

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 网安2003班**

**学 号： U202012068**

**姓 名： 叶礼亮**

**指导教师： 余林琛**

**报告日期： 2020/10/17**

**网络空间安全学院**

**目 录**

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

[1.1□□实验目的 1](#_Toc404837921)

[1.2□□实验内容 1](#_Toc404837922)

[1.3□□实验小结 1](#_Toc404837923)

[**2□□□流程控制实验 2**](#_Toc404837924)

[2.1□□实验目的 1](#_Toc404837921)

[2.2□□实验内容 1](#_Toc404837922)

[2.3□□实验小结 1](#_Toc404837923)

[**3□□□函数与程序结构实验 3**](#_Toc404837929)

[3.1□□实验目的 1](#_Toc404837921)

[3.2□□实验内容 1](#_Toc404837922)

[3.3□□实验小结 1](#_Toc404837923)

[**4□□□编译预处理实验 4**](#_Toc404837934)

[4.1□□实验目的 1](#_Toc404837921)

[4.2□□实验内容 1](#_Toc404837922)

[4.3□□实验小结 1](#_Toc404837923)

[**5□□□数组实验 5**](#_Toc404837938)

[5.1□□实验目的 1](#_Toc404837921)

[5.2□□实验内容 1](#_Toc404837922)

[5.3□□实验小结 1](#_Toc404837923)

[**6□□□指针实验 6**](#_Toc404837943)

[6.1□□实验目的 1](#_Toc404837921)

[6.2□□实验内容 1](#_Toc404837922)

[6.3□□实验小结 1](#_Toc404837923)

[**7□□□结构与联合实验 7**](#_Toc404837948)

[7.1□□实验目的 1](#_Toc404837921)

[7.2□□实验内容 1](#_Toc404837922)

[7.3□□实验小结 1](#_Toc404837923)

[**8□□□文件实验 8**](#_Toc404837953)

[8.1□□实验目的 1](#_Toc404837921)

[8.2□□实验内容 1](#_Toc404837922)

[8.3□□实验小结 1](#_Toc404837923)

[**参考文献 9**](#_Toc404837957)

# 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

**1.2.1 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

2) 第10行的scanf变量前要加&，正确形式为：scanf("%d", &f );

3) 第11行的5/9会得0,应该用double类型，正确形式为：

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

4) 第15行的scanf,double类型应该用%lf格式化，正确形式为：

scanf("%lf", &r);

5) Task3的newint变量没有声明，正确形式为：int newint;

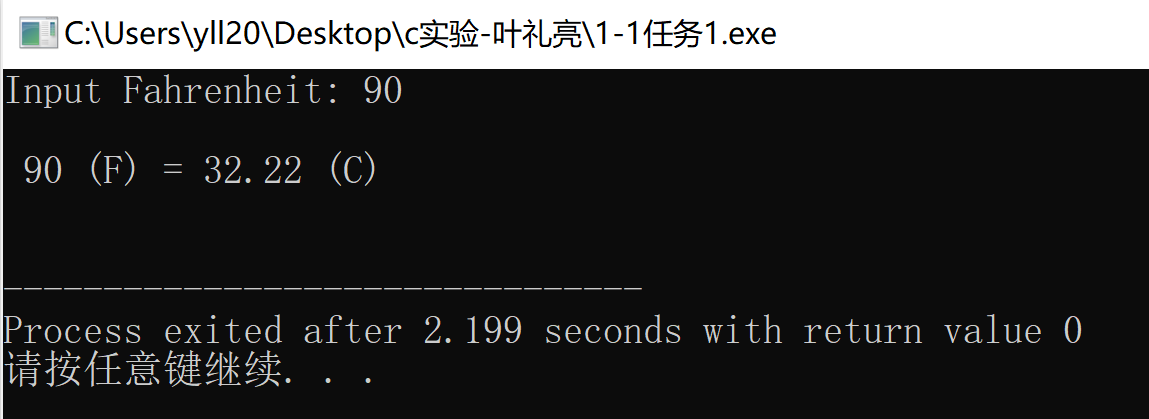
6) 第21行的h应该&0x00ff，正确形式为：

newint = (p&0xff00)|((k>>8)&0x00ff);

