

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.1.7**

**软件学院**

**目 录**

**[1 表达式和标准输入与输出 2](#_Toc404837920)**

**[2 流程控制实验 1](#_Toc404837921)5**

**[3 函数与程序结构试验 3](#_Toc404837922)4**

**[4 编译预处理 5](#_Toc404837923)2**

**5 数组实验 67**

**6 指针实验.................................................................................................................80**

**7 结构与联合实验.....................................................................................................104**

**8 文件实验.................................................................................................................122**



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.10.15**

**软件学院**

**目 录**

**[1 表达式和标准输入输出实验 3](#_Toc404837920)**

[1.1 实验目的 3](#_Toc404837921)

[1.2 实验内容 3](#_Toc404837922)

[1.3 实验小结 1](#_Toc404837923)1

# 1 表达式和标准输入与输出实验

# 1.1 实验目的

（1）熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型及运算过程中的类型转换，重点是C语言特有的运算符，例如位运算符，问号运算符，逗号运算符等；熟记运算符的优先级和结合性。

（2）掌握getchar, putchar, scanf 和printf 函数的用法。

（3）掌握简单C程序（顺序结构程序）的编写方法。

## 1.2 实验内容

### 1 源程序改错

下面给出了一个简单C语言程序例程，用来完成以下工作：

（1）输入华氏温度f，将它转换成摄氏温度c后输出；

（2）输入圆的半径值ｒ，计算并输出圆的面积ｓ；

（3）输入短整数ｋ、ｐ，将ｋ的高字节作为结果的低字节，ｐ的高字节作为结果的高字节，拼成一个新的整数，然后输出；

在这个例子程序中存在若干语法和逻辑错误。要求参照1.3和1.4的步骤对下面程序进行调试修改，使之能够正确完成指定任务。

1 #include<stdio.h>

2 #define PI 3.14159;

3 voidmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

11 c = 5/9\*(f-32) ;

12 printf( “ \n %d (F) = %.2f (C)\n\n ”, f, c ) ;

13 /\* for task 2 \*/

14 printf("input the radius r:");

15 scanf("%f", &r);

16 s = PI \* r \* r;

17 printf("\nThe acreage is %.2f\n\n",&s);

18 /\* for task 3 \*/

19 printf("input hex int k, p :");

20 scanf("%x %x", &k, &p );

21 newint = (p&0xff00)|(k&0xff00)<<8;

22 printf("new int = %x\n\n",newint);

}

**解答**（1）错误修改：

1. 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.14159

1. 第5行的f的类型错误，正确形式为:

double f；

3) 第10行的scanf(“%d”,f)漏掉取址符&，且f的类型有误，正确形式为:

scanf(“%lf”,&f);

4)第11行中“5/9”得到整数，与c的类型不符，正确形式为:

c=5.0/9.0\*(f-32);

1. 第12行表达式结果与变量类型不符，正确形式为：

printf( “ \n %.2lf (F) = %.2lf (C)\n\n ”, f, c ) ;

1. 第15行由于r的类型为double,所以应改为:

scanf("%lf", &r);

7) 第17行的s前不需取址符，正确形式为:

printf("\nThe acreage is %.2f\n\n",s);

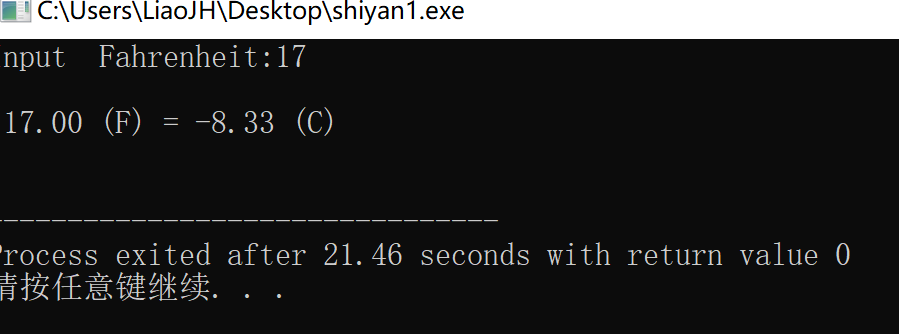
8) 第21行出现未定义的变量newint,应在前面补上：

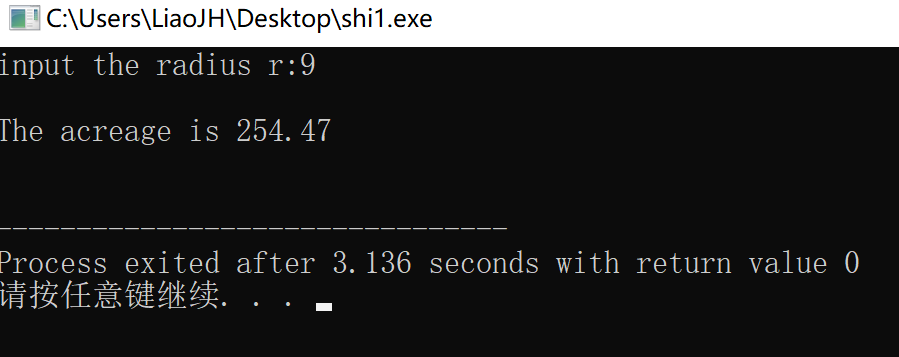
int newint;

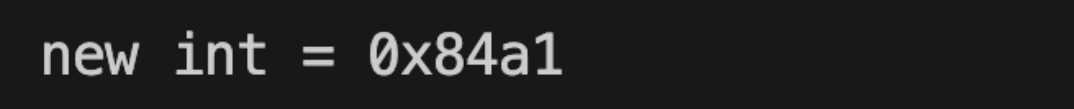
9）第21行运算有误，应改为:

newint=(p&0xff00)|(k>>8);

（2）错误修改后运行结果：

**（task 1）**

**(task 2)**

**(task 3)**

### 2 源程序修改替换

下面的程序利用常用的中间变量法实现两数交换，请改用不使用第3个变量的方法实现。该程序中t是中间变量，要求将定义语句中的t删除，修改下划线处的语句，使之实现两数对调的操作。

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

t=a ；a=b；b=t；

prinf(“\na=%d,b=%d”,a,b);

}

**解答：**

替换后的程序如下所示：

#include<stdio.h>

void main( )

{

int a, b;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

a=a^b；

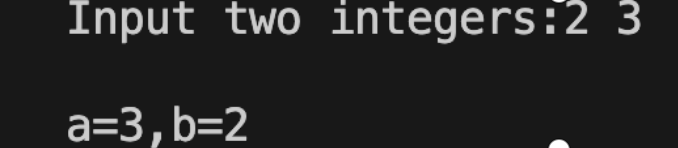
b=a^b；

a=a^b；

printf(“\na=%d,b=%d”,a,b);

}

运行结果如下：

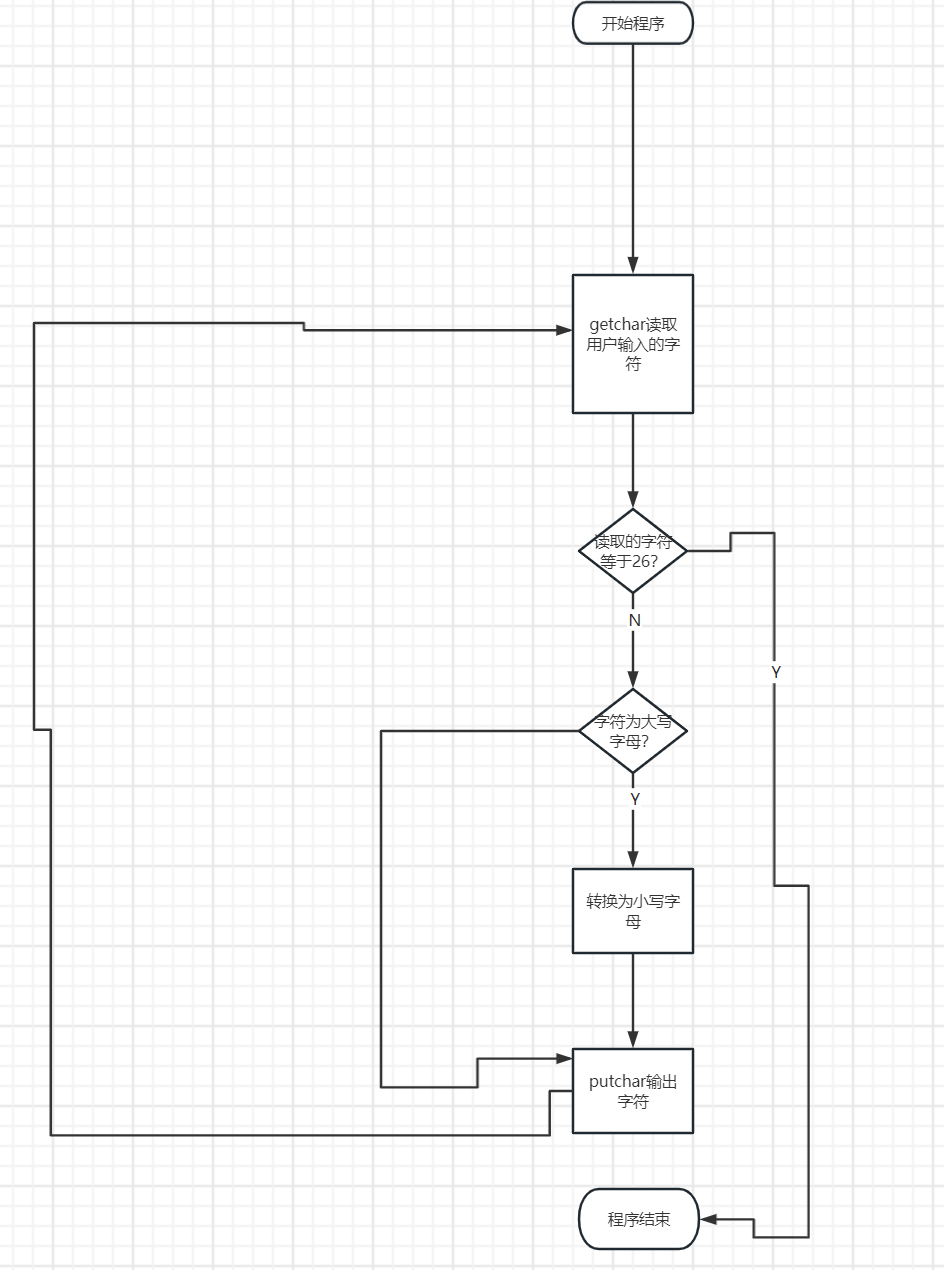


### 3 程序设计

（1）输入字符ｃ，如果ｃ是大写字母，则将ｃ转换成对应的小写，否则ｃ的值不变，输入Ctrl+Z程序结束。要求①用条件表达式；②字符的输入输出用getchar和putchar函数。程序应能循环接受用户的输入，直至输入Ctrl+Z程序结束。

**解答：**

1. 算法流程如下图：



2）源程序清单

#include <stdio.h>

int main() {

char c;

while(1) {

printf("请输入一个字符：\n");

c = getchar();

if(c == (char)26) {

break;

}

if(c >= 'A' && c <= 'Z') {

c = c + ('a' - 'A');

}

printf("转换后的字符是：%c\n", c);

}

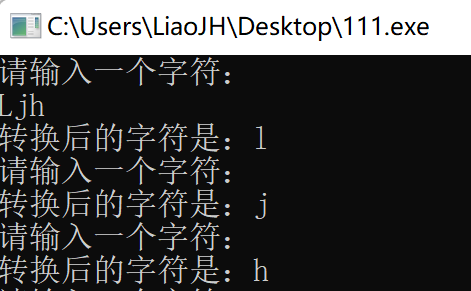
return 0;

}

3）测试

（a） 测试数据：Ljh

（b） 对应测试数据的运行结果截图



（2）输入无符号短整数x，ｍ，ｎ（0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ）,取出x从第ｍ位开始向左的ｎ位（ｍ从右至左编号为0～15），并使其向左端（第15位）靠齐。要求：①检查m和n的范围；②x的值以十六进制输入，m和n以十进制输入；③结果以十六进制输出。

**解答：**

1) 解题思路：

1.输入x，m，n，为了方便分析测试结果，x的输入采用16进制

2.如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，转2.1，否则转3.

2.1 首先x>>m，将要处理的n位移动到最右；

2.2 再将上一步的结果左移(14-n)位，即：(x>>m)<<(16-n)

2.3 用16进制输出结果并转4.

3. 显示输入错误信息；

4. 结束

2）程序清单

 #include<stdio.h>

 int main()

{

    unsigned short int m, n, x, y;

    scanf("%hx%hd%hd", &x, &m, &n);

    if (m < 0 || m > 15 || n < 1 || n > 16 - m)

    {

        printf("error");

        return 0;

    }

    y = (x >> m) << (16 - n);

    printf("%hx", y);

    return 0;

}

3）测试

（a） 测试数据：

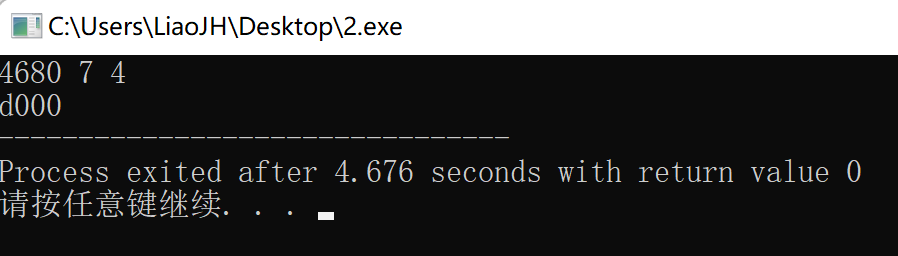
叙述选择测试数据的方法:以m,n的各自取值与二者组合是否在所给范围取3组数值测试，保证测试的全面性和一般性。

如表1-1所示。

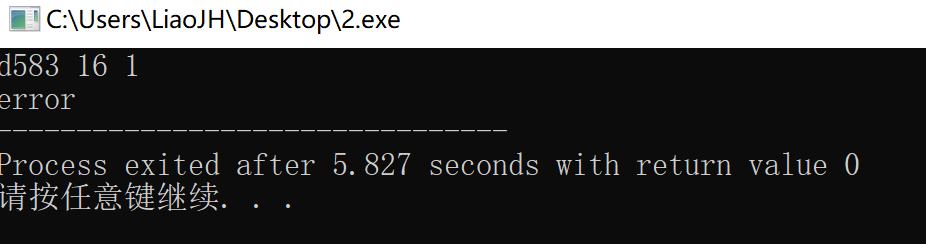
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 | 运 行 结 果 |
| X | m | N |
| 用例1 | 0100 0110 1000 0000（4680） | 7 | 4 | 计算结果1101 0000 0000 0000 即D000 | D000 |
| 用例2 | 1101 0101 1000 0011（D583） | 16 | 1 | 输入错误（m值超范围） | error |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） | error |

表1-1 编程题2的测试数据

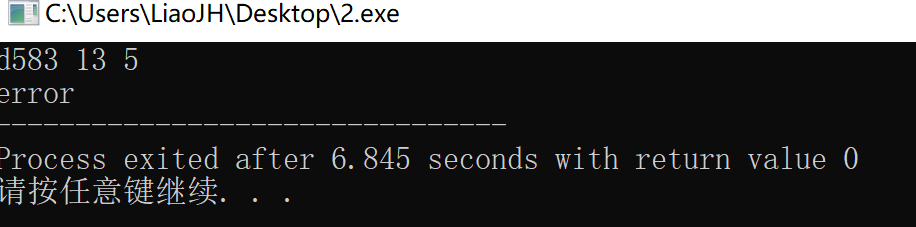
（b）对应测试测试用例1的运行结果如下图所示。



对应测试测试用例2的运行结果如下图所示。



对应测试测试用例3的运行结果如下图所示。



说明上述的运行结果与理论分析吻合，验证了程序的正确性。

(3) IP地址通常是4个用句点分隔的小整数（即点分十进制），但这些地址在机器中是用一个无符号长整型数表示的。例如3232235876，其机内二进制表示就是11000000 10101000 00000001 01100100，按照8位一组用点分开，该IP地址就写成192.168.1.100。

读入无符号长整型数表示的互联网IP地址，对其译码，以常见的点分十进制形式输出。要求循环输入和输出，直至输入Ctrl+Z结束。

**解答：**

1. 解题思路：
2. 定义unsigned long int *IP*;
3. 如果IP!=EOF就转2.1，否则转3；

2.1将IP分解为四个八位的数字ab,c,d；

2.2计算结果后依次输出；

3.结束。

2)程序清单

#include <stdio.h>

int main()

{

unsigned long int IP;

unsigned long int a, b, c, d;

while (scanf("%ld", &IP) != EOF)

{a = (address & 0xff000000) >> 24,

b = (address & 0x00ff0000) >> 16,

c = (address & 0x0000ff00) >> 8,

d = address & 0x000000ff,

printf("%d.%d.%d.%d\n", a, b, c, d);}

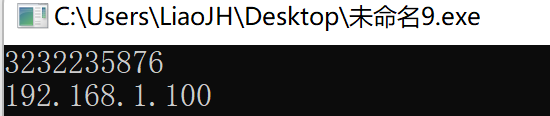
return 0;

}

3）测试

（a） 测试数据：3232235876

（b） 对应测试数据的运行结果截图



说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1.3 实验小结

在本次实验我通过三个程序改错和实际编写和调试三个程序，深入了解了C语言特有的运算符和输入输出函数的用法，也积累了处理程序调试问题的经验。

在调试过程中，我采用了逐步调试的方式，逐渐缩小问题范围，最终准确定位问题所在。同时，熟练掌握了使用调试器的各种技巧，如断点设置、单步执行等。

通过本次实验，我体会到了理论学习和实践应用相结合的重要性。只有将理论知识运用到实际编程中，才能更好地理解和掌握它们。这些经验将对今后的学习产生积极的影响。



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.10.20**

**软件学院**

**目 录**

2. 流程控制实验.........................................................................................2

2.1 实验目的.................................................................................................2

2.2 实验内容.................................................................................................2

2.3 实验小结.................................................................................................20

# 实验2 流程控制实验

2.1 实验目的

（1）掌握复合语句、if语句、switch语句的使用，熟练掌握for、while、do-while三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

2.2 实验内容及要求

**1．程序改错**

下面的实验2-1程序是合数判断器（合数指自然数中除了能被1和本身整除外，还能被其它数整除的数），在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够正确完成指定任务。

/\* 实验2-1改错题程序：合数判断器\*/

1 #include <stdio.h>

2 int main( )

3 {

4 int i, x, k, flag = 0;

5 printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

6 while (scanf("%d", &x) !=EOF && x>1) {

7 for(i=2,k=x>>1;i<=k;i++)

8 if (!x%i) {

9 flag = 1;

10 break;

11 }

12 if(flag=1) printf("%d是合数", x);

13 else printf("%d不是合数", x);

flag=0;

14 }

15 return 0;

