>

// 计算阶乘

double factorial(int n) {

double result = 1.0;

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

result \*= i;

}

return result;

}

// 计算公式的值

double sumFunc(double x) {

double sum = 0.0;

double term = x;

int n = 1;

while (fabs(term) >= 1e-5) {

sum += term;

n++;

term = pow(-1, n - 1) \* pow(x, n) / factorial(n);

}

return sum;

}

int main() {

double x;

// 输入实数x

printf("请输入一个实数x: ");

scanf\_s("%lf", &x);

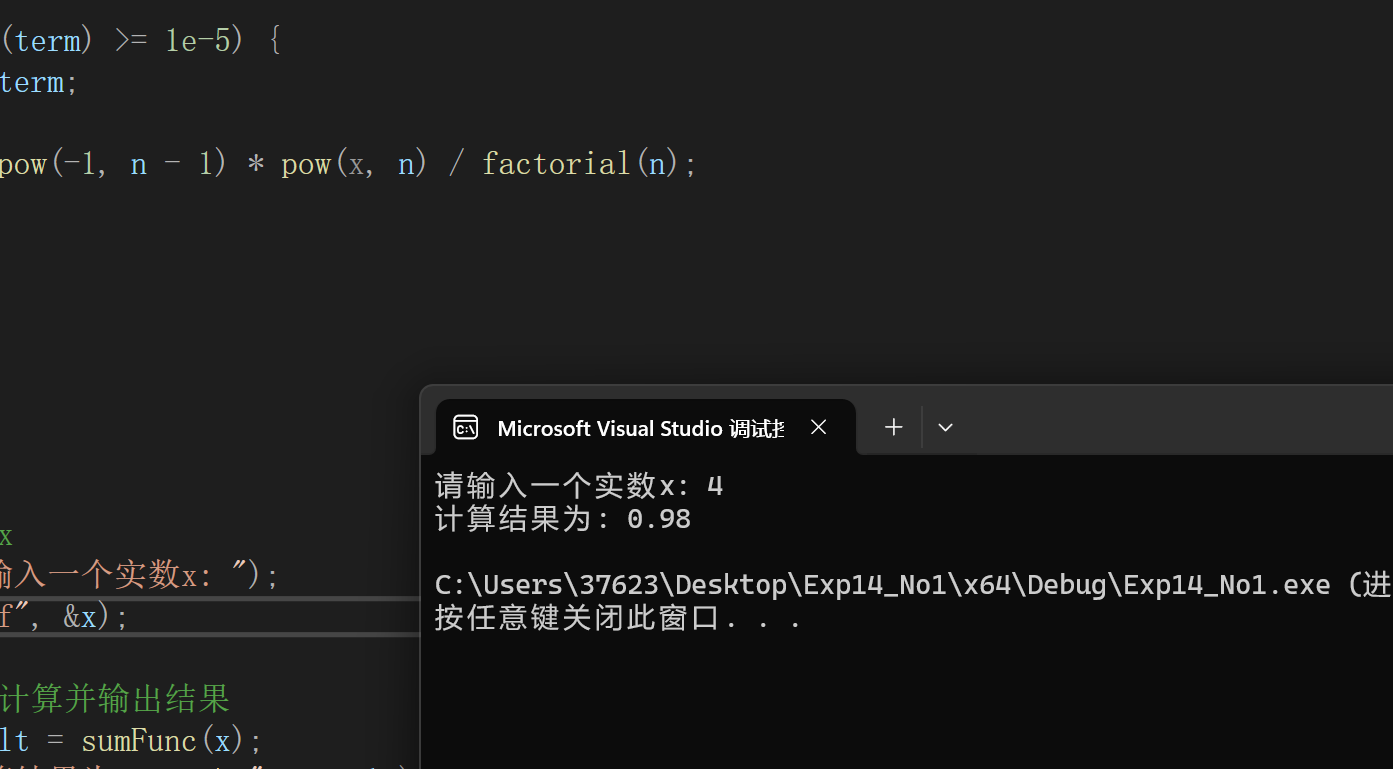
// 调用函数计算并输出结果

double result = sumFunc(x);

printf("计算结果为: %.2f\n", result);

return 0;