7) 第22行的int类型应该用%hx格式化，正确形式为：

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

（2）错误修改后运行结果：



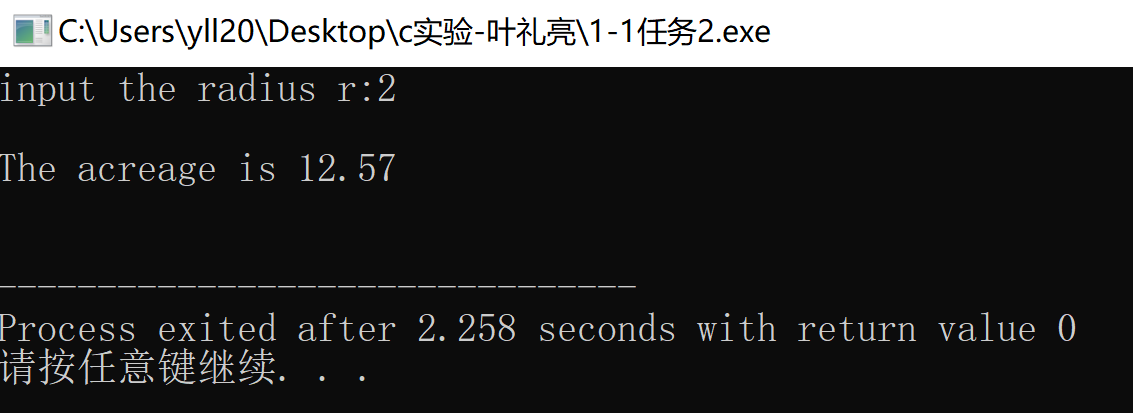
****

图1-1 源程序改错题1的运行结果

**1.2.2 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：

/\* 实验1-2程序分析与修改题源程序 \*/

#include <stdio.h>

int main( )

{

int m, n, k, p, i, d;

printf("input m, n \n");

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

if (m<n) /\* 交换m和n \*/

{

m = m^n;

n = m^n;

m = m^n;//交换m,n

}

k = 0;

while ((m&1)==0 && (n&1)==0) /\* m和n均为偶数 \*/

{

m = m>>1; /\* 用2约简m和n \*/

n = n>>1;

k++;

}

p = 1;

p = p<<k; /\* 求p=2^k \*/

while((d=m-n)!=n)

{

if(d>n) m = d;

else

{

m-=(n=d);

}

}

d \*=p;

printf("the greatest common divisor : %d", d);

return 0;