16 }

解答：

（1）错误修改

1) 第6行while的判断补上x>1,以满足程序要求（输入大于1的整数）

2）第8行由于运算符优先级问题出错，正确形式为：

！（x%i）；

3）第12行if语句的条件判断不应用赋值运算符，正确形式为：

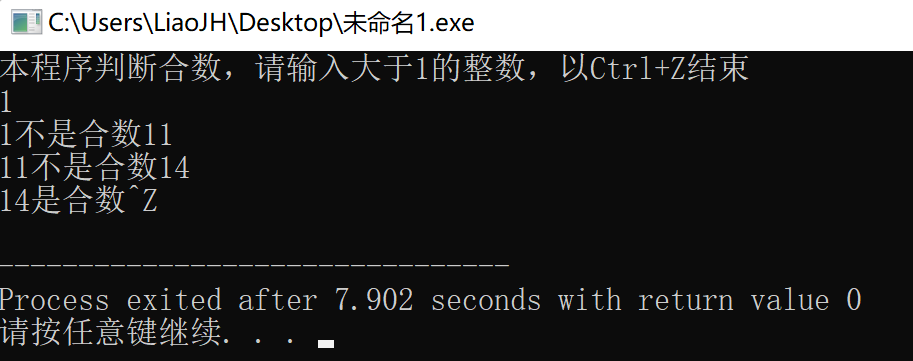
flag==1；

4）第13行在合数判断结束后应重新将flag的值调回0以进行下一个数字判断，应加上：

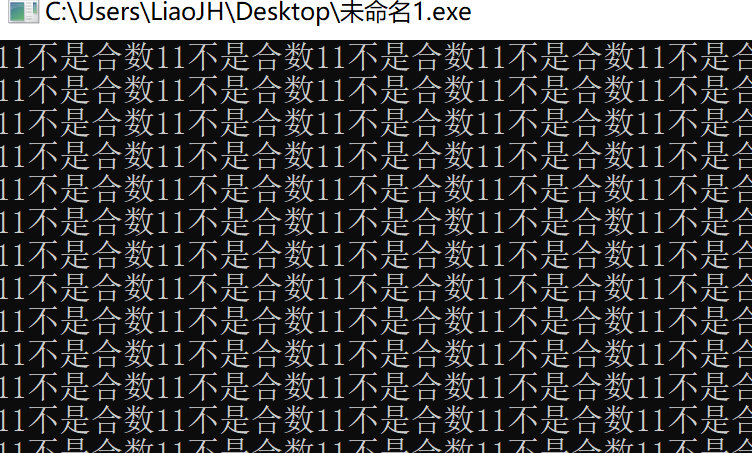
flag=0;

1. 错误修改后运行结果：

输入整数并以Ctrl+Z结束：



若输入不符合要求的数（如浮点数），则会出现死循环，用户可知输入不合要求：



**2．程序修改替换**

（1）修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句。

#include <stdio.h>

int main( )

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

while (scanf("%d", &x) !=EOF) {

for(i=2,k=x>>1;i<=k;i++)

if (!(x%i)) {

flag = 1;

}

if(flag==1) printf("%d是合数", x);

else printf("%d不是合数", x);

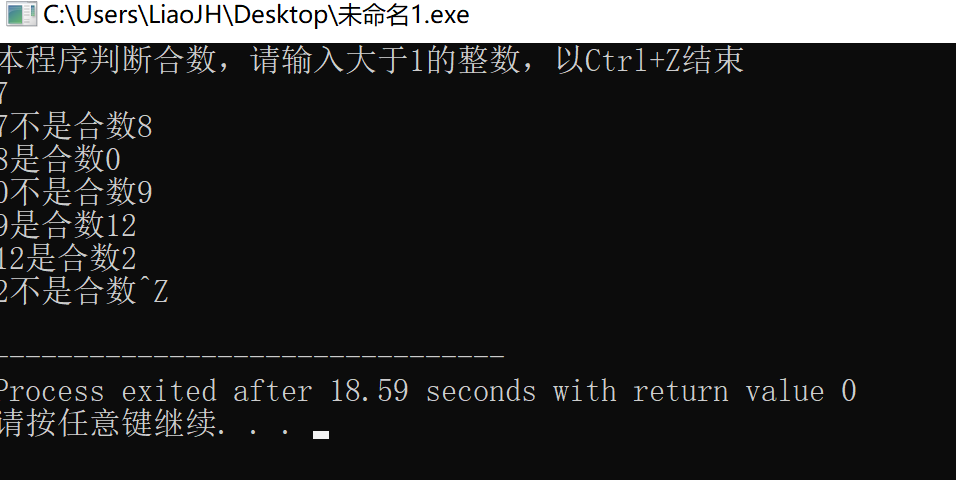
flag=0;

}

return 0;

}

程序运行截图：



1. 修改实验2-1程序，将for循环改用do-while循环。

解答：

#include <stdio.h>

int main() {

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

while (scanf("%d", &x) != EOF) {

flag = 0;

i = 2;

k = x >> 1;

do {

if (x % i == 0) {8

flag = 1;

break;

}

i++;

} while (i <= k);

if (flag == 1) {

printf("%d是合数\n", x);

} else {

printf("%d不是合数\n", x);

}

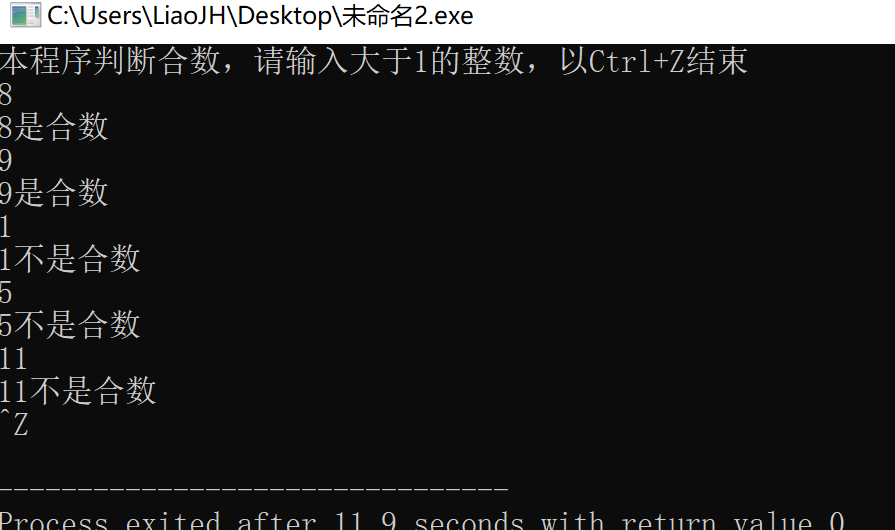
flag = 0;

}

return 0;

}

程序运行结果如图：



1. 修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。

解答：在2-1基础上将合数判断器改造为一个函数，然后再遍历三位数通过isComposite函数判断是否为纯粹合数并打印输出

#include <stdio.h>

int isComposite(int x) {

int i;

for (i = 2; i <= x / 2; i++) {

if (x % i == 0) {

return 1;

}

}

return 0;

}

int main() {

int i, j;

for (i = 100; i < 1000; i++) {

if (isComposite(i) && isComposite(i/10) && isComposite(i/100) ) {

printf("%d\n", i);

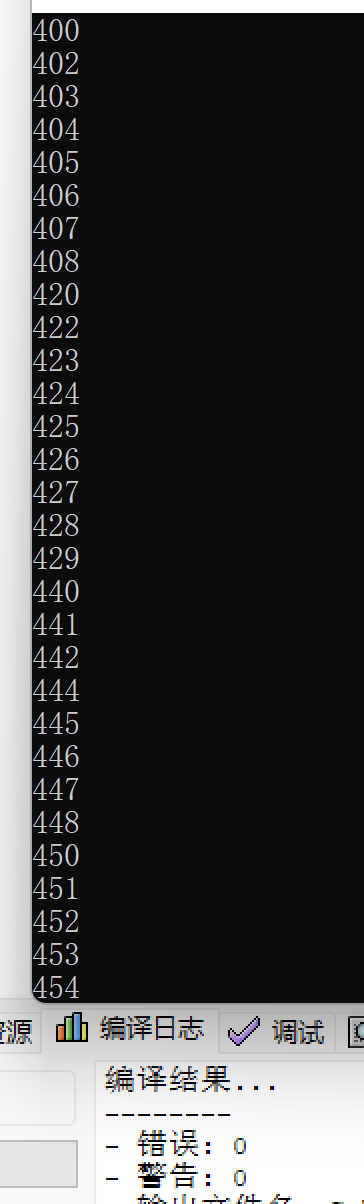
}

}

return 0;

}

程序运行结果截图：

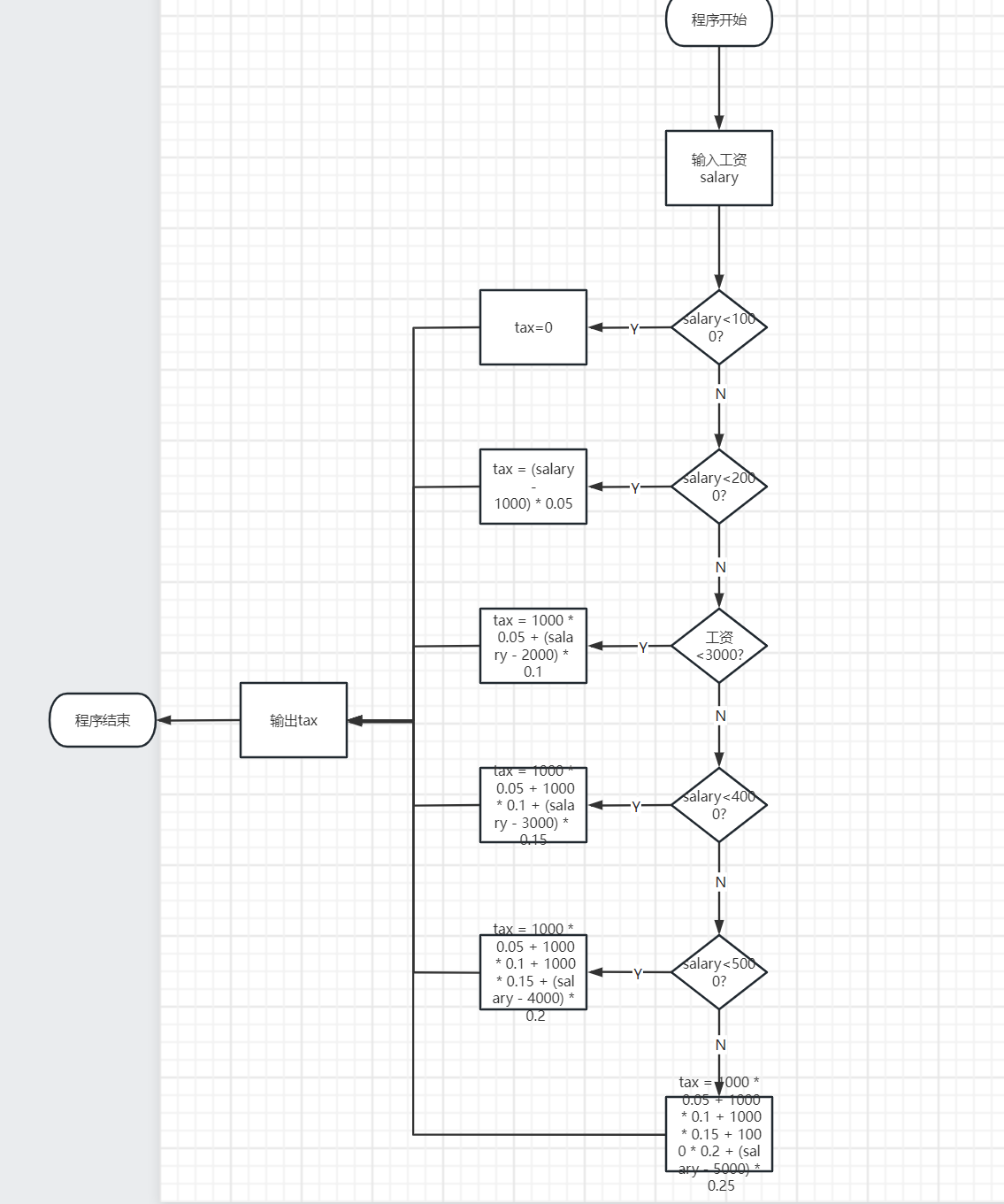


**3．程序设计**

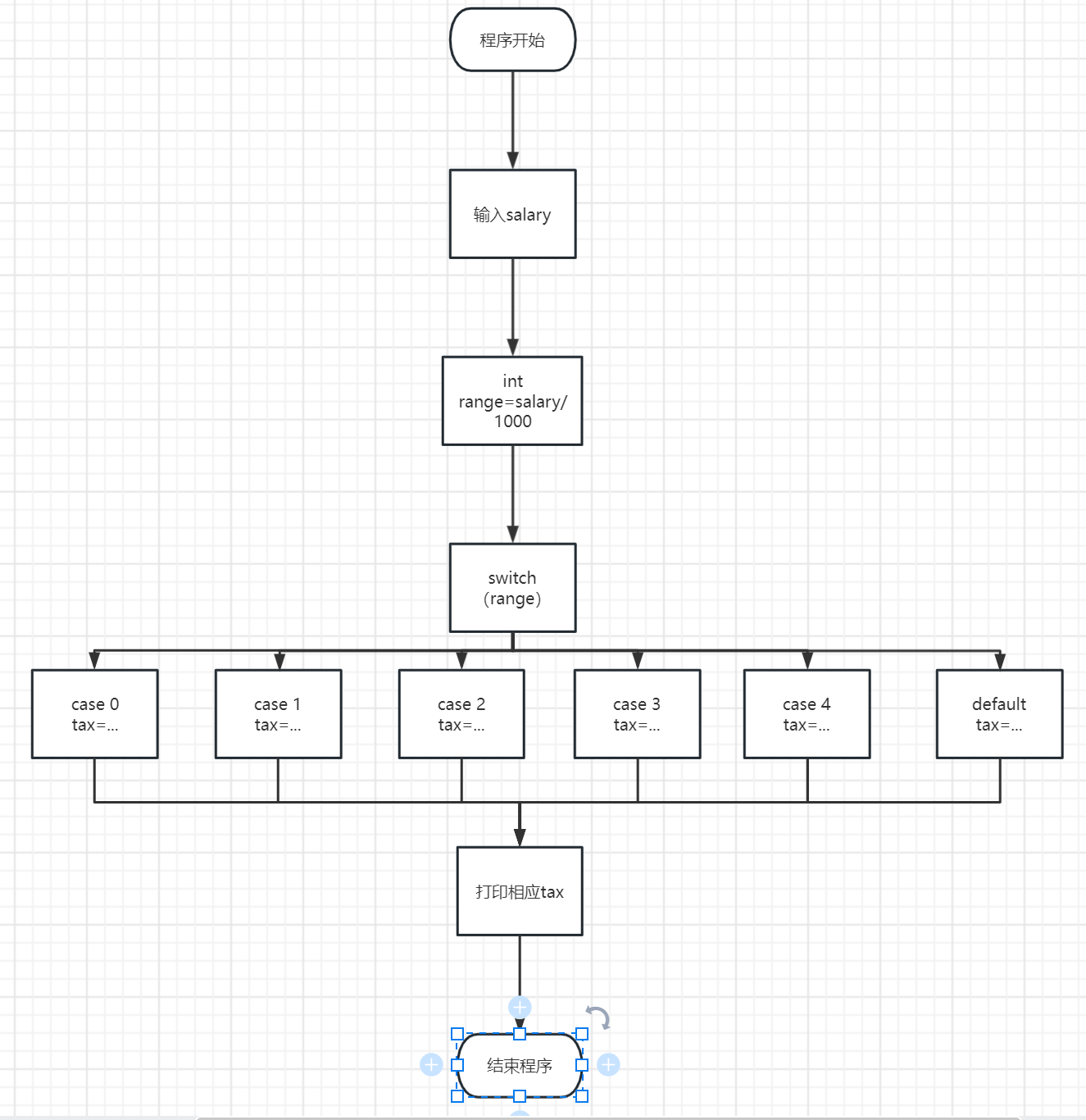
（1） 假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；5000 ≤ x，收取25%的税金。（注意税金的计算按照阶梯计税法，比如，工资为4500，那么税金=1000\*5% + 1000\*10% + 1000\*15% + 501\*20%）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

1. 算法流程图

（a）if语句：



（b）switch语句：



1. 源程序清单：
2. if语句版本

#include <stdio.h>

int main() {

double salary, tax;

printf("请输入您的工资：");

scanf("%lf", &salary);

if (salary < 1000) {

tax = 0;

} else if (salary < 2000) {

tax = (salary-1000) \* 0.05;

} else if (salary < 3000) {

tax = 1000 \* 0.05 + (salary - 2000) \* 0.1;

} else if (salary < 4000) {

tax = 1000 \* 0.05 + 1000 \* 0.1 + (salary - 3000) \* 0.15;

} else if (salary < 5000) {

tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 + (salary - 4000) \* 0.2;

} else {

tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 + 1000 \* 0.2 + (salary - 5000) \* 0.25;

}

printf("您应缴纳的税金为：%.2lf\n", tax);

return 0;

}

1. switch语句版本：

#include <stdio.h>

int main() {

double salary, tax=0;

int range;

printf("请输入您的工资：\n");

scanf("%lf",&salary);

range=salary/1000;

switch(range) {

case 0:

tax = 0;

break;

case 1:

tax = (salary-1000) \* 0.05;

break;

case 2:

tax = 1000 \* 0.05 + (salary - 2000) \* 0.1;

break;

case 3:

tax = 1000 \* 0.05 + 1000 \* 0.1 + (salary - 3000) \* 0.15;

break;

case 4:

tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 + (salary - 4000) \* 0.2;

break;

default:

tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 + 1000\* 0.2 + (salary - 5000) \* 0.25;

break;

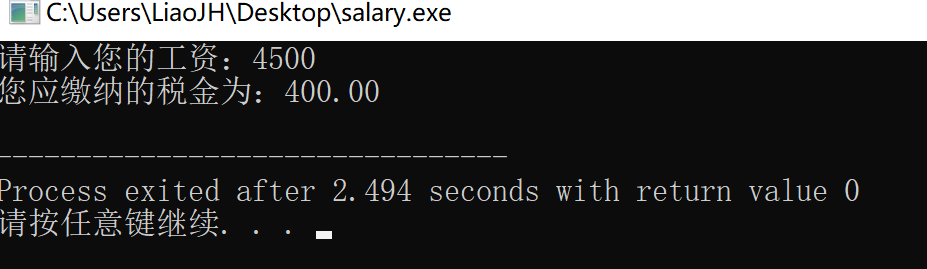
}

printf("您应缴纳的税金为：%2lf\n", tax);

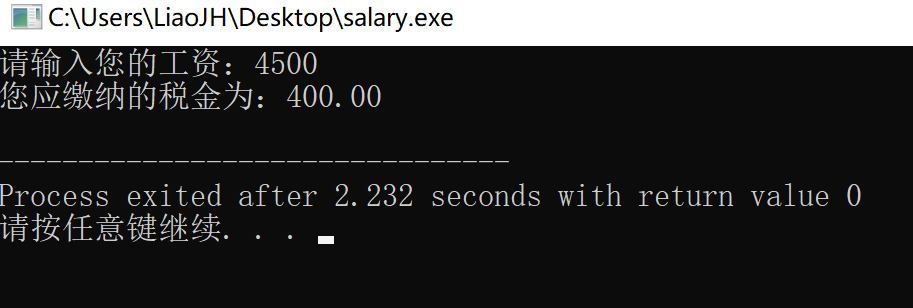
}

1. 测试
2. 测试数据：4500
3. 对应测试数据的运行结果截图

if版本：



switch版本：



1. 输入一段以!结尾的短文(最多5行,每行不超过50个字符)，要求将它复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替。