}



1. 如果一个数恰好等于它的所有因子（包括 1 但不包括自身）之和，则称之为“完数”。例如： 6 的因子是 1、2、3，且 6=1+2+3 ，因此 6 是完数。编写程序找出 5到500 之内的所有完数并求出这些完数的和。

要求：输出m至n之间的所有完数并求和的过程在函数int sumWanshu(int m, int n)中实现，在主程序中进行测试。其中，在本例中m=5、n=500，函数的返回值为m至n所有完数的总和。

#include <stdio.h>

// 判断一个数是否为完数

int isPerfectNumber(int num) {

int sum = 1; // 1是所有数的因子

// 遍历2到num/2，找出所有因子并求和

for (int i = 2; i <= num / 2; i++) {

if (num % i == 0) {

sum += i;

}

}

// 判断是否为完数

return (sum == num);

}

// 找出m到n之间的所有完数并求和

int sumWanshu(int m, int n) {

int sum = 0;

printf("在 %d 到 %d 之间的完数有: ", m, n);

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

if (isPerfectNumber(i)) {

printf("%d ", i);

sum += i;

}

}

printf("\n");

return sum;

}

int main() {

int m = 5, n = 500;

// 调用函数找出完数并求和

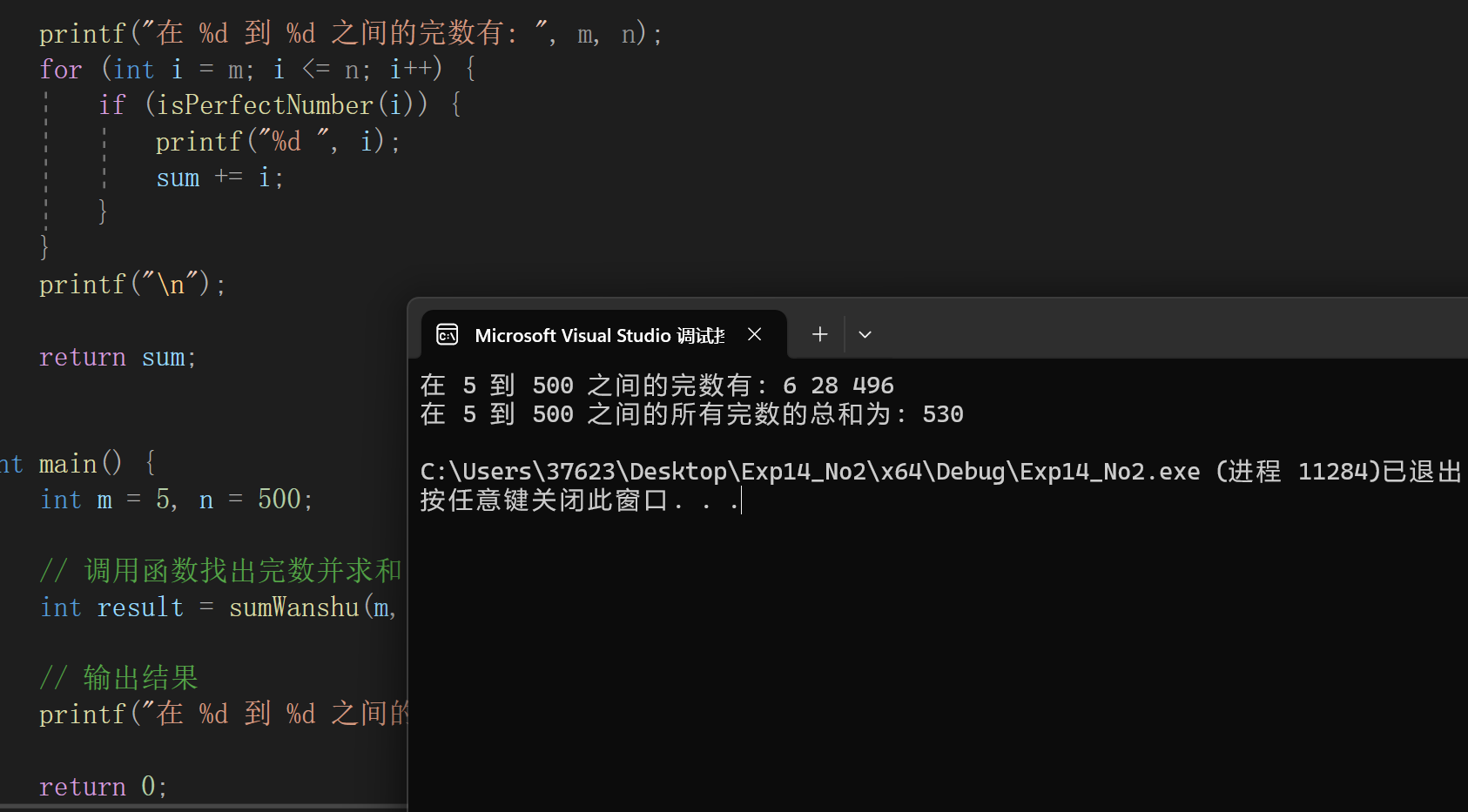
int result = sumWanshu(m, n);