1. 算法流程：

程序首先从标准输入中读取一个字符，如果该字符不是感叹号（'!'），则继续读取下一个字符。如果读取到的字符是空格，则检查之前是否已经输出过一个空格，如果已经输出过，则不输出该空格，并将空格计数器清零。如果读取到的字符不是空格，则输出该字符，并统计当前行中的空格数。如果读取到的字符是换行符（'\n'），则将空格计数器清零。当读取到感叹号时，程序结束。

1. 源程序清单：

 #include<stdio.h>

int main()

{

    char c, i = '0';

    int flag = 0;

    while (scanf("%c", &c) != EOF)

    {

        if (c != ' ')

            putchar(c);

        else

        {

            if (i == c)

                continue;

            else

                putchar(c);

        }

        i = c;

    }

    return 0;

}

1. 测试：



（3）打印如下的杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

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

1 9 36 84 126 126 84 36 9 1

第i行第j列位置的数据值可以由组合表示，而的计算如下：

|  |  |
| --- | --- |
|  | (i=0,1,2,…) |
|  | (j=0,1,2,3,…,i) |

根据以上公式，采用顺推法编程，输入最后一行的编号N（0<=N<=6），要求输出金字塔效果的杨辉三角形。

特别要注意空格的数目，一位数之间是3个空格，两位数之间有2个空格，3位数之间只有一个空格。第N行行首是N个空格。每行末尾是3个空格和换行。

1. 算法流程:

该程序主体为一个for嵌套的双重循环和一个for循环构成。输入行数N后，前者用于打印杨辉三角中的空格，后者用于打印三角中相应位置的值value，value的值根据题目所给的公式算出。

1. 源程序清单

#include <stdio.h>

int main() {

int N,i,j,k;

scanf("%d", &N);

for ( i = 0; i <= N; i++) {

for ( k = 1; k <= 2 \* (N - i) + N; k++) {

printf(" ");

}

int value = 1;

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

printf("%-4d", value);

value = value \* (i - j) / (j + 1);

}

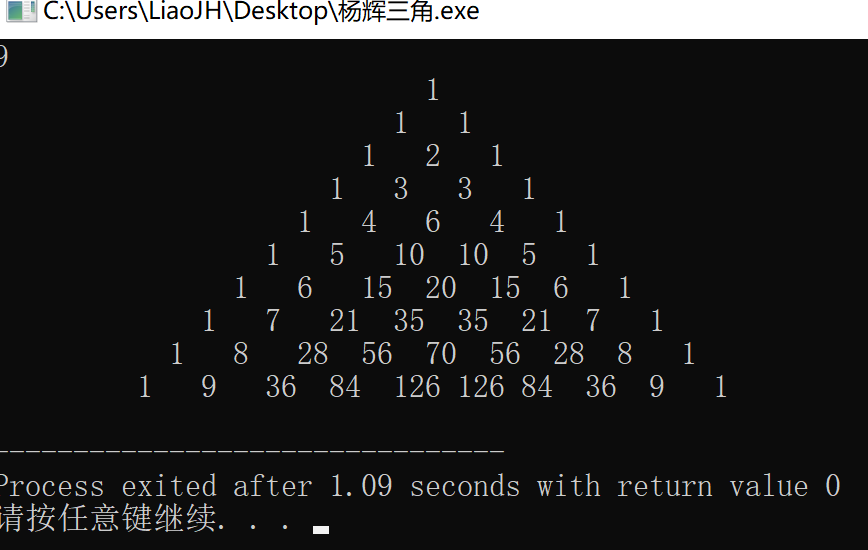
printf("\n");

}

return 0;

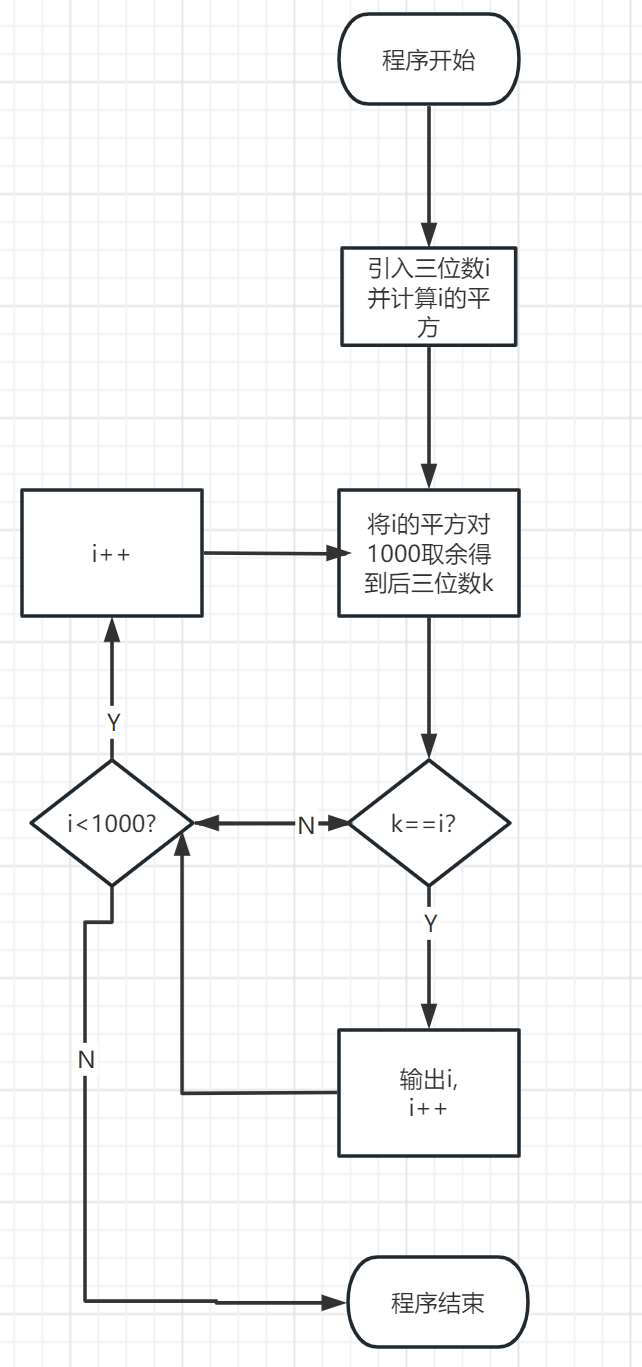
}

1. 测试
2. 测试数据：9
3. 测试结果如图：



（4）625这个数很特别，625的平方等于390625，其末3位也是625。请编程输出所有这样的3位数：它的平方的末3位是这个数本身。要求这些数字从小到大排列，每个数字单独占一行。

1. 算法流程如图所示：



1. 源程序清单：

#include<stdio.h>

int i=100;

int main() {

    for(i = 100; i < 1000; i++) {

        int square = i \* i;

        int k = square % 1000;

        if(k == i) {

            printf("%d\n", i);

        }

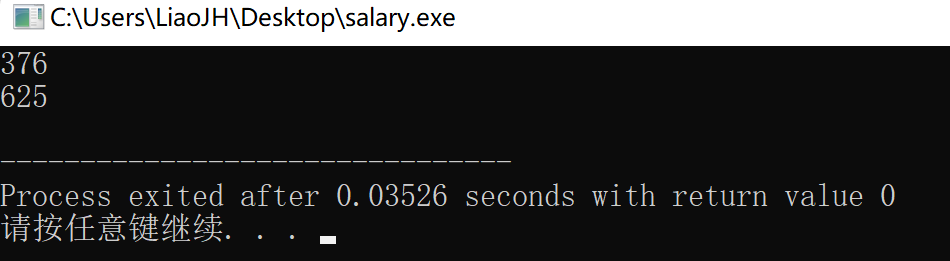
    }

    return 0;

}

1. 测试：

程序运行结果如图：



**2.3实验小结：**

在这次实验中，我学习了如何使用复合语句、if语句、switch语句以及for、while、do-while三种基本的循环控制语句。通过练习，我深入理解了这些语句的用法，并掌握了它们在编程中的实际应用。此外，我还利用集成开发环境中的调试功能，如单步执行、设置断点、观察变量值等，对程序进行细致的调试和分析。这些功能帮助我更好地理解程序的运行过程，并确保程序的正确性。

总之，这次实验让我更深入地理解了这些编程语句和调试技术，提高了我的编程能力和问题解决能力。我相信这些经验将对我未来的编程工作产生积极的影响。



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.10.23**

**软件学院**

**目 录**

3 函数与程序结构实验

3.1 实验目的

3.2 实验内容

3.3 实验小结

# 实验3 函数与程序结构实验

3.1实验目的

（1）熟悉和掌握函数的定义、声明；函数调用与参数传递，函数返回值类型的定义和返回值使用。

（2）熟悉和掌握不同存储类型变量的使用。

（3）练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

3.2实验内容

**1．程序改错题**

下面是计算s=1!+2!+3!+…+n!的源程序(n<20)。在这个源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够输出如下结果：

k=1 the sum is 1

k=2 the sum is 3

k=3 the sum is 9

……

k=20 the sum is 2561327494111820313

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

1 #include <stdio.h>

2 int main(void)

3 {

4 int k;

5 for(k=1;k<=20;k++)

6 printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));

7 return 0;

8 }

9 long sum\_fac(int n)

10 {

11 long s=0;

12 int i,fac;

13 for(i=1;i<=n;i++)

14 fac\*=i;

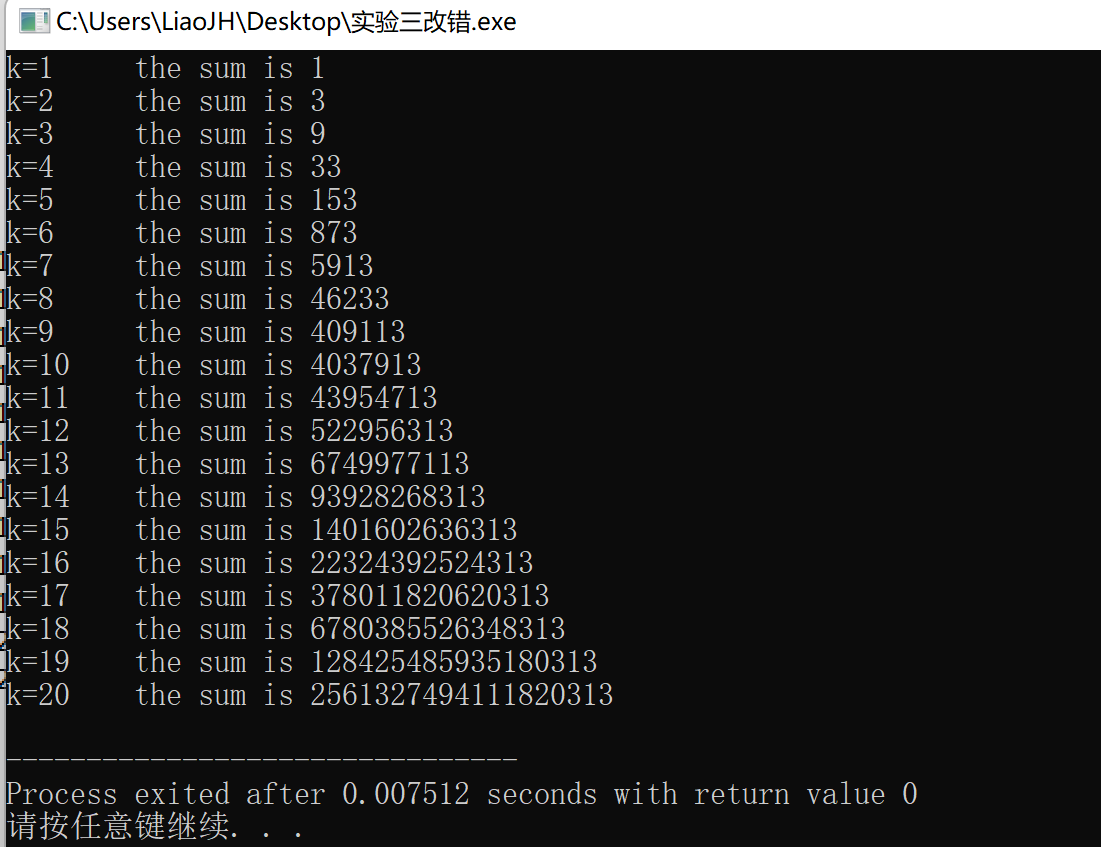
15 s+=fac;

16 return s;

17 }

解答：

1. 错误修改：
2. 阶乘求和最后结果数字超大，要把相关的函数和变量修改为unsigned long long类型，正确形式为：long sum\_fac(int n) 改为 unsigned long long sum\_fac(int n)，int fac 改为 unsigned long long fac，long s 改为 unsigned long long s
3. 函数调用前未声明，应在第1行和第2行之间补上：unsigned long long sum\_fac(int n);
4. fac变量应初始化为1才能进行阶乘：unsigned long long fac=1
5. 错误修改后运行结果：



**2．程序修改替换题**

（1）根据将实验3-1改错题程序中sum\_fac函数修改为一个递归函数，用递归的方式计算。

修改后程序：

#include <stdio.h>

unsigned long long sum\_fac(int n);

int main(void) {

int k;

for (k = 1; k <= 20; k++) {

printf("k=%d\tthe sum is %llu\n", k, sum\_fac(k));

}

return 0;

}

unsigned long long sum\_fac(int n) {

if (n == 1) {

return 1;

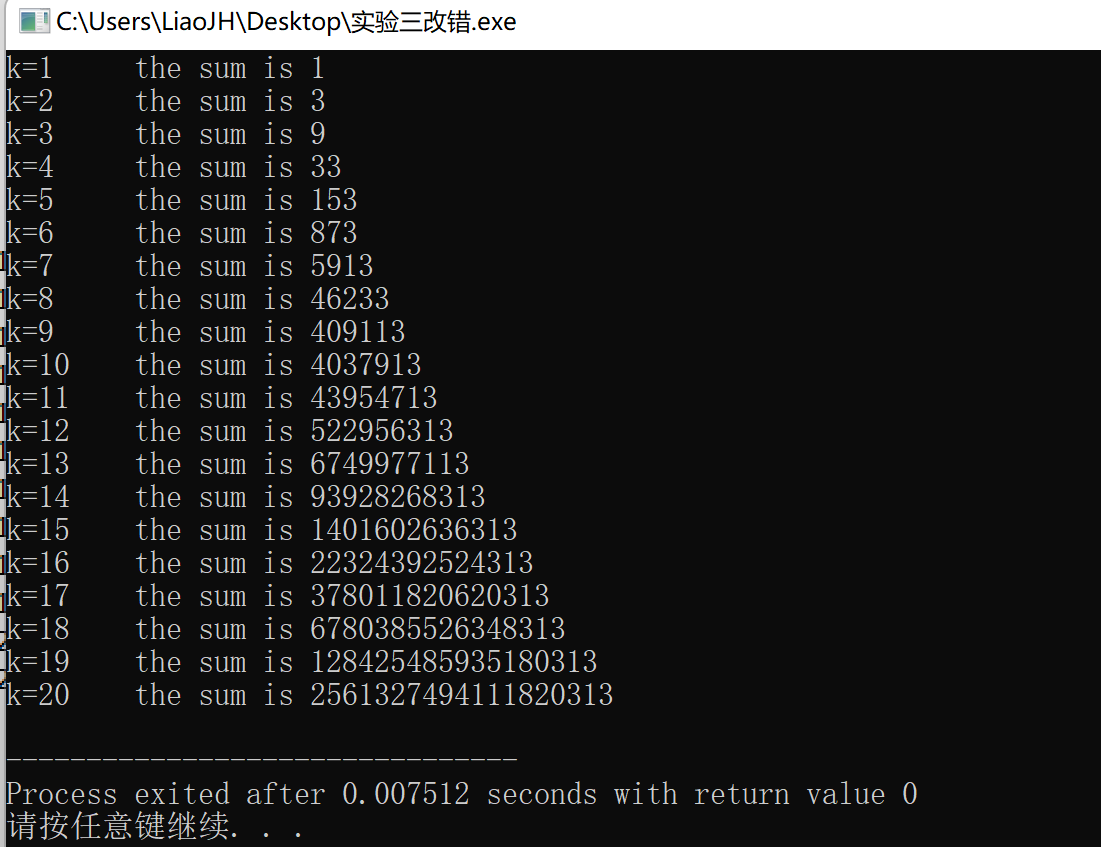
} else {

return n \* sum\_fac(n - 1);

}

}

程序运行结果截图：



（2）下面是计算的源程序，其中x是浮点数，n是整数。从键盘输入x和n，然后计算s的值。修改该程序中的sum和fac函数，使之计算量最小。

/\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n)

{

int i;

double z=1.0;

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

{

z=z+mulx(x,i)/fac(i);

}

return z;

}

double mulx(double x,int n)

{

int i;

double z=1.0;

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

{

z=z\*x;

}

return z;

}

long fac(int n)

{

int i;

long h=1;

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

{

h=h\*i;

}

return h;

}

int main()

{

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf:",sum(x,n));

return 0;

}

程序优化：

mulx函数中，不需要使用一个循环来乘以x，只需要乘以n次即可。fac函数和mulx函数有些重复，fac函数计算阶乘，而mulx函数实际上是在计算x的阶乘。因此，可以将这两个函数合并为一个函数，该函数既能计算阶乘也能计算连续的乘积。在sum函数中，我们只需要将每一项的结果相加。因此，我们可以预先计算每一项的值，然后将其相加，而不是在循环中每次都进行乘法和除法。

下面是修改后的代码：

#include<stdio.h>

long fac(int n) {

int i;

long h = 1;

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

h = h \* i;

}

return h;

}

double mulx(double x, int n) {

if (n == 1) {

return x;

} else {

return x \* mulx(x, n - 1);

}

}

double sum(double x, int n) {

double z = 0.0;

int i;

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

z = z + mulx(x, i) / fac(i);

}

return z;

}

int main() {

double x;

int n;

printf("Input x and n:");

scanf("%lf%d", &x, &n);

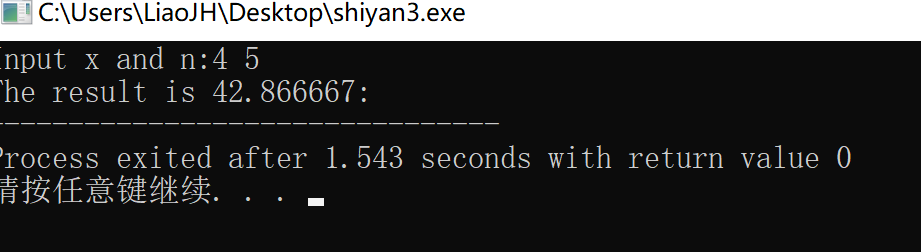
printf("The result is %lf:", sum(x, n));

return 0;

}

程序运行结果：

1. 测试数据：x=4，n=5
2. 运行结果：



**3．跟踪调试题**

下面是计算fabonacci数列前n项和的源程序，现要求单步执行该程序，在watch窗口中观察Ik,sum,n值。具体操作如下：

（1）设输入5，观察刚执行完“scanf("%d",&k);”语句时，sum、k的值是多少？

（2）在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值是多少？返回后光条停留在哪个语句上？

（3）在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了哪里？

（4）在fabonacci函数内部单步执行，观察函数的递归执行过程。体会递归方式实现的计算过程是如何完成数计算的，并特别注意什么时刻结束递归，然后直接从第一个return语句返回到了哪里？

（5）在fabonacci函数递归执行过程中观察参数n的变化情况，并回答为什么k、sum在fabonacci函数内部不可见？

**/\*实验3-3跟踪调试题程序：**计算fabonacci数列前n项和**\*/**

#include<stdio.h>

int main(void)

{

int i,k;

long sum=0,fabonacci(int n);

printf("Inut n:");

scanf("%d",&k);

for(i=1;i<=k;i++){

sum+=fabonacci(i);

printf("i=%d\tthe sum is %ld\n",i,sum);

}

return 0;

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

1.在执行"scanf("%d",&k);"后，k的值会被设为输入的值。此时sum的值为0。

2.在从main函数第一次进入fabonacci函数前的一刻，i的值为1，sum的值为0。执行sum += fabonacci(i);后，会进入递归调用，然后光标会停留在第一次调用fabonacci函数的返回值上。这是因为fabonacci(i)是一个函数调用，它的返回值被加到了sum上。

3.在从main函数第一次进入fabonacci函数后的一刻，光条会停留在第一次调用fabonacci函数的返回值上。这是因为fabonacci(i)是一个函数调用，它的返回值被加到了sum上。然后会继续执行for循环，直到下一次调用fabonacci函数。

4.在fabonacci函数内部，递归过程会一直持续，直到n等于1或2。此时，函数会直接从第一个return语句返回，并返回1。这个结束递归的过程是由if(n == 1 || n == 2)这一行代码控制的。当n等于1或2时，函数直接返回1，结束递归。否则，函数会递归调用自身两次，一次以n-1为参数，一次以n-2为参数，并将两者的结果相加。

5.在递归过程中，参数n会逐渐减小。这是因为每次函数调用都会使n减1，直到n等于1或2为止。在fabonacci函数内部，每次函数调用都会使n减1，因此k和sum在内部是不可见的。这是因为k和sum是main函数的变量，而不是fabonacci函数的变量。在递归过程中，每个函数调用都有自己的堆栈帧，这个堆栈帧包含了该函数的所有变量（包括n）。因此，每个函数调用的变量都是独立的，不能访问其他函数调用的变量。

**4．程序设计**

以下（1）至（3）题对应Educoder教学平台 “C语言实验”课程，实验3，第7关实验3-1、第8关实验3-2，以及第9关实验3-3。

（1）编程验证歌德巴赫猜想：任何一个大于等于4的偶数都是两个素数之和。要求设计一个函数，接受形参n，以“n=n1+n2”的形式输出结果，若有多种分解情况，取n1最小的一个输出。

例如：n=6，输出“6=3+3”。

main函数循环接收从键盘输入的整数n，如果n是大于或等于4的偶数，调用上述函数进行验证。

1. 算法流程：

在这个程序中，首先定义函数isPrime来判断一个数是否为素数。然后定义了另一个函数verifyGoldbach遍历所有可能的i值（从2到n/2），并检查是否存在两个素数i和n-i的和等于n。如果找到了这样的两个素数，就输出它们并继续寻找更小的i值。如果找不到任何两个素数的和等于n，会输出无法找到。最后，在main函数中，我们循环接收从键盘输入的整数n，当输入一个大于或等于4的偶数时，我们调用verifyGoldbach函数来验证歌德巴赫猜想。

1. 源程序清单：

#include<stdio.h>

int Isprime(int x)

{

    for (int i = 2; i <= x / 2; i++)

    {

        if (x % i == 0)

            return 0;

    }

    return 1;

}

void Goldbach(int n)

{

    int i;

    for (i = 2; i <= n / 2; i++)

    {

        if (Isprime(i) && Isprime(n - i))

        {

            printf("%d=%d+%d\n", n, i, n - i);

            break;

        }

    }

}

int main()

{

    int n;

    while (scanf("%d", &n) != EOF)

    {

        if(n >= 4 && n % 2 == 0)

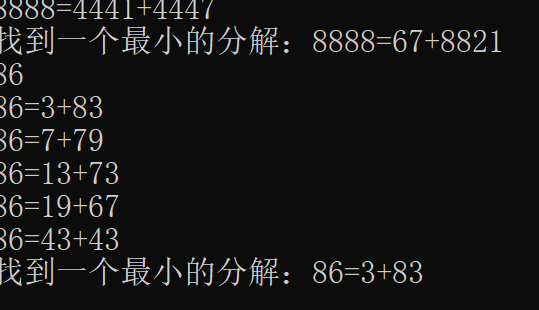
            Goldbach(n);

    }

    return 0;

}

1. 测试：
2. 测试数据：8888 86
3. 相关测试结果如截图所示：



（2）完全数（Perfect number），又称完美数或完备数，特点是它的所有真因子（即除了自身以外的约数，包括1）之和恰好等于它本身。例如6=1+2+3，28=1+2+4+7+14等。

编程寻找10000以内的所有完全数。

要求设计一个函数，判定形参n是否为完全数，如果是，返回1，否则返回0。在main函数中调用该函数求10000以内的所有完全数，并以完全数的真因子之和的形式输出结果，例如“6=1+2+3”。程序输出中，每个完全数单独占一行。

1. 算法流程：

程序首先自定义函数isPerfect来判断一个数是否为完全数，若是返回1，不是则返回0。在主函数中，首先调用isPerfect函数找出10000以内的所有完全数，每找出一个完全数，都会进入一个for循环找出这个完全数的所有真因子，最后以“真因子之和=完全数”的形式打印输出。

1. 源程序清单：

#include<stdio.h>

void PrintComp(int n);

int Iscomp(int n)

{

    int sum = 0;

    for (int i = 1; i < n; i++)

    {

        if (n % i == 0)

            sum += i;

    }

    if (sum == n)

    {

        PrintComp(n);

        return 1;

    }

    else

        return 0;

}

void PrintComp(int n)

{

    printf("%d=1", n);

    for (int i = 2; i < n; i++)

    {

        if (n % i == 0)

            printf("+%d", i);

    }

    printf("\n");

}

int main()

{

    for (int i = 1; i < 10000; i++)

    {

        Iscomp(i);

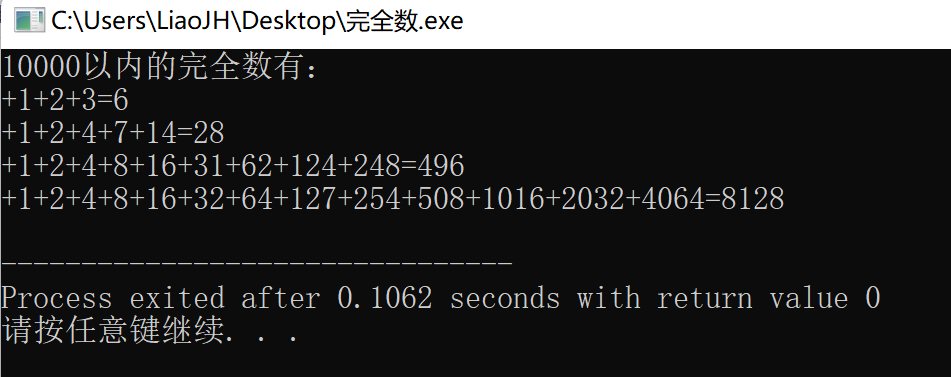
    }

    return 0;

}

1. 测试：

程序运行结果如图：



（3）自幂数是指一个n位数，它的每个位上的数字的n次幂之和等于它本身。水仙花数是3位的自幂数，除此之外，还有4位的四叶玫瑰数、5位的五角星数、6位的六合数、7位的北斗星数、8位的八仙数等。

编写一个函数，判断其参数n是否为自幂数，如果是，返回1；否则，返回0。要求main函数能反复接收从键盘输入的整数k，k代表位数，然后调用上述函数求k位的自幂数，输出所有k位自幂数，并输出相应的信息，例如“3位的水仙花数共有4个153，370，371，407”。当k=0时程序结束执行。

1. 算法流程：

在程序中，is\_self\_power函数判断一个数是否为自幂数，如果是返回1，否则返回0。find\_self\_power函数用于找出所有k位的自幂数并输出结果。在主函数中，程序反复接收从键盘输入的整数k，然后调用find\_self\_power函数求k位的自幂数，并输出相应的信息。当输入的k等于0时，程序结束执行。

1. 源程序清单：

#include<stdio.h>

void Selfmul(int k)

{

    switch (k)

    {

    case(3):

        printf("3位的水仙花数共有4个153,370,371,407\n");

        break;

    case(4):

        printf("4位的四叶玫瑰数共有3个1634,8208,9474\n");

        break;

    case(5):

        printf("5位的五角星数共有3个54748,92727,93084\n");

        break;

    case(6):

        printf("6位的六合数共有1个548834\n");

        break;

    case(7):

        printf("7位的北斗星数共有4个1741725,4210818,9800817,9926315\n");

        break;

    case(8):

        printf("8位的八仙数共有3个24678050,24678051,88593477\n");

        break;

    }

}

int main()

{

    int k;

    while (scanf("%d", &k) && k != 0)

    {

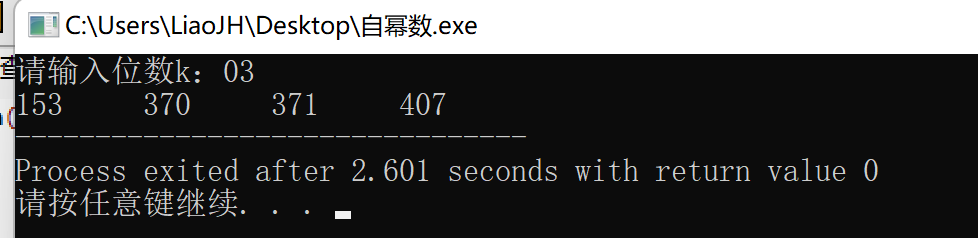
        Selfmul(k);

    }

    return 0;

}

1. 测试
2. 测试数据：3
3. 测试结果：



**3.3实验小结：**

**通过本次实验，我成功地熟悉和掌握了函数的定义、声明、调用和参数传递，以及函数返回值类型的定义和返回值使用。此外，我也深入了解了不同存储类型变量的使用，如自动变量、寄存器变量和静态变量等。最后，我还练习了使用集成开发环境中的调试功能，如单步执行、设置断点和观察变量值等。**

**综上，本次实验帮助我巩固了函数的定义、声明、调用和参数传递等方面的知识，也让我更深入地了解了不同存储类型变量的使用和调试功能的使用。这些知识和技能将提高我的编程效率和代码质量。**



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.11.6**

**软件学院**

**目 录**

**[4 编译预处理 3](#_Toc404837920)**

[4.1 实验目的 3](#_Toc404837921)

[4.2 实验内容 3](#_Toc404837922)

[4.3 实验小结 1](#_Toc404837923)5

# 实验4 编译预处理实验

**4.1实验目的**

（1）掌握文件包含、宏定义、条件编译和assert宏的使用；

（2）练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

（3）熟悉多文件编译技术

4.2实验内容

**1．程序改错**

下面是用宏来计算平方差、交换两数的源程序.在这个源程序中存在若干错误，要求对该程序进行调试修改，使之能够正确完成指定任务。

/\*实验4-1改错与跟踪调试题程序：计算平方差、交换两数\*/

1 #include<stdio.h>

2 #define SUM a+b

3 #define DIF a-b

4 #define SWAP(a,b) a=b,b=a

5 int main()

6 {

7 int a,b;

8 printf("Input two integers a, b:");

9 scanf("%d%d", &a,&b);

10 printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM\*DIF);

11 SWAP(a,b);

12 printf("\nNow a=%d,b=%d\n",a,b);

13 return 0;

14 }

解答：

1. 错误修改：
2. 第2行和第3行宏定义的参数运算结果没有带括号导致后面计算平方差出错， 正确形式为：#define SUM （（a）+（b）） #define DIF （（a）-（b））
3. 第4行的宏定义SWAP无法完成交换两数的功能，可以多引入一个中间变量t，修改为：

#define SWAP(a,b,t) t=a,a=b,b=t

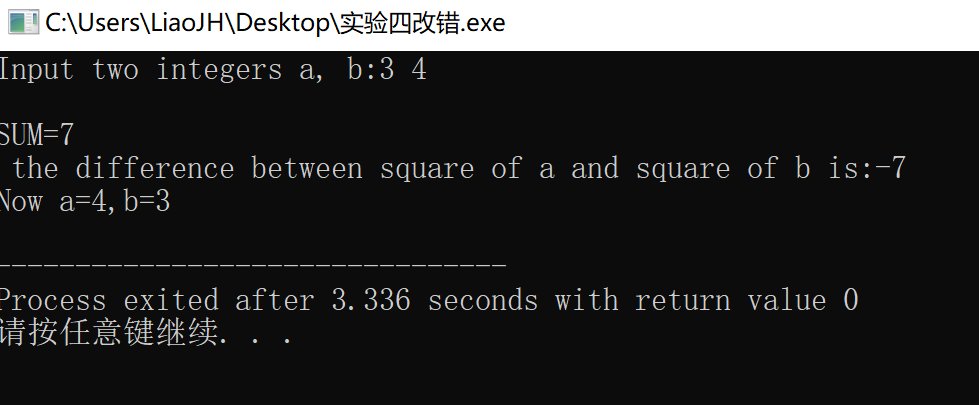
3)与之相对应的，第7行还要引入变量t，修改为：

int a,b,t;

1. 第7行使用宏定义作相应修改：

SWAP(a,b,t)

1. 错误修改后运行结果：



**2．程序修改替换**

下面是用函数实现求三个数中最大数、计算两浮点数之和的程序。在这个源程序中存在若干语法和逻辑错误。

要求：（1）对这个例子程序进行调试修改，使之能够正确完成指定任务；

（2）用带参数的宏替换函数max，来实现求最大数的功能。

/\*实验4-2程序修改替换题程序\*/

#include<stdio.h>

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

scanf("%d %d %d",&a,&b,&c);

printf("\nThe maximum of them is %d\n",max(a,b,c));

printf("Input two floating point numbers:");

scanf("%f %f",&d,&e);

printf("\nThe sum of them is %f\n",sum(d,e));

return 0;

}

int max(int x, int y, int z)

{

int m=z;

if (x>y)

if(x>z) m=x;

else

if(y>z) m=y;

return m;

}

float sum(float x, float y)

{

return x+y;

}

**解答：**

替换后的程序如下所示：

#include<stdio.h>

#define MAX(x,y,z) (((x)>(y))&&((x)>(z))?(x):((y)>(x))&&((y)>(z))?(y):(z))

float sum(float x, float y);

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

scanf("%d %d %d",&a,&b,&c);

printf("\nThe maximum of them is %d\n",MAX(a,b,c));

printf("Input two floating point numbers:");

scanf("%f %f",&d,&e);

printf("\nThe sum of them is %f\n",sum(d,e));

return 0;

}

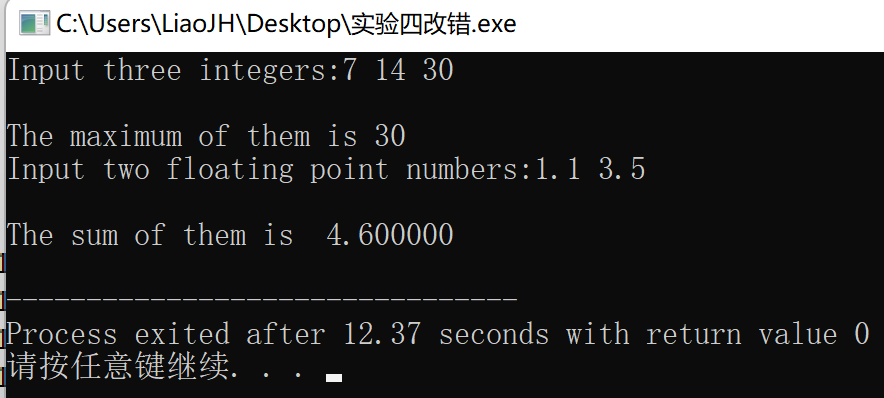
float sum(float x, float y)

{

return x+y;

}

运行结果如下：



**3．跟踪调试**

下面程序利用R计算圆的面积s，以及面积s的整数部分。现要求：

（1）修改程序，使程序编译通过且能运行；

（2）单步执行。进入函数integerl\_fraction时，watch窗口中x为何值？在返回main时, watch窗口中i为何值？

（3）修改程序，使程序能输出面积s值的整数部分（要求四舍五入），不会输出错误信息assertion failed。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

#define R

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

assert((s-s\_integer)<0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

解答：

1. 程序改错：

首先需要使用#include指令包含<stdio.h>和<assert.h>两个头文件，然后在main函数前对定义的int integer\_fraction(float x)函数进行声明，即可通过编译。（但没有办法四舍五入取整数）

#define R

#include <stdio.h>

#include <assert.h>

int integer\_fraction(float x);

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

assert((s-s\_integer)<0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

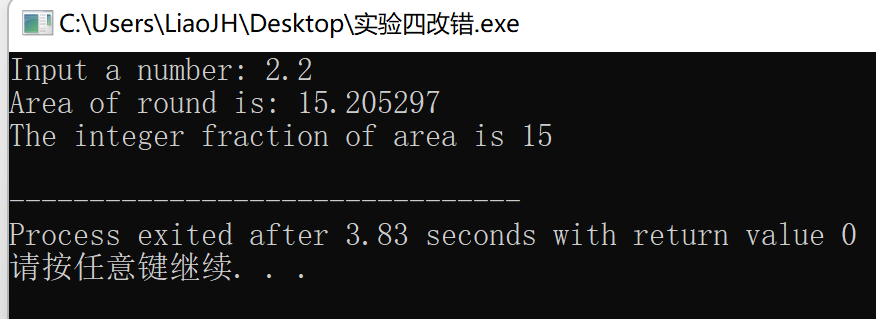
{

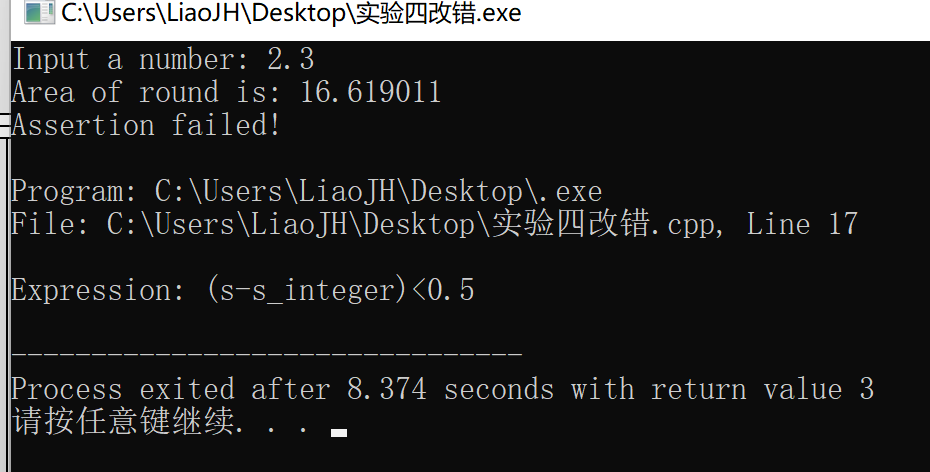
int i=x;

return i;