// 输出结果

printf("在 %d 到 %d 之间的所有完数的总和为: %d\n", m, n, result);

return 0;

}



3. 从键盘输入10个整数保存在一个数组中。编写一个程序，它能够把从指定位置nStart开始的nInverseNumber个数按逆序重新排列，并输出新的完整的数列。逆序排列过程由函数实现，在主函数中完成输入和输出，并调用函数进行逆序。函数原型为

void inverse(int \*array, int nStart, int nInverseNumber)；或者

void inverse(int array[],int nStart, int nInverseNumber)；

例：原数组为 6，8，10，12，14，16，18，20，22，24若要求把从第6个数开始的4个数按逆序重新排列，则得到新数列为 6，8，10，12，14，22，20， 18， 16，24

#include <stdio.h>

// 逆序排列函数

void inverse(int\* array, int nStart, int nInverseNumber) {

int startIndex = nStart - 1; // 数组索引从0开始，nStart从1开始

int endIndex = startIndex + nInverseNumber - 1; // 逆序部分的结束索引

// 逆序排列

while (startIndex < endIndex) {

// 交换两个元素

int temp = array[startIndex];

array[startIndex] = array[endIndex];

array[endIndex] = temp;

// 移动索引

startIndex++;

endIndex--;

}

}

int main() {

int array[10]; // 定义数组，存储10个整数

int nStart, nInverseNumber;

// 输入10个整数

printf("请输入10个整数:\n");

for (int i = 0; i < 10; i++) {

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

}

// 输入起始位置和逆序的个数

printf("请输入起始位置nStart (1-10): ");

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

printf("请输入逆序的个数nInverseNumber: ");

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

// 调用函数进行逆序排列

inverse(array, nStart, nInverseNumber);

// 输出新的数组

printf("逆序排列后的数组为:\n");

for (int i = 0; i < 10; i++) {

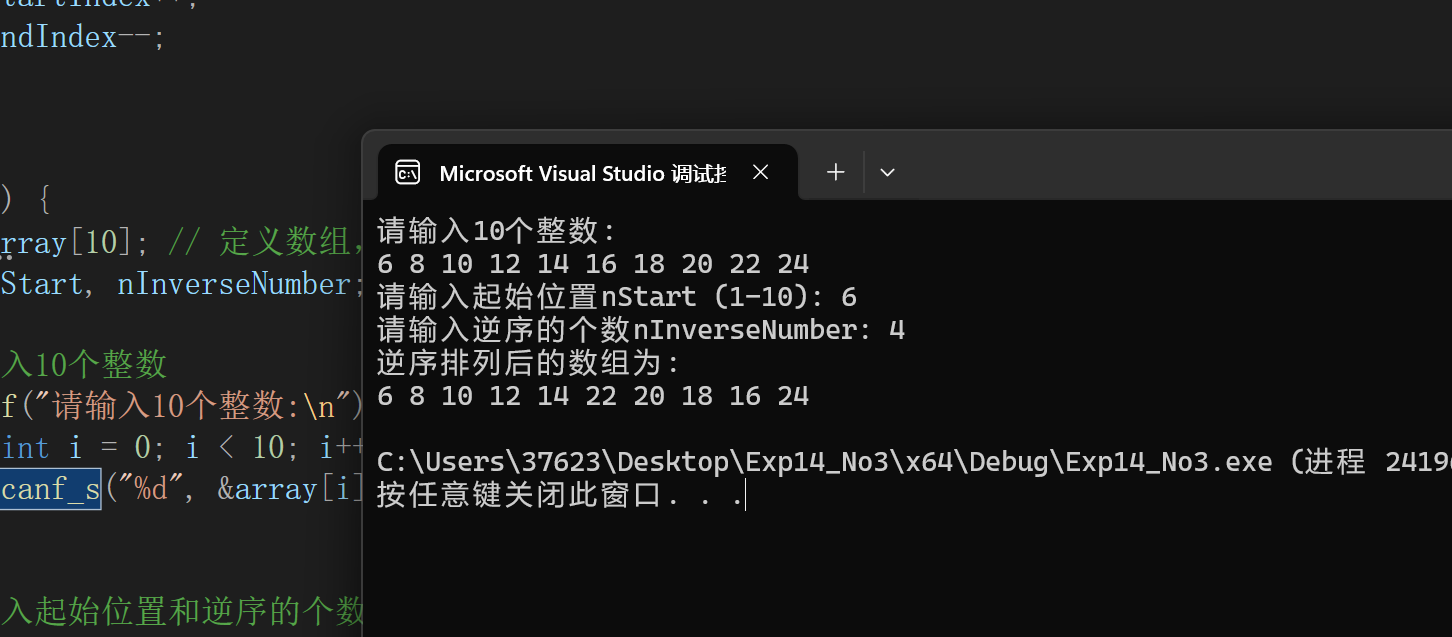
printf("%d ", array[i]);

}

printf("\n");

return 0;

}



4. 请编写函数将字符串str2拼接到字符串str1的后面，拼接时在两个字符串中间加一个空格，在新字符串最后加一个’!’，如字符串str1为”Hello”，字符串 str2为”Beijing”，拼接后的结果为”Hello Beijing!”。并在主程序中测试。

函数原型为char \* StrCat(char\* Str1, char\* Str2)。

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

// 字符串拼接函数

char\* StrCat(char\* Str1, char\* Str2) {

// 计算拼接后的字符串长度

int len1 = strlen(Str1);

int len2 = strlen(Str2);

int totalLen = len1 + len2 + 3; // 加1个空格和1个'!'，再加1个字符串结束符

// 分配足够的内存来存储拼接后的字符串

char\* result = (char\*)malloc(totalLen \* sizeof(char));

if (result == NULL) {

printf("内存分配失败！\n");

return NULL;

}

// 复制Str1到结果字符串

strcpy(result, Str1);

// 在Str1后面添加一个空格

strcat(result, " ");

// 将Str2拼接到结果字符串

strcat(result, Str2);

// 在结果字符串末尾添加'!'

strcat(result, "!");

return result;

}

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

// 调用StrCat函数进行拼接

char\* result = StrCat(str1, str2);