}

程序运行结果：（测试数据2.2和2.3）





1. 单步执行：

当进入函数integer\_fraction时，watch窗口中x为何值？

π\*r^2 = (3.14159)^3

在返回main时, watch窗口中i为何值？

等于上面的π\*r^2。

（3）程序修改：

删去assert，用if语句来写四舍五入，当(s-s\_integer)>=0.5时，使s-s\_integer加一即可。

#define R

#include <stdio.h>

int integer\_fraction(float x);

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

if((s-s\_integer)>=0.5)

s\_integer++;

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

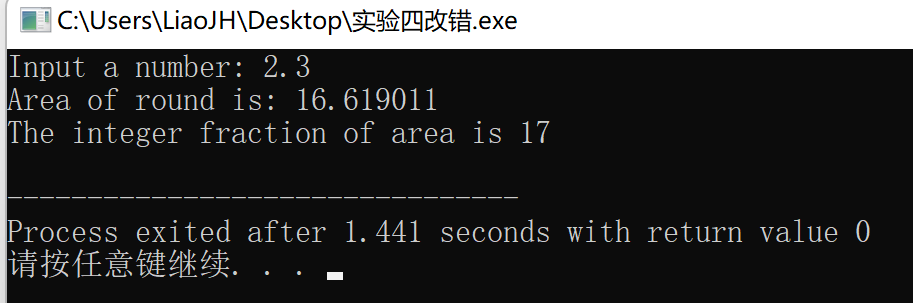
{

int i=x;

return i;

}

程序修改后运行结果：



**4．程序设计**

（1）三角形的面积是，其中，a,b,c为三角形的三边，要求编写程序用带参数的宏来计算三角形的面积。定义两个带参数的宏，一个用来求s，另一个用来求area。

1. 算法流程：

定义两个带参数的宏S来求s和AREA用来求area。

2）源程序清单：

#include <stdio.h>

#include <math.h>

#define S(a, b, c) ((a + b + c) / 2)

#define AREA(a, b, c) (sqrt(S(a, b, c) \* (S(a, b, c) - a) \* (S(a, b, c) - b) \* (S(a, b, c) - c)))

int main() {

double a, b, c;

printf("请输入三角形的三边长：\n");

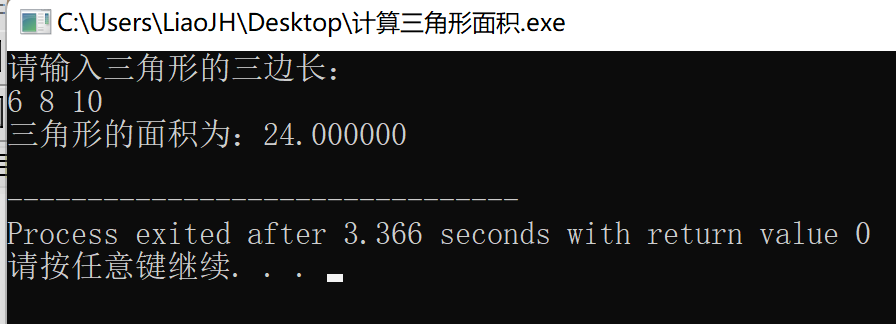
scanf("%lf %lf %lf", &a, &b, &c);

printf("三角形的面积为：%lf\n",AREA(a, b, c));

return 0;

}

1. 程序运行结果：



1. 用条件编译方法来编写程序。输入一行英文字符序列，可以任选两种方式之一输出：一为原文输出；二为变换字母的大小写后输出。例如小写‘a’变成大写‘A’，大写‘D’变成小写‘d’，其他字符不变。用#define命令控制是否变换字母的大小写。例如，#define CHANGE 1 则输出变换后的文字，若#define CHANGE 0则原文输出。
2. 算法流程：

用条件编译来控制输出方式。当定义了CHANGE为1时，程序会转换输入字符串中的字母大小写并输出；否则，程序会直接输出原文。要想输出想要的文本，只需调整宏CHANGE的值。

1. 源程序清单：

#include <stdio.h>

#include <ctype.h>

#define CHANGE 1

int main() {

char str[100];

printf("请输入一行英文字符序列：\n");

fgets(str, sizeof(str), stdin);

#if CHANGE

for (int i = 0; str[i] != '\0'; i++) {

if (islower(str[i])) {

str[i] = toupper(str[i]);

} else if (isupper(str[i])) {

str[i] = tolower(str[i]);

}

}

#endif

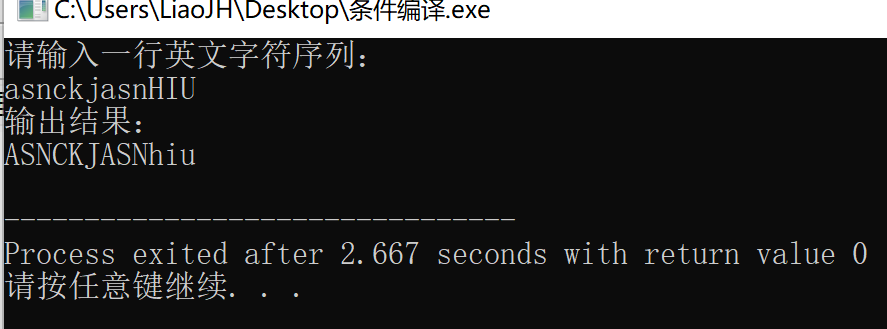
printf("输出结果：\n%s", str);

return 0;

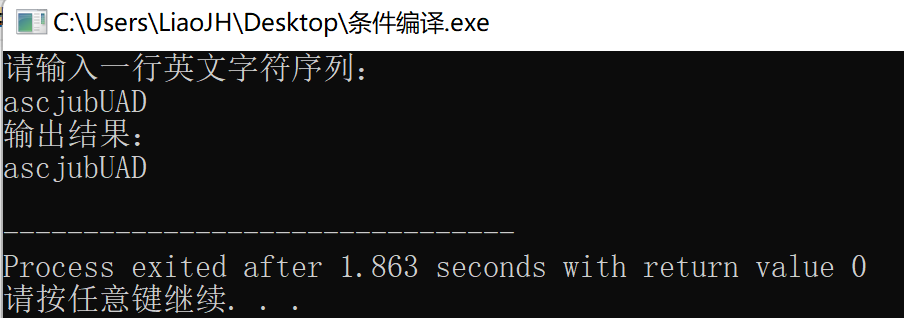
}

1. 运行结果：

A.当宏CHANGE定义为1时：



1. 当宏CHANGE为0时：



（3）假设一个C程序由file1.c和file2.c两个源文件及一个file.h头文件组成，file1.c、file2.c和file.h的内容分别如下所述。试编辑该多文件C程序，补充file.h头文件内容，然后编译和链接。然后运行最后生成的可执行文件。

/\*源文件file1.c的内容\*/

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);

func1();

return 0;

}

/\*源文件file2.c的内容为：\*/

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

1）file.h头文件内容：

extern int x, y;

extern char ch;

void func1(void);

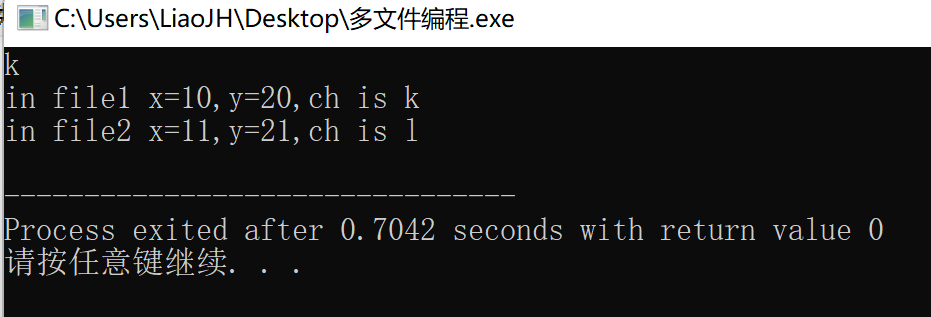
1. 编译链接生成program文件：

gcc -c file1.c

gcc -c file2.c

gcc -o program file1.o file2.o

1. 执行文件：



**4.3 实验小结：**

在实验过程中，我使用了C语言的文件包含和宏定义功能来简化代码编写和方便程序维护。通过条件编译，我能够根据不同的编译条件来生成不同的代码，增强了程序的灵活性和可维护性。Assert宏则用于在程序中添加调试断言，帮助我发现潜在的错误和问题。

总之，本次实验使我更有能力编写高质量的程序。



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.11.13**

**软件学院**

**目 录**

[5](#_Toc404837920) **[数组实验 3](#_Toc404837920)**

[5.1 实验目的 3](#_Toc404837921)

[5.2 实验内容 3](#_Toc404837922)

[5.3 实验小结 1](#_Toc404837923)2

实验5 数组实验

5.1 实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

5.2 实验内容及要求

1、 源程序改错与跟踪调试

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgramingLnguage

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。

解答：

1）进跟踪进入strcate时，字符数组t的内容为字符串“Programming”，s的内容为字符串“Language”，结果正确；

2）当单步执行光条刚落在第二个while语句所在行时，i为12，t[i]为0‘000/’,结果错误，数组越界访问导致t[i]出现随机值；

3）当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t的内容为字符串“Programming”，s的内容为字符串“Language”，并未实现字符串连接。

1. 跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。

解答：

1. 进入函数strdelc时，字符数组s中的内容为字符串“Programming”，字符c的值为‘a’，结果正确；
2. 问题：最后一个字符‘g’被多重复一次；
3. 当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为“Progrmmingg”，未完全实现删除要求。

/\*实验5程序改错与跟踪调试题程序\*/

#include<stdio.h>

void strcate(char [],char []);

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[]="Programming";

printf("%s %s\n", b,a);

strcate(b,a); printf("%s %s\n",b,a);

strdelc(b, 'a'); printf("%s\n",b);

return 0;

}

void strcate(char t[],char s[])

{

int i = 0, j = 0;

while(t[i++]) ;

while((t[i++] = s[j++] )!= '\0');

}

void strdelc(char s[], char c)

{

int j,k;

for(j=k=0; s[j] != '\0'; j++)

if(s[j] != c) s[k++] = s[j];

}

2、 源程序完善和修改替换

(1) 下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/

if(j)

for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

printf(“%6d”, b[i]);

printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

解答：

补充完整后的程序：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

b[M-i] = j ? a[j-1]:a[1]; /\* 将报数为N的人的编号存入数组b \*/

if(j)

for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

a[k]=a[k+1];

}

for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

printf("%6d", b[i]);

printf("%6d\n", a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

②上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

解答：

增加了一个数组 mark用来做标记。初始时，所有的标记都为0，表示未出圈。每当有人出圈时，就将相应位置的标记设置为1。在压缩数组的过程中，如果发现当前位置的人已经被标记为出圈，就跳过压缩步骤。这样就可以省掉压缩数组的时间。最后，遍历 mark 数组，输出所有出圈人的编号即可。

#include <stdio.h>

#define M 10

#define N 3

int main(void) {

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int mark[M]; /\* 用来做标记的数组 \*/

int i, j, k;

for (i = 0; i < M; i++) {

/\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

mark[i] = 0; /\* 初始标记为0，表示未出圈 \*/

}

for (i = M, j = 0; i > 1; i--) { /\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for (k = 1; k <= N; k++) { /\* 1至N报数 \*/

if (++j > i - 1) {

j = 0; /\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

}

if (mark[j]) { /\* 如果当前位置的人已经出圈，则不做标记 \*/

b[M - i] = j ? a[j - 1] : a[1]; /\* 将报数为N的人的编号存入数组b \*/

mark[j] = 0; /\* 将当前位置的人标记为未出圈 \*/

} else {

mark[j] = 1; /\* 将当前位置的人标记为已出圈 \*/

}

}

}

for (i = 0; i < M - 1; i++) { /\* 按次序输出出圈人的编号 \*/

printf("%6d", b[i]);

}

printf("%6d\n", a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

3. 程序设计

以下（1）至（3）题对应Educoder教学平台 “C语言实验”课程，实验5，第10关实验5-1、第11关实验5-2，以及第12关实验5-3。

1. 输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。
2. 算法流程：

程序的思路是先将整数转换为二进制字符串，然后将字符串反转，最后输出反转后的二进制字符串。在转换过程中，使用位运算将每一位的值提取出来，并将其转换为对应的数字字符。

1. 源程序清单：

#include<stdio.h>

#include<stdio.h>

int main()

{

    int a, i;

    scanf("%d", &a);

    char b[33] = { 0 };

    for (i = 31; i >= 0; i--)

    {

        b[i] = (a & 1) + '0';

        a = a >> 1;

    }

    b[32] = '\0';

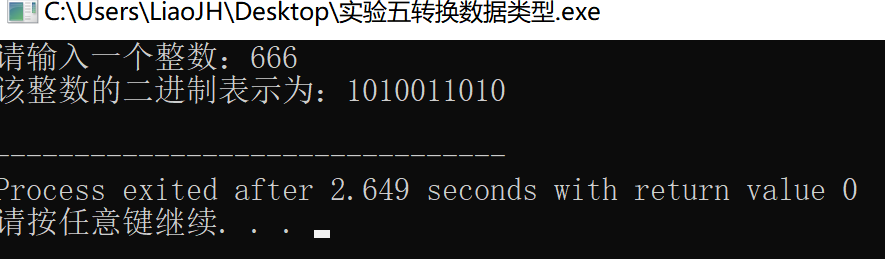
    for (i = 0; i <32; i++)

        printf("%c", b[i]);

    return 0;

}

3）程序运行结果：



（2）编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：

①“成绩输入”，输入n个学生的姓名和C语言课程的成绩。

②“成绩排序”，将成绩按从高到低的次序排序，姓名同时进行相应调整。

③“成绩输出”，输出排序后所有学生的姓名和C语言课程的成绩。

④“成绩查找”，输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

1. 算法流程：
2. 源程序清单：

#include<stdio.h>

struct student

{

    char name[20];

    int grade;

};

struct student\* st;

void Ginput(int base, int len)

{

    char iname[20];

    int igrade;

    struct student\* str = (struct student\*)realloc(st, len \* sizeof(struct student));

    if (str == NULL)

    {

        printf("malloc failed!\n");

        exit(-1);

    }

    st = str;

    str = NULL;

    for (int i = base; i < len; i++)

    {

        scanf("%s %d", &iname, &igrade);

        strcpy(st[i].name,iname);

        st[i].grade = igrade;

    }

    printf("%d records were input!\n", len - base);

}

void Gsort(struct student\* st, int len)

{

    struct student tmp;

    for (int i = 0; i < len; i++)

    {

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

        {

            if (st[i].grade < st[j].grade)

            {

                tmp = st[i];

                st[i] = st[j];

                st[j] = tmp;

            }

        }

    }

    printf("Reorder finished!\n");

}

void Goutput(struct student\* st, int len)

{

    for (int i = 0; i < len; i++)

    {

        printf("%s %d\n", st[i].name, st[i].grade);

    }

}

void Gsearch(struct student\* st, int left, int len, int tar)

{

    if (left > len)

    {

        printf("not found!\n");

        return;

    }

    int i = (left + len) / 2;

    if (st[i].grade > tar)

        Gsearch(st, i + 1, len, tar);

    else if (st[i].grade < tar)

        Gsearch(st, left, i - 1, tar);

    else

    {

        printf("%s %d\n", st[i].name, st[i].grade);

        for (int j = i - 1; j > left && st[j].grade == tar; j--)

            printf("%s %d\n", st[j].name, st[j].grade);

        for (int j = i + 1; j < len && st[j].grade == tar; j++)

            printf("%s %d\n", st[j].name, st[j].grade);

    }

    return;

}

int main()

{

    int grade[100];

    int i, len = 0, search, n;

    while(1)

    {

        scanf("%d", &i);

        switch (i)

        {

        case 1:

            scanf("%d", &n);

            Ginput(len, len + n);

            len += n;

            break;

        case 2:

            Gsort(st, len);

            break;

        case 3:

            Goutput(st, len);

            break;

        case 4:

            scanf("%d", &search);

            Gsearch(st, 0, len, search);

            break;

        case 0:

            free(st);

            st = NULL;

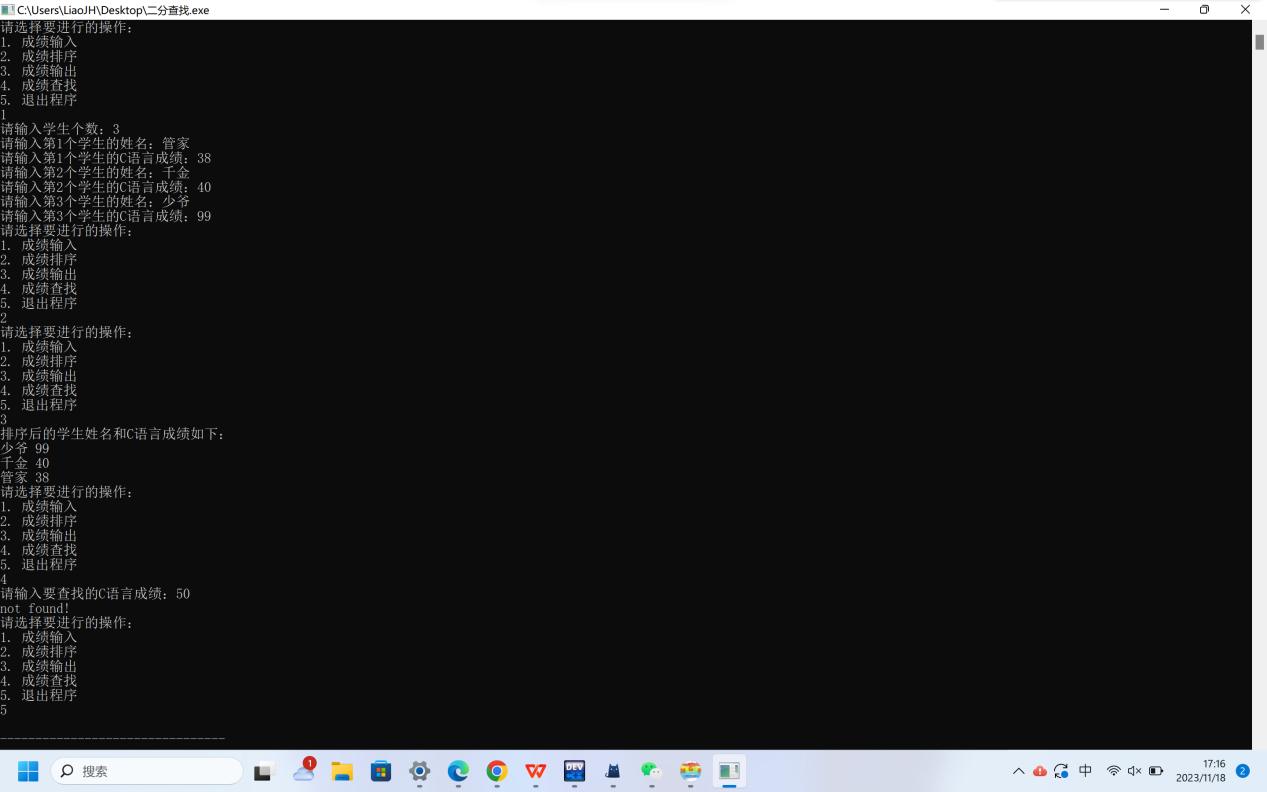
            return 0;

        }

    }

}

1. 程序运行结果：



1. 求解N皇后问题，即在N×N的棋盘上摆放N个皇后，要求任意两个皇后不能在同一行、同一列、同一对角线上。输入棋盘的大小N（N取值1-10），如果能满足摆放要求，则输出所有可能的摆放法的数量，否则输出“无解。”
2. 算法流程：

程序使用了一个二维数组 q 来表示棋盘，其中 q[i][j] 表示第 i 行、第 j 列上的皇后是否放置。初始时，所有位置都为空，即 q[i][j] = 0。如果某个位置没有被占据，则 q[i][j] 的值为 0；如果某个位置被占据，则 q[i][j] 的值为 1。在搜索过程中，如果某个位置可以被占据，则将该位置设置为 1，然后继续搜索下一行。如果当前位置不能被占据，则回溯到上一行，将当前位置设置为 0，然后继续搜索下一行。当搜索到最后一行时，说明找到了一种可行的摆放方法，此时将计数器加一。最终输出计数器的值即可。

1. 源程序清单：

**#include<stdio.h>**

**#include<stdio.h>**

**#include <stdbool.h>**

**#include <malloc.h>**

**#include <math.h>**

**int N;**

**int totalSolutions;**

**bool isSafe(int\* board, int x, int y)**

**{**

**for (int i = 0; i < x; i++)**

**{**

**if (board[i] == y || abs(i - x) == abs(board[i] - y))**

**{**

**return false;**

**}**

**}**

**return true;**

**}**

**void solveNQueensUtil(int\* board, int x)**

**{**

**if (x == N)**

**{**

**totalSolutions++;**

**}**

**else**

**{**

**for (int y = 0; y < N; y++)**

**{**

**if (isSafe(board, x, y))**

**{**

**board[x] = y;**

**solveNQueensUtil(board, x + 1);**

**}**

**}**

**}**

**}**

**void solveNQueens()**

**{**

**int\* board = (int\*)malloc(N \* sizeof(int));**

**totalSolutions = 0;**

**solveNQueensUtil(board, 0);**

**if (totalSolutions != 0)**

**{**

**printf("%d\n", totalSolutions);**

**}**

**else**

**printf("无解\n");**

**free(board);**

**}**

**int main()**

**{**

**scanf("%d", &N);**

**solveNQueens();**

**return 0;**

**}**

**5.3实验小结：**

在本次实验，我掌握了数组的基本概念和操作，了解了如何声明、初始化和使用数组。学会了如何将一维数组作为函数参数，理解了实参和形参的用法，以及它们在函数调用过程中的传递方式。熟悉了一些常用的字符串处理函数的设计方法，理解了二分查找算法和选择法排序算法的基本思想，并掌握了它们的实现方法。这些算法在数据处理中有着广泛的应用，对于提高程序的效率和性能具有重要意义。



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.11.27**

**软件学院**

**目 录**

**[6 指针实验 3](#_Toc404837920)**

[6.1 实验目的 3](#_Toc404837921)

[6.2 实验内容 3](#_Toc404837922)

[6.3 实验小结 2](#_Toc404837923)4

实验6 指针实验

6.1、实验目的

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

6.2、实验题目及要求

1、源程序改错题

在下面所给的源程序中，函数strcopy(t, s)的功能是将字符串s复制给字符串t，并且返回串t的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:

programming↙ （键盘输入）

programming

Input a string again:

language↙ （键盘输入）

language

#include<stdio.h>

char \*strcopy(char \*, const char \*);

int main(void)

{

char \*s1, \*s2, \*s3;

printf("Input a string:\n", s2);

scanf("%s", s2);

strcopy(s1, s2);

printf("%s\n", s1);

printf("Input a string again:\n", s2);

scanf("%s", s2);

s3 = strcopy(s1, s2);

printf("%s\n", s3);

return 0;

}

/\*将字符串s复制给字符串t，并且返回串t的首地址\*/

char \* strcopy(char \*t, const char \*s)

{

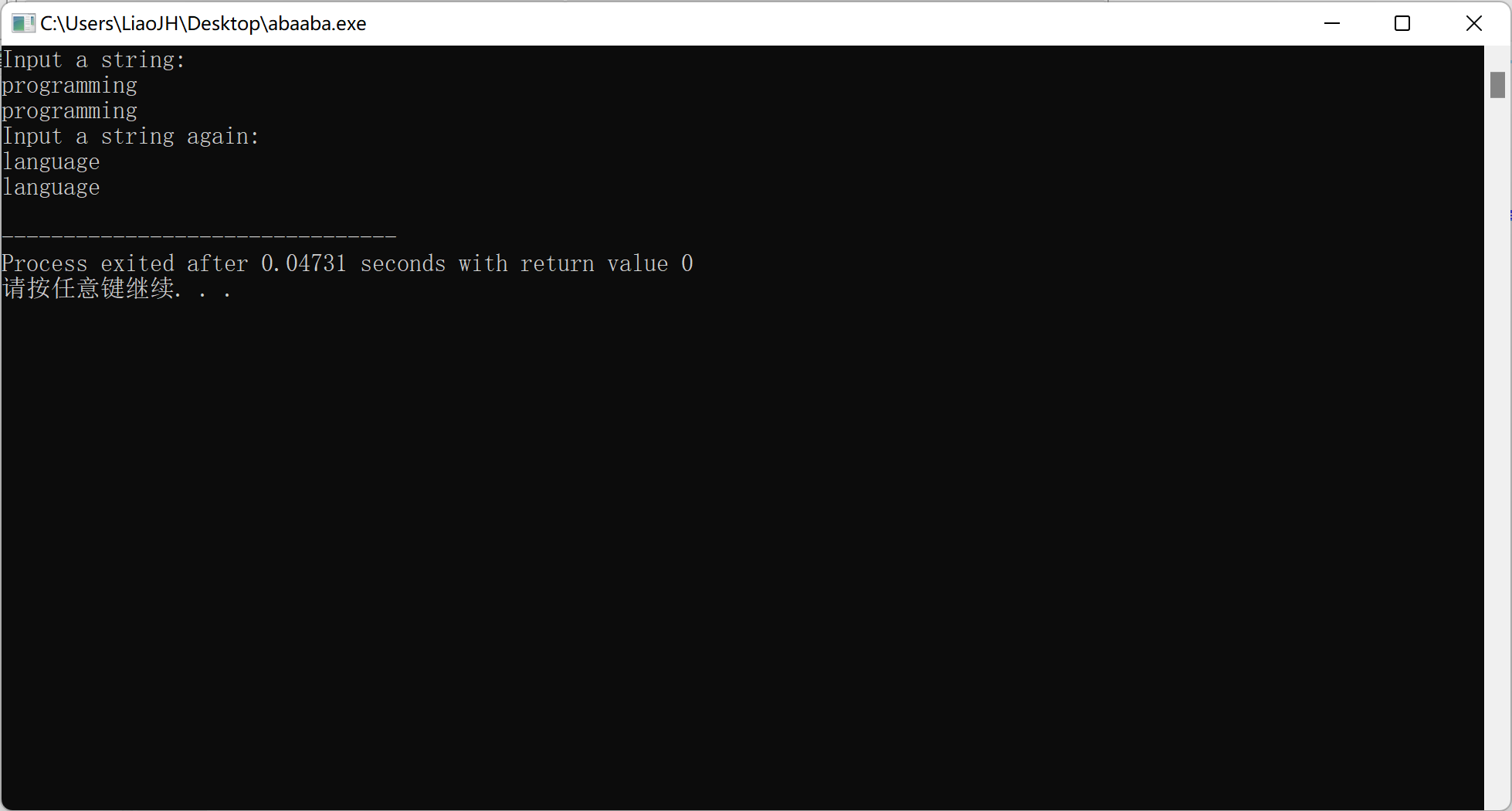
while(\*t++ = \*s++);

return (t);

}

解答：

1. 错误修改：
2. 初始化定义三个数组；
3. 删去两个printf中的s2；
4. 运行结果：



2、源程序完善、修改替换题

（1）下面程序中函数strsort用于对字符串进行升序排序，在主函数中输入N个字符串（字符串长度不超过49）存入通过malloc动态分配的存储空间，然后调用strsort对这N个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#include<\_\_\_\_\_\_\_\_\_>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

\_\_\_\_\_\_\_temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (\_\_\_\_\_\_\_\_\_\_\_)

{

temp = s[j];

\_\_\_\_\_\_\_\_\_\_\_;

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

for (i=0; i<N; i++)

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(\_\_\_\_\_\_\_);

}

strsort(\_\_\_\_\_\_\_\_);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

解答：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

char \*temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (strcmp(s[j],s[j+1])>0)

{

temp = s[j];

s[j]=s[j+1];

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

for (i=0; i<N; i++)

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(s[i],t);

}

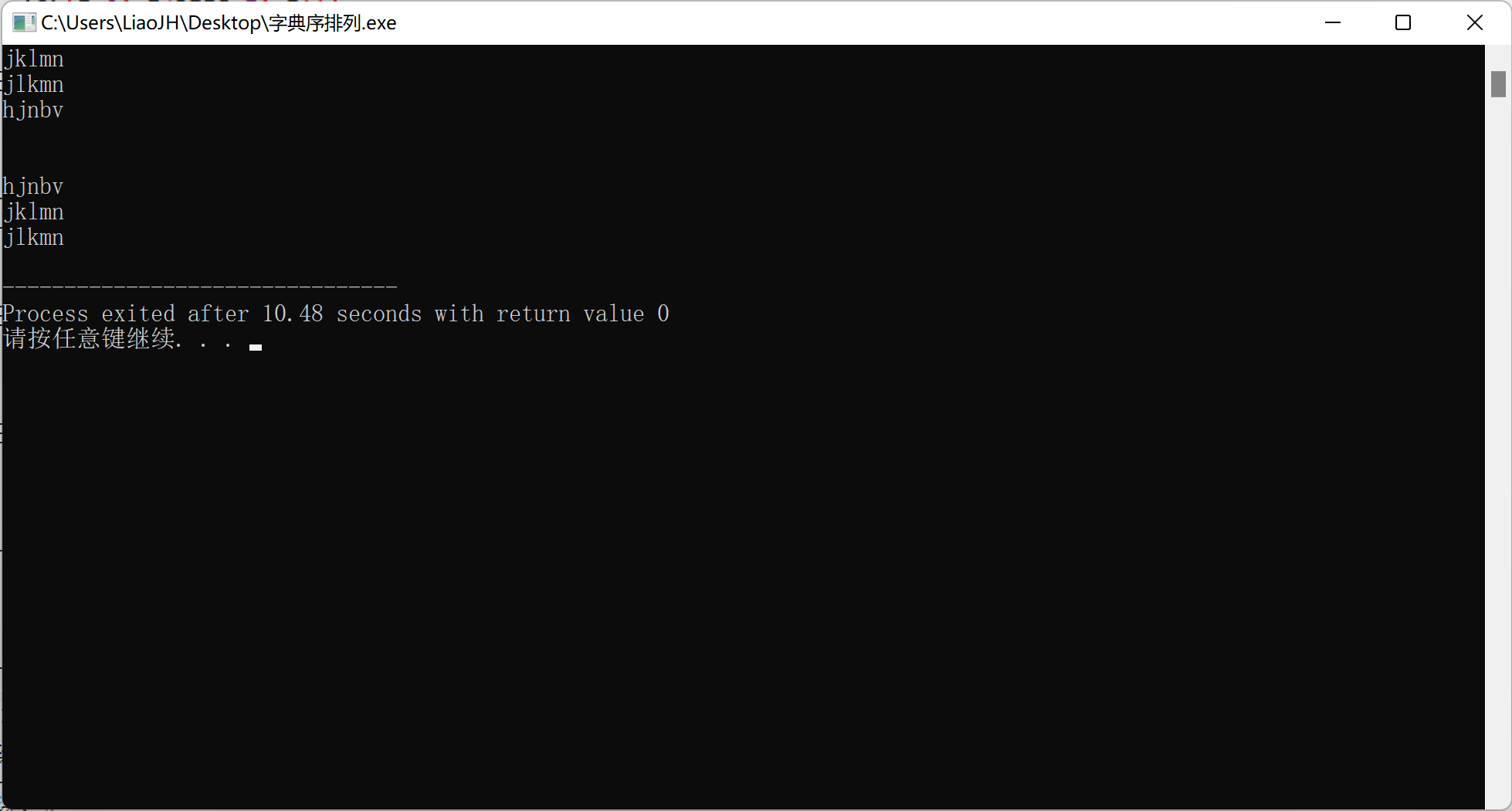
strsort(s,N);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

运行结果:



②数组作为函数参数其本质类型是指针。例如，对于形参char \*s[]，编译器将其解释为char \*\*s，两种写法完全等价。请用二级指针形参重写strsort函数，并且在该函数体的任何位置都不允许使用下标引用。

解答：直接使用二级指针char \*\*s作为形参，这样就可以通过指针来访问字符串数组中的字符串，而不需要使用下标引用。

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void strsort(char \*\*s, int size) {

char \*temp;

int i, j;

for (i = 0; i < size - 1; i++) {

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

if (strcmp(s[i], s[j]) > 0) {

temp = s[i];

s[i] = s[j];

s[j] = temp;

}

}

}

}

int main() {

int i;

char \*s[4], t[50];

for (i = 0; i < 4; i++) {

fgets(t, sizeof(t), stdin);

s[i] = (char \*)malloc(strlen(t) + 1);

strcpy(s[i], t);

}

strsort(s, 4);

for (i = 0; i < 4; i++) {

puts(s[i]);

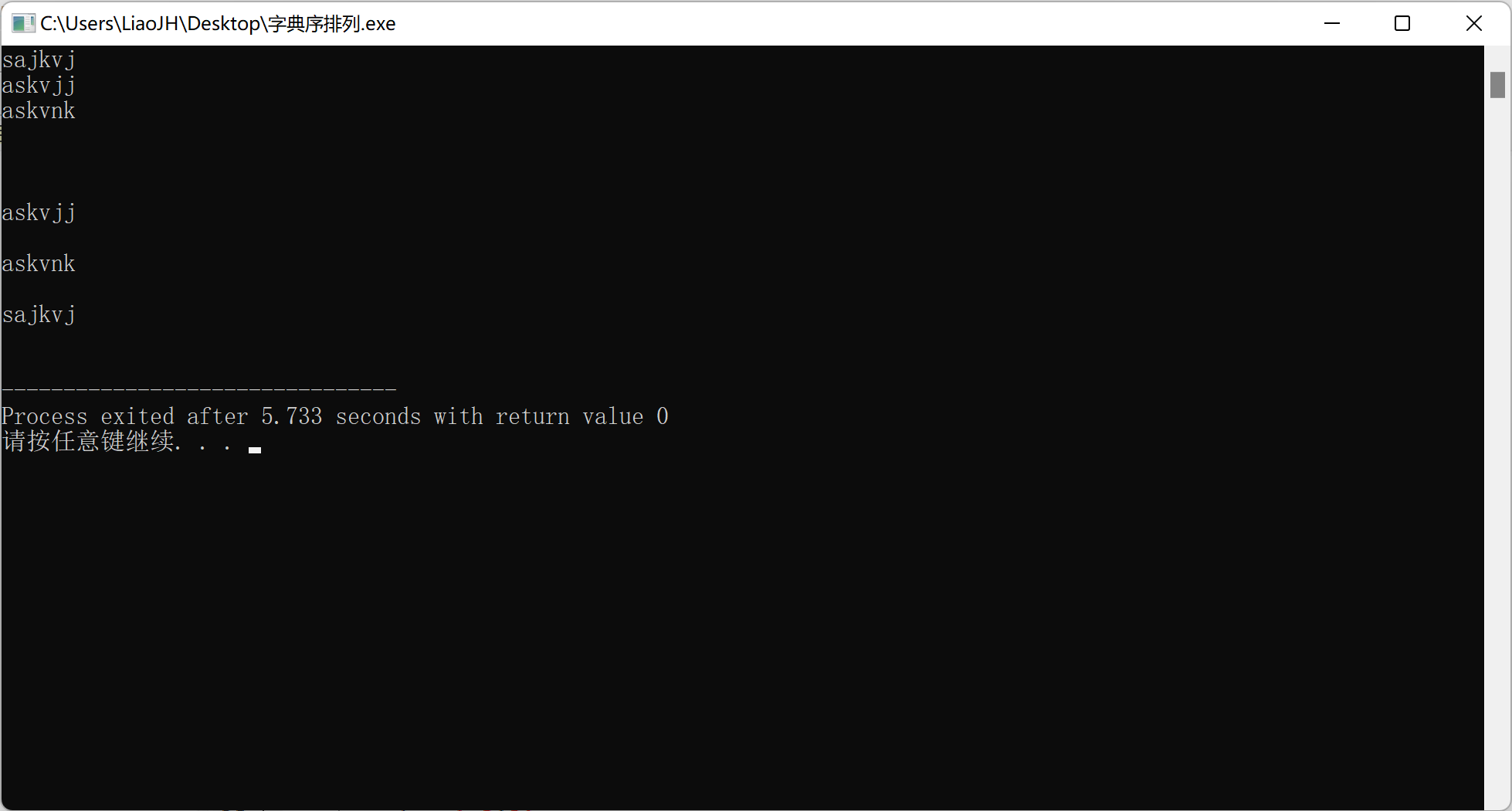
free(s[i]);

}

return 0;

}

运行结果：



（2）下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

①请在源程序中的下划线处填写合适的代码来完善该程序，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

解答：

# include<stdio.h>

# include<string.h>

int main (void)

{

char \*p;

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: char (\*p)(char\*,char\*) = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

fgets(a,sizeof(a),stdin);

printf("input the second string please!\n");

fgets(b,sizeof(b),stdin);

result =p(a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

解答：

# include<stdio.h>

# include<string.h>

int main (void)

{

char \*p;

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

fgets(a,sizeof(a),stdin);

printf("input the second string please!\n");

fgets(b,sizeof(b),stdin);

result =p(a, b);

printf("the result is %s\n", result);