// 输出结果

if (result != NULL) {

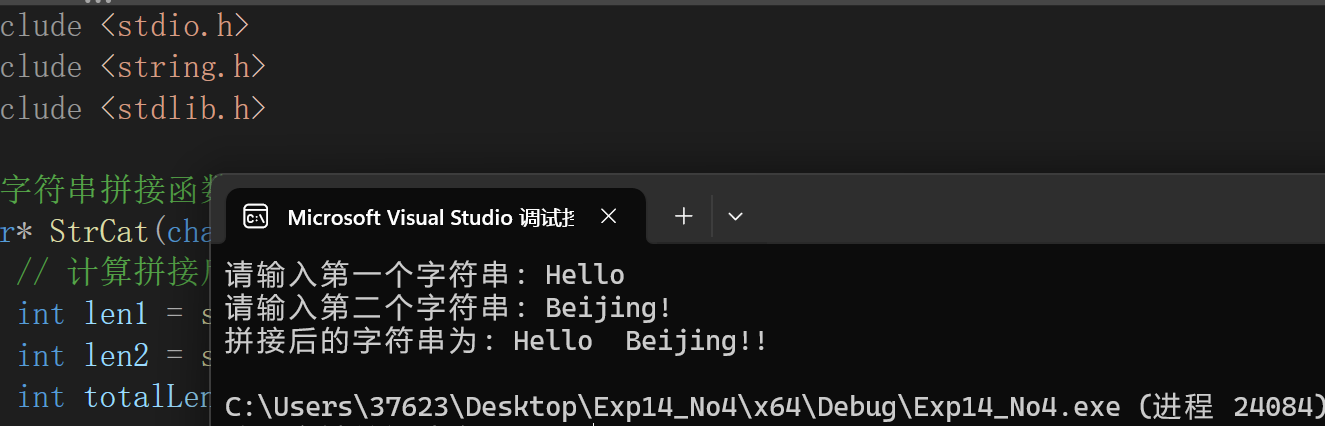
printf("拼接后的字符串为: %s\n", result);

free(result); // 释放动态分配的内存

}

return 0;

}



5. 设有如下结构描述一本书：

typedef struct{

char Name[50]; //书名

double Price; //价钱

int Pages; //页数

}BOOK;

设有一批书如下(以下是测试样例，可以有多本书)：

3

Machine Learning 88.00 425

C Language 34.50 378

Data Mining 39.00 154

输入包含nLen+1行：第一行是一个整数nLen（0 < nLen <= 10）, nLen表征有多少本书。第二行至nLen +1行是nLen本书的记录：书名、价钱、页数，每本书占一行，不同分量之间用一个空格隔开。

1) 请编写如下函数：int MaxPages(BOOK \*pBook, int nLen)找出该批书中页数最多的一本书并打印其所有信息，返回这本书的页数。在主程序中测试。

2) 请编写如下函数：double MeanPages (BOOK \*pBook, int nLen)求出这批书的平均页数，返回该平均页数。在主程序中测试。

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <stdio.h>

#include <string.h>

// 定义书的结构

typedef struct {

char Name[50]; // 书名

double Price; // 价钱

int Pages; // 页数

} BOOK;

// 找出页数最多的一本书并打印其信息，返回这本书的页数

int MaxPages(BOOK\* pBook, int nLen) {

int maxPages = pBook[0].Pages; // 假设第一本书的页数最多

int maxIndex = 0; // 记录页数最多的书的索引

// 遍历所有书，找出页数最多的一本

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

if (pBook[i].Pages > maxPages) {

maxPages = pBook[i].Pages;

maxIndex = i;

}

}

// 打印页数最多的一本书的信息

printf("页数最多的一本书是: %s\n", pBook[maxIndex].Name);

printf("价钱: %.2lf\n", pBook[maxIndex].Price);

printf("页数: %d\n", pBook[maxIndex].Pages);

return maxPages;

}

// 求出这批书的平均页数，返回该平均页数

double MeanPages(BOOK\* pBook, int nLen) {

int totalPages = 0;

// 计算所有书的页数总和

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

totalPages += pBook[i].Pages;

}

// 计算平均页数

return (double)totalPages / nLen;

}

int main() {

int nLen; // 书的数量

BOOK books[10]; // 定义书的数组，最多10本书

char line[100]; // 用于存储每行输入

// 输入书的数量

printf("请输入书的数量 (0 < nLen <= 10): ");

scanf("%d", &nLen);

getchar(); // 读取换行符，避免影响后续输入

// 输入每本书的信息

printf("请输入每本书的书名、价钱和页数:\n");

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

printf("第 %d 本书: ", i + 1);

fgets(line, sizeof(line), stdin); // 读取一行输入

sscanf(line, "%s %lf %d", books[i].Name, &books[i].Price, &books[i].Pages);

}

// 找出页数最多的一本书并打印其信息

int maxPages = MaxPages(books, nLen);

printf("页数最多的一本书的页数是: %d\n", maxPages);

// 计算并输出平均页数

double meanPages = MeanPages(books, nLen);

printf("这批书的平均页数是: %.2lf\n", meanPages);

return 0;

}