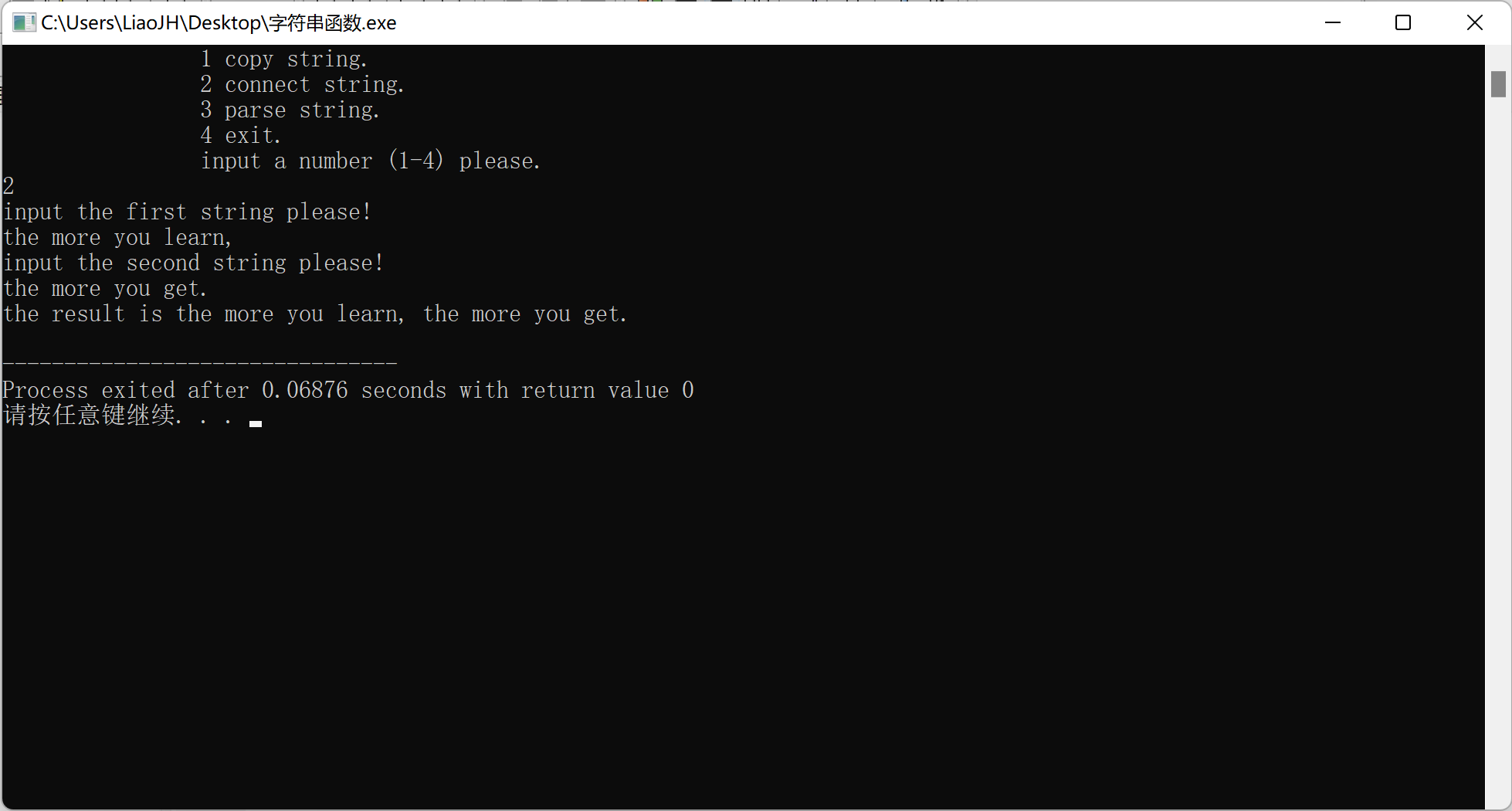
}

down:

return 0;

}

运行结果：



②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

解答：

定义一个函数指针数组转移表table，其中每个元素都是一个函数指针，指向不同的字符串处理函数。然后，在主循环中，使用输入的数字来选择要执行的函数，并通过函数指针数组来调用相应的函数。

#include <stdio.h>

#include <string.h>

int main(void) {

char \*p;

char a[80], b[80], \*result;

int choice;

int (\*table[])(char\*, char\*) = {0};

table[1] = strcpy;

table[2] = strcat;

table[3] = strtok;

while (1) {

do {

printf("\t\t1. copy string.\n");

printf("\t\t2. connect string.\n");

printf("\t\t3. parse string.\n");

printf("\t\t4. exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

} while (choice < 1 || choice > 4);

p = table[choice](a, b);

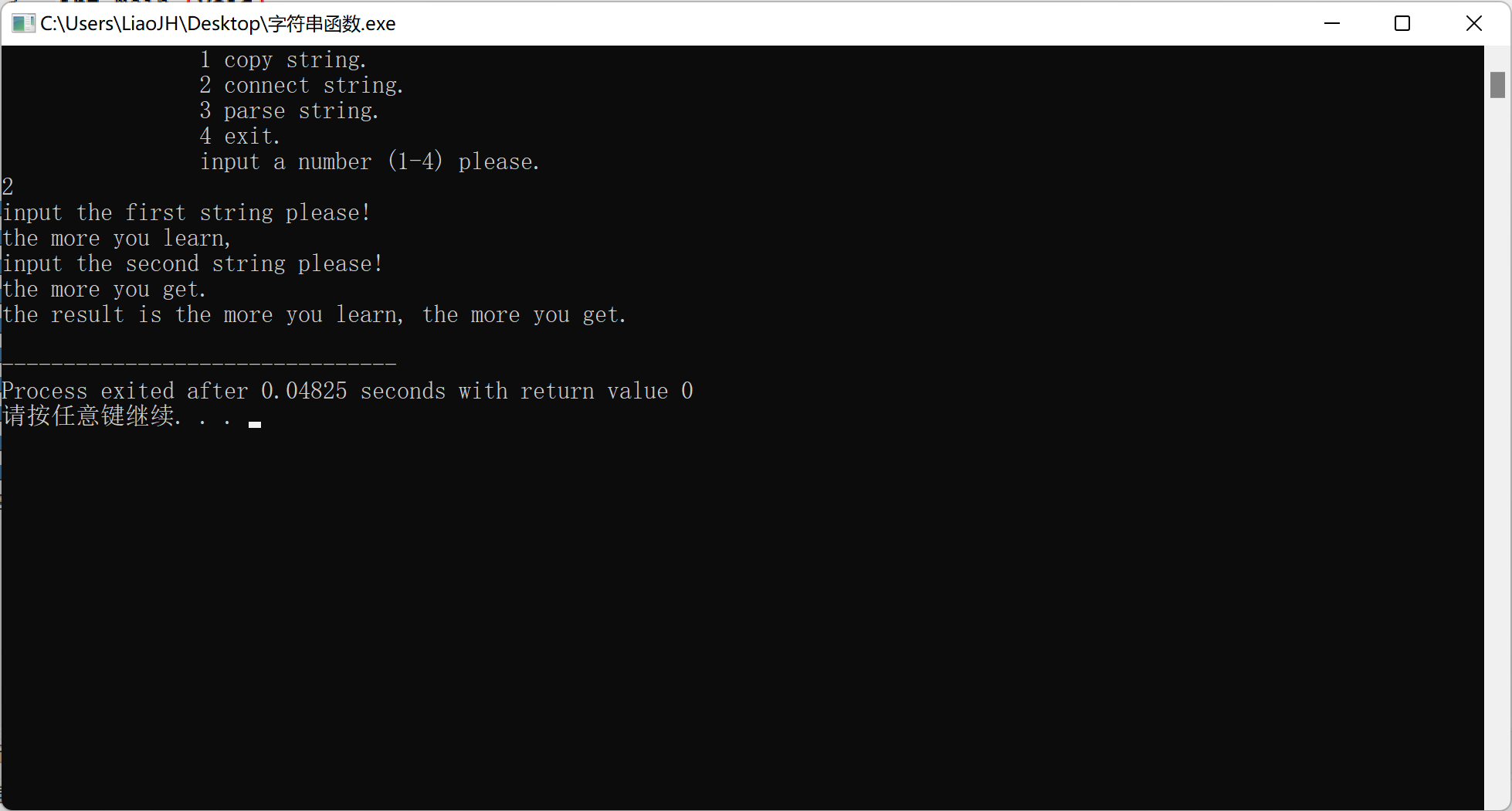
printf("the result is %s\n", p);

}

return 0;

}

运行结果：



3、编程设计题

（1）指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

1. 算法流程：在 “传递给主程序的参数”文本框中输入main函数的参数arg1 arg2 arg3，通过 argv[1] argv[2] argv[3]来输出三个参数
2. 源程序清单：

#include <stdio.h>

int main(int argc, char \*argv[]) {

printf("arg1: %s\n", argv[1]);

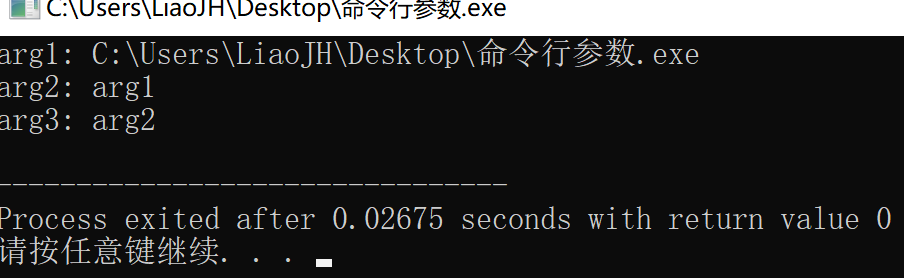
printf("arg2: %s\n", argv[2]);

printf("arg3: %s\n", argv[3]);

return 0;

}

1. 运行结果：



（2）一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。输入一个长整型变量，要求从高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示，通过指针取出每字节。

样例输入：15

样例输出：0000000F

1. 算法流程：利用char指针取出长整型数的字节数据，再利用位运算提取字节数据的高4位和低4位。
2. 源程序清单：

 #include<stdio.h>

int main()

{

    long int a;

    scanf("%ld", &a);

    char arr[32];

    for (int i = 31; i >= 0; i--)

    {

        arr[i] = (a & 1) + '0';

        a >>= 1;

    }

    char\* p = arr;

    while (p < arr + 32)

    {

        int result = (p[0] - '0') \* 8 + (p[1] - '0') \* 4 + (p[2] - '0') \* 2 + (p[3] - '0');

        printf("%X", result);

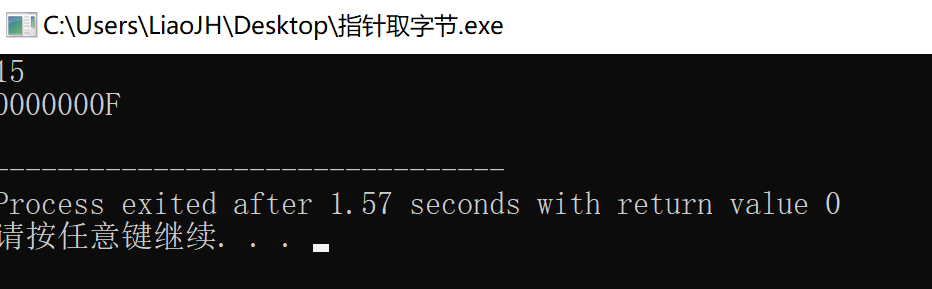
        p += 4;

    }

    return 0;

}

1. 运行结果：



1. 旋转是图像处理的基本操作，编程实现将一个图像逆时针旋转90°。

提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像。输出原始矩阵逆时针旋转90°后的矩阵。

样例输入：

2 3

1 5 3

3 2 4

样例输出：

3 4

5 2

1 3

1. 算法流程：

首先从标准输入中读入整数 n 和 m表示矩阵的行数和列数。然后，程序定义一个数组 a存储输入的矩阵元素，调用 spin 函数，使用两个嵌套的循环遍历整个矩阵，读入矩阵的每一个元素，使用另外两个嵌套的循环遍历整个矩阵，并按照特定的规则输出矩阵中的元素。具体来说，输出的元素是按照从右下角到左上角的对角线顺序输出的，输出的每一个元素后面都跟着一个空格，除了最后一个元素后面不跟空格。每输出一行就换行。

1. 源程序清单：

# include<stdio.h>

int n, m;

void spin(int\* p) {

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

{

for (int j = 0; j < m; j++)

scanf("%d", p + (i \* m + j));

}

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

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

printf("%d", \*(p + (j \* m + m - 1 - i)));

if (j != n - 1)printf(" ");

}printf("\n");

}

}

int main() {

int a[100];

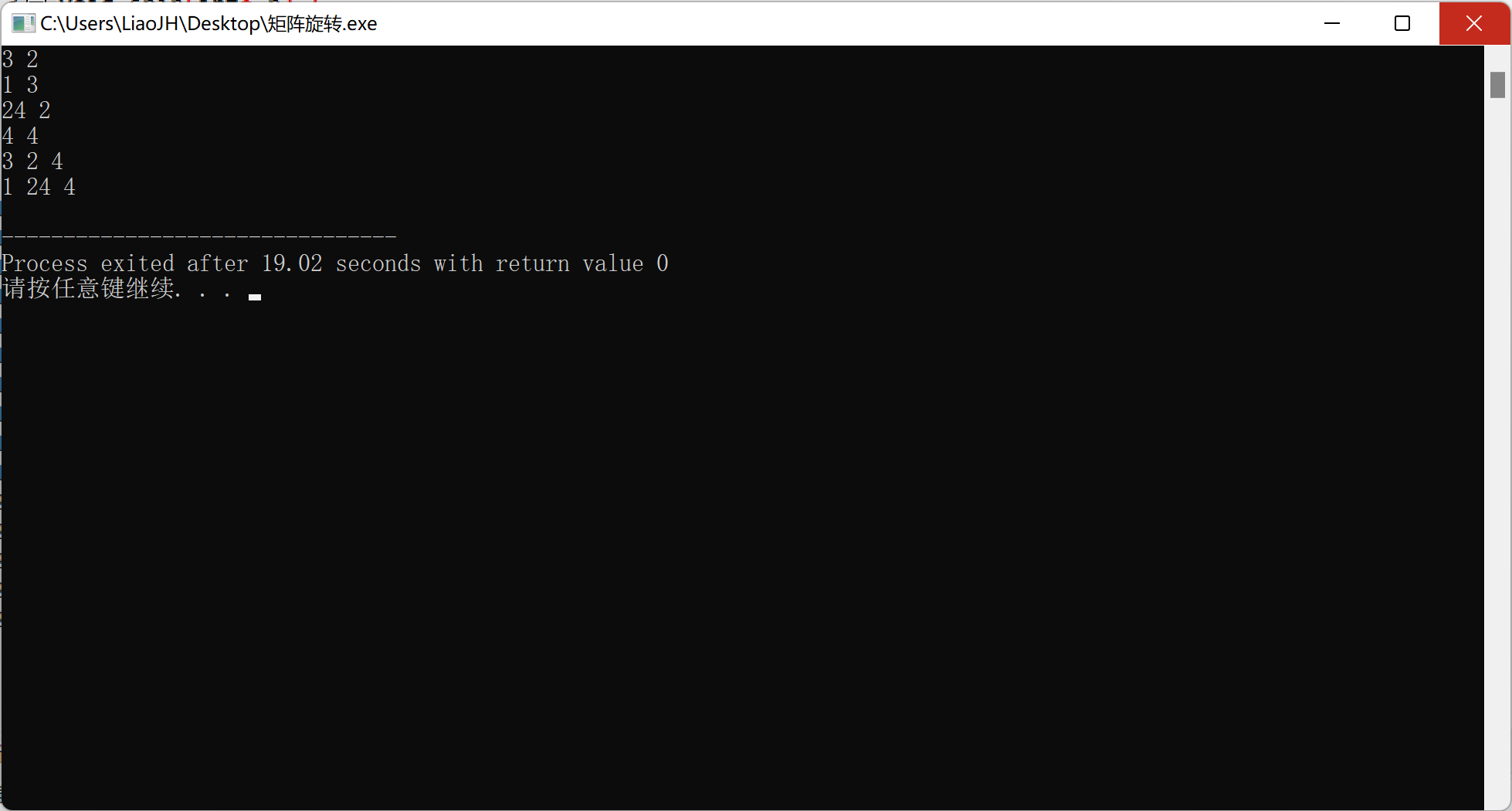
scanf("%d %d", &n, &m);

spin(a);

return 0;

}

3）运行结果：



（4）输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本，且n行文本的存储无冗余，删除每一行中的前置空格（' '）和水平制表符（'\t'）。要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。

1. 算法流程：

首先定义了一个名为remove\_spaces的函数，该函数接受一个字符指针参数str，用于删除该字符串中的前置空格和水平制表符。在main函数中，先读入文本的行数n，然后使用一个循环从标准输入中读取n行文本，并为每行文本分配内存空间。接下来，调用remove\_spaces函数删除每行文本中的前置空格和水平制表符，并输出删除前置空格后的文本。最后，释放每行文本的内存空间，以避免内存泄漏。

1. 源程序清单：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void remove\_spaces(char \*str) {

int i, j = 0;

for (i = 0; str[i]; i++) {

if (str[i] != ' ' && str[i] != '\t') {

str[j++] = str[i];

}

}

str[j] = '\0';

}

int main() {

int n, i;

char \*text[100];

char str[81];

printf("请输入文本行数n：");

scanf("%d", &n);

printf("请输入%d行文本：\n", n);

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

fgets(str, 81, stdin);

text[i] = (char\*)malloc(strlen(str) + 1);

strcpy(text[i], str);

remove\_spaces(text[i]);

}

printf("删除前置空格后的文本：\n");

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

printf("%s", text[i]);

free(text[i]);

}

return 0;

}

1. 运行结果：



（5）编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。

①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。

②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。

③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

例如，当输入13607122并回车，程序运行结果如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

1. 算法流程：

分别编写8个任务函数和调度函数和执行函数，在main函数中调用调度函数和执行函数进行任务调度和执行。

1. 源程序清单：

#include <stdio.h>

#include <string.h>

#define maxn 50

int a[maxn];

void task0()

{

    printf("task0 is called!\n");

}

void task1()

{

    printf("task1 is called!\n");

}

void task2()

{

    printf("task2 is called!\n");

}

void task3()

{

    printf("task3 is called!\n");

}

void task4()

{

    printf("task4 is called!\n");

}

void task5()

{

    printf("task5 is called!\n");

}

void task6()

{

    printf("task6 is called!\n");

}

void task7()

{

    printf("task7 is called!\n");

}

void(\*p[8])() = { task0,task1,task2,task3,task4,task5,task6,task7 };

void execute(void(\*\*p)(), int a[], int cnt)

{

    for (int i = 0; i < cnt; i++)

    {

        p[a[i]]();

    }

}

void scheduler()

{

    char s[50];

    scanf("%s", s);

    int len = strlen(s);

    for (int i = 0; i < len; i++)

    {

        a[i] = s[i] - '0';

    }

    execute(p, a, len);

}

int main()

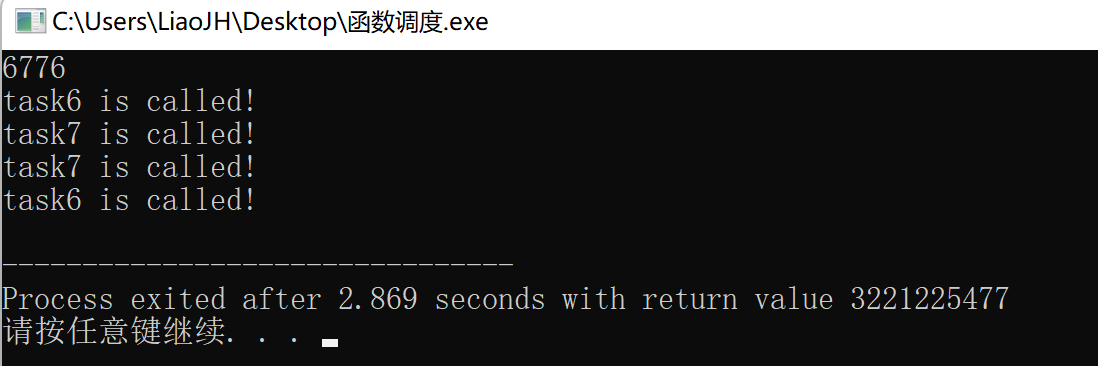
{

    scheduler();

    return 0;

}

3）运行结果：



**6.3 实验小结：**

指针是C语言中一个重要的概念，通过本次实验，我熟悉了指针的说明、赋值和使用，我们可以更灵活地操作变量和数组，提高程序的效率和可读性。以及指针函数的用法，能够编写更加灵活和高效的函数；同时，main函数是C语言程序的入口函数，通过本次实验我了解了带有参数的main函数的用法和处理命令行参数，实现更加灵活和可扩展的程序。



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.12.4**

**软件学院**

**目 录**

**[7 结构与联合 3](#_Toc404837920)**

[7.1 实验目的 3](#_Toc404837921)

[7.2 实验内容 3](#_Toc404837922)

[7.3 实验小结 1](#_Toc404837923)0

# 实验7 结构与联合实验

**7.1 实验目的**

1．通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。

2．通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。

3．了解字段结构和联合的用法。

**7.2 实验题目及要求**

**1．表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | A,A | A,A |
| 3 | \*p++->t,\*p->t | U,U | U,U |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

**2．源程序修改替换题**

给定一批整数，以0作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的指头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

1. 源程序中存在什么样的错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

void create\_list(struct s\_list \*headp,int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

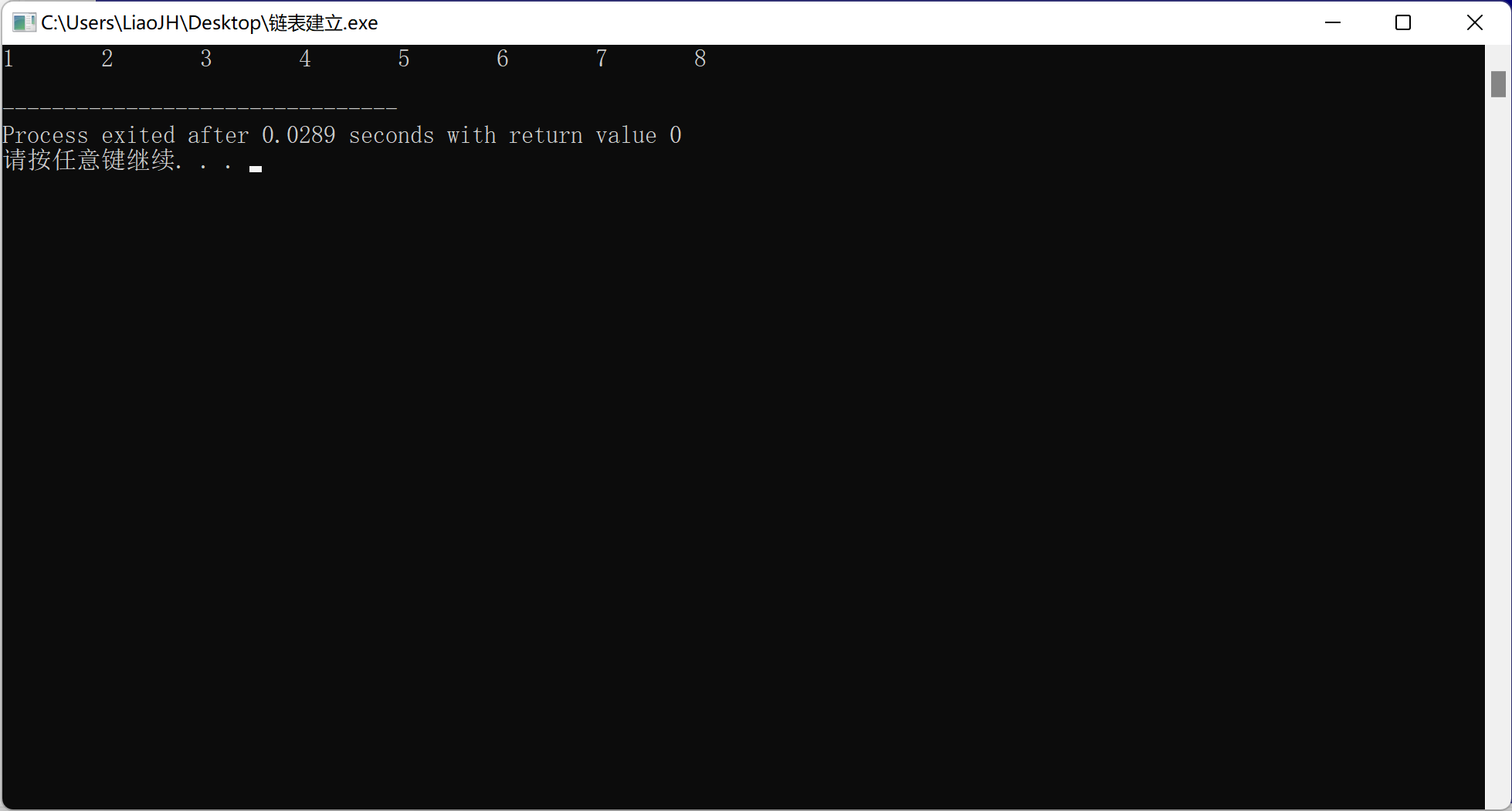
}

headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

解答：

1. 错误修改：
2. 首先我的编译器不支持void main，把它改成int main；
3. 函数参数传递错误，将void create\_list (struct s\_list \*headp,int \*p);改为void create\_list (struct s\_list \*\*headp,int \*p);（传递指针的指针）；
4. 在判断链表是否结束时只检查了数组的第一个元素是否为0，这是不正确的，应该检查当前元素是否为0，而不是数组的第一个元素。
5. 程序修改后运行结果：



1. 修改替换create\_list函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

create\_list函数修改如下：

void create\_list(struct s\_list \*\*headp, int \*p) {

struct s\_list \*loc\_head = NULL, \*tail = NULL; // loc\_head指向新创建的节点，tail始终指向最后一个创建的节点。

while (\*p) { // 遍历数组直到遇到结束标记0。

loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list)); // 为新节点分配内存。

loc\_head->data = \*p; // 设置新节点的数据。

loc\_head->next = tail; // 将新节点设置为最后一个节点的下一个节点，实现后进先出的效果。

tail = loc\_head; // 更新tail为新节点。

p++; // 处理下一个数组值。

}

if (tail) { // 如果链表不为空。

tail->next = NULL; // 确保最后一个节点的下一个节点设置为NULL。

}

\*headp = loc\_head; // 使头指针headp指向新创建的链表。

}

**3 程序设计**

以下（1）至（3）题对应Educoder 教学平台“C语言实验”课程，实验7，第17关实验7-1、第18关实验7-2，以及第19关实验7-3。

（1）本关任务：用单向链表建立一张班级成绩单，包括每个学生的学号、姓名、英语、高等数学、普通物理、C语言程序设计四门课程的成绩。用菜单实现下列功能：

① 输入每个学生的各项信息。

② 输出每个学生的各项信息。

③ 修改指定学生的指定数据项的内容。

④ 统计每个同学的平均成绩（保留2位小数）。

⑤ 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

 #include <stdio.h>

#include <string.h>

#include <malloc.h>

struct paper

{

    char num[15];

    char name[15];

    int eng, math, phy, clan;

    struct paper\* next;

};

struct paper\* head;

void Inputinfor(int n)

{

    head = (struct paper\*)malloc(sizeof(struct paper));

    struct paper\* ptr = head;

    for (int i = 0; i < n; i++)

    {

        scanf("%s%s%d%d%d%d", ptr->num, ptr->name, &ptr->eng, &ptr->math, &ptr->phy, &ptr->clan);

        if (i < n - 1)

        {

            ptr->next = (struct paper\*)malloc(sizeof(struct paper));

            ptr = ptr->next;

        }

        else

            ptr->next = NULL;

    }

    printf("完成了%d位同学的成绩输入。\n", n);

}

void Outputinfor()

{

    struct paper\* ptr = head;

    while (ptr != NULL)

    {

        printf("%s %s %d %d %d %d\n", ptr->num, ptr->name, ptr->eng, ptr->math, ptr->phy, ptr->clan);

        ptr = ptr->next;

    }

}

void Change()

{

    char arr[15];

    int n;

    scanf("%s%d", arr, &n);

    struct paper\* ptr = head;

    int input1;

    char input2[15];

    while (ptr != NULL)

    {

        if (strcmp(ptr->num, arr) == 0)

        {

            switch (n)

            {

            case 0:

                scanf("%s", input2);

                strcpy(ptr->name, input2);

                break;

            case 1:

                scanf("%d", &input1);

                ptr->eng = input1;

                break;

            case 2:

                scanf("%d", &input1);

                ptr->math = input1;

                break;

            case 3:

                scanf("%d", &input1);

                ptr->phy = input1;

                break;

            case 4:

                scanf("%d", &input1);

                ptr->clan = input1;

                break;

            }

            printf("%s %s %d %d %d %d\n", ptr->num, ptr->name, ptr->eng, ptr->math, ptr->phy, ptr->clan);

            break;

        }

        ptr = ptr->next;

    }

}

void Stagrade()

{

    struct paper\* ptr = head;

    double average;

    while (ptr != NULL)

    {

        average = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

        printf("%s %s %.2lf\n", ptr->num, ptr->name,average);

        ptr = ptr->next;

    }

}

void Outputgrade()

{

    struct paper\* ptr = head;

    int total;

    double average;

    while (ptr != NULL)

    {

        total = ptr->clan + ptr->eng + ptr->math + ptr->phy;

        average = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

        printf("%s %s %d %.2lf\n", ptr->num, ptr->name, total, average);

        ptr = ptr->next;

    }

}

void Sort()

{

    struct paper\* ptr = head;

    struct paper temp1, \*temp2, \*temp3;

    double ave1, ave2;

    for (int i = 0; i < 5; i++)

    {

        ptr = head;

        while (ptr->next->next != NULL)

        {

            ave1 = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

            ave2 = (ptr->next->clan + ptr->next->eng + ptr->next->math + ptr->next->phy) / 4.0;

            if (ave1 > ave2)

            {

                temp2 = ptr->next;

                temp3 = temp2->next;

                ptr->next = temp3;

                temp2->next = temp2;

                temp1 = \*ptr;

                \*ptr = \*temp2;

                \*temp2 = temp1;

            }

            ptr = ptr->next;

        }

    }

}

int main()

{

    int input, n;

    while(1)

    {

        scanf("%d", &input);

        switch (input)

        {

        case 1:

            scanf("%d", &n);

            Inputinfor(n);

            break;

        case 2:

            Outputinfor();

            break;

        case 3:

            Change();

            break;

        case 4:

            Stagrade();

            break;

        case 5:

            Outputgrade();

            break;

        case 6:

            Sort();

            Stagrade();

            break;

        case 0:

            return 0;

        }

    }

    return 0;

}

（2）本关任务：对程序设计题第（1）题的程序，⑥增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。

 #include <stdio.h>

#include <string.h>

#include <malloc.h>

struct paper

{

    char num[15];

    char name[15];

    int eng, math, phy, clan;

    struct paper\* next;

};

struct paper\* head;

void Inputinfor(int n)

{

    head = (struct paper\*)malloc(sizeof(struct paper));

    struct paper\* ptr = head;

    for (int i = 0; i < n; i++)

    {

        scanf("%s%s%d%d%d%d", ptr->num, ptr->name, &ptr->eng, &ptr->math, &ptr->phy, &ptr->clan);

        if (i < n - 1)

        {

            ptr->next = (struct paper\*)malloc(sizeof(struct paper));

            ptr = ptr->next;

        }

        else

            ptr->next = NULL;

    }

    printf("完成了%d位同学的成绩输入。\n", n);

}

void Outputinfor()

{

    struct paper\* ptr = head;

    while (ptr != NULL)

    {

        printf("%s %s %d %d %d %d\n", ptr->num, ptr->name, ptr->eng, ptr->math, ptr->phy, ptr->clan);

        ptr = ptr->next;

    }

}

void Change()

{

    char arr[15];

    int n;

    scanf("%s%d", arr, &n);

    struct paper\* ptr = head;

    int input1;

    char input2[15];

    while (ptr != NULL)

    {

        if (strcmp(ptr->num, arr) == 0)

        {

            switch (n)

            {

            case 0:

                scanf("%s", input2);

                strcpy(ptr->name, input2);

                break;

            case 1:

                scanf("%d", &input1);

                ptr->eng = input1;

                break;

            case 2:

                scanf("%d", &input1);

                ptr->math = input1;

                break;

            case 3:

                scanf("%d", &input1);

                ptr->phy = input1;

                break;

            case 4:

                scanf("%d", &input1);

                ptr->clan = input1;

                break;

            }

            printf("%s %s %d %d %d %d\n", ptr->num, ptr->name, ptr->eng, ptr->math, ptr->phy, ptr->clan);

            break;

        }

        ptr = ptr->next;

    }

}

void Stagrade()

{

    struct paper\* ptr = head;

    double average;

    while (ptr != NULL)

    {

        average = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

        printf("%s %s %.2lf\n", ptr->num, ptr->name,average);

        ptr = ptr->next;

    }

}

void Outputgrade()

{

    struct paper\* ptr = head;

    int total;

    double average;

    while (ptr != NULL)

    {

        total = ptr->clan + ptr->eng + ptr->math + ptr->phy;

        average = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

        printf("%s %s %d %.2lf\n", ptr->num, ptr->name, total, average);

        ptr = ptr->next;

    }

}

void Sort()

{

    struct paper\* ptr = head;

    struct paper temp1, \*temp2, \*temp3;

    double ave1, ave2;

    for (int i = 0; i < 5; i++)

    {

        ptr = head;

        while (ptr->next->next != NULL)

        {

            ave1 = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

            ave2 = (ptr->next->clan + ptr->next->eng + ptr->next->math + ptr->next->phy) / 4.0;

            if (ave1 > ave2)

            {

                temp2 = ptr->next;

                temp3 = temp2->next;

                ptr->next = temp3;

                temp2->next = temp2;

                temp1 = \*ptr;

                \*ptr = \*temp2;

                \*temp2 = temp1;

            }

            ptr = ptr->next;

        }

        ave1 = (ptr->clan + ptr->eng + ptr->math + ptr->phy) / 4.0;

        ave2 = (ptr->next->clan + ptr->next->eng + ptr->next->math + ptr->next->phy) / 4.0;

        if (ave1 > ave2)

        {

            temp2 = ptr->next;

            temp2->next = temp2;

            ptr->next = NULL;

            temp1 = \*ptr;

            \*ptr = \*temp2;

            \*temp2 = temp1;

        }

    }

}

int main()

{

    int input, n;

    while(1)

    {

        scanf("%d", &input);

        switch (input)

        {

        case 1:

            scanf("%d", &n);

            Inputinfor(n);

            break;

        case 2:

            Outputinfor();

            break;

        case 3:

            Change();

            break;

        case 4:

            Stagrade();

            break;

        case 5:

            Outputgrade();

            break;

        case 6:

            Sort();

            Stagrade();

            break;

        case 0:

            return 0;

        }

    }

    return 0;

}

7.3实验小结：

本次实验深入探讨了结构的多种应用方式，包括结构的说明和引用、结构的指针、结构数组，以及在函数中的使用。同时掌握了动态储存分配函数的实际操作，对自引用结构有了更深入的理解。此外，单向链表的创建、遍历、结点的增删和查找等操作也得到了实践。



**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程202301班**

**学 号： U202317252**

**姓 名： 廖家欢**

**指导教师： 万琳**

**报告日期： 2023.12.15**

**软件学院**

**目 录**

**[8 文件实验 3](#_Toc404837920)**

[8.1 实验目的 3](#_Toc404837921)

[8.2 实验内容 3](#_Toc404837922)

[8.3 实验小结 1](#_Toc404837923)0

实验8 文件操作实验

8.1实验目的

（1）熟悉文本文件和二进制文件在磁盘中的存储方式；

（2）熟练掌握流式文件的读写方法。

8.2实验题目及要求

1．文件类型的程序验证题

设有程序：

#include <stdio.h>

int main(void)

{

short a=0x253f,b=0x7b7d;

char ch;

FILE \*fp1,\*fp2;

fp1=fopen("d:\\abc1.bin","wb+");

fp2=fopen("d:\\abc2.txt","w+");

fwrite(&a,sizeof(short),1,fp1);

fwrite(&b,sizeof(short),1,fp1);

fprintf(fp2,"%hx %hx",a,b);

rewind(fp1); rewind(fp2);

while((ch = fgetc(fp1)) != EOF)

putchar(ch);

putchar('\n');

while((ch = fgetc(fp2)) != EOF)

putchar(ch);

putchar('\n');

fclose(fp1);

fclose(fp2);

return 0;

}

1. 请思考程序的输出结果，然后通过上机运行来加以验证。
2. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？

（3）将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

**2．源程序修改替换题**

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

type filename

1. 源程序中存在什么样的逻辑错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

#include<stdio.h>

#include<stdlib.h>

int main(int argc, char\* argv[])

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/

printf("Can't open %s file!\n",argv[1]);

exit(-1);

}

while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

1. 用输入输出重定向freopen改写main函数。

**3．程序设计**

（1）编写一个程序，用给定的字符串替换文件中的目标字符串，并显示输出替换的个数。

注意：读取的文件路径请使用experiment/src/step8/source.txt

若文件为

`There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!`

样例输入：`you they`

样例输出：

`3`

`There are moments in life when they miss someone so much that they just want to pick them from theyr dreams and hug them for real!`

1）源程序清单：

 #include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

    char s1[20], s2[20], ch, result[1000] = { 0 }, \* p, ori[1000], temp[100];

    int i = 0, count = 0;

    scanf("%s%s", s1, s2);

    FILE\* fp = fopen("experiment/src/step8/source.txt", "r");

    while ((ch = fgetc(fp)) != EOF)

    {

        ori[i] = ch;

        i++;

    }

    fclose(fp);

    ori[i] = '\0';

    while(1)

    {

        if ((p = strstr(ori, s1)) != NULL)

        {

            strncpy(temp, ori, p - ori);

            temp[p - ori] = '\0';

            strcat(result, temp);

            strcat(result, s2);

            strcat(result, p + strlen(s1));

            int j = 0;

            for (j = 0; result[j]; j++)

                ori[j] = result[j];

            ori[j] = '\0';

            memset(result, 0, sizeof(result));

            count++;

        }

        else

            break;

    }

    printf("%d\n", count);

    printf("%s", ori);

    return 0;

}

**实验小结：**

通过实验，我深入了解了文本文件和二进制文件在磁盘中的存储方式，掌握了流式文件的读写方法。这为后续数据处理和文件操作奠定了坚实基础。