}

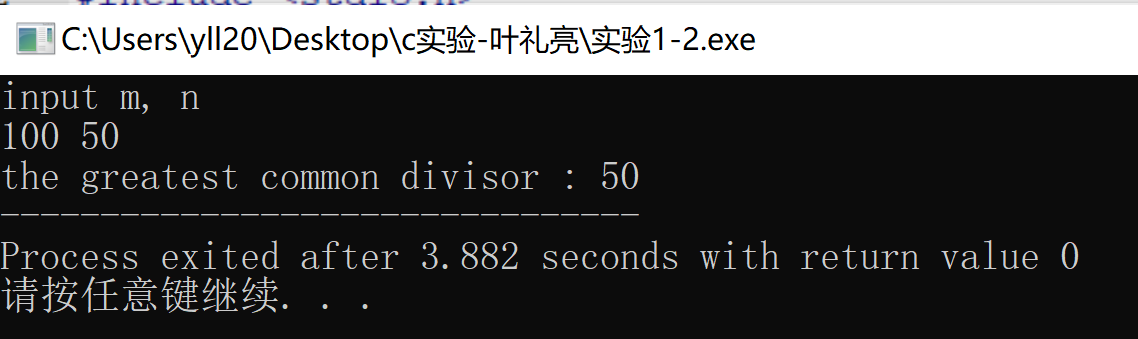


图1-2 源程序修改替换题的运行结果

**1.2.3 程序设计**

（1）编写一个程序，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，最后输出ｃ。

**解答：**

1） 算法流程如图1-3所示。

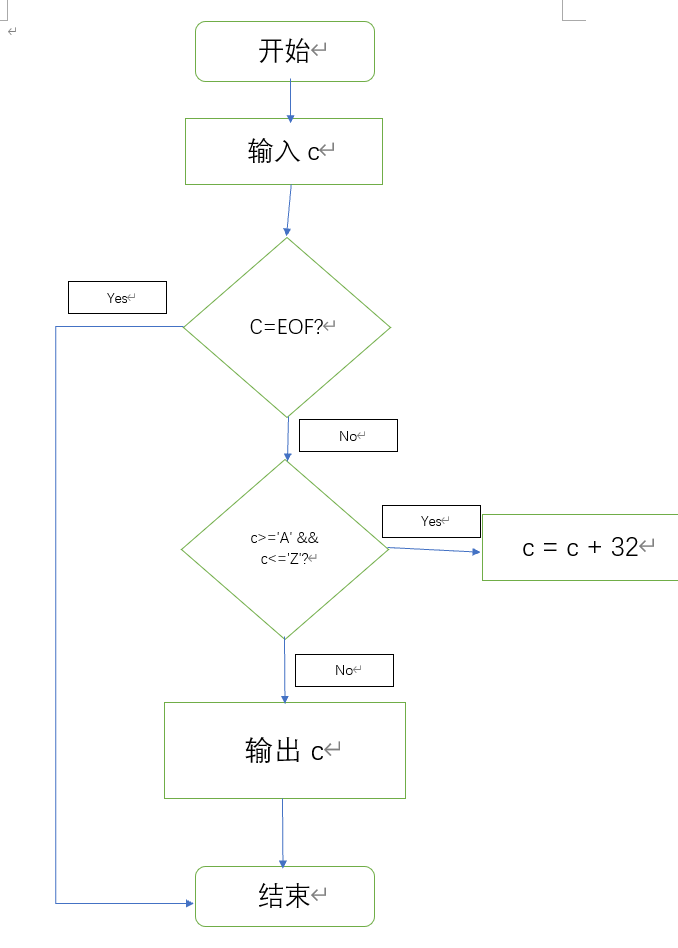


图1-3 程序设计题1的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

char c;

while((c = getchar())!=EOF)

{

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

{

c = c + 32;

}

putchar(c);

}

return 0;

}

3）测试

（a） 测试数据：

1.A,2.c

（b） 对应测试数据的运行结果截图

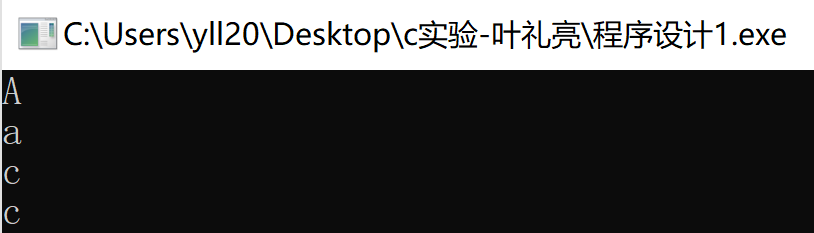


图1-4 程序设计题1的运行结果图

（2）编写一个程序，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。

**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×××××位，即： ×××××

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 x, m, n;

scanf("%x %hu %hu",&x,&m,&n);

if (!(((m>=0)&&(m<=15))&&((n>=1)&&(n<=16-m))))

//如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-m继续,否则结束

{

printf("参数范围错误");

return 0;

}

x = x>>m;

x = x<<(16-n);

printf("%x",x);

return 0;

}

3）测试

（a） 测试数据：

如表1-1所示。

表1-1 编程题3的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 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 |
| 用例2 | 1101 0101 1000 0011（D583） | 16 | 1 | 输入错误（m值超范围） |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） |

（b） 对应测试测试用例的运行结果如图1-5所示。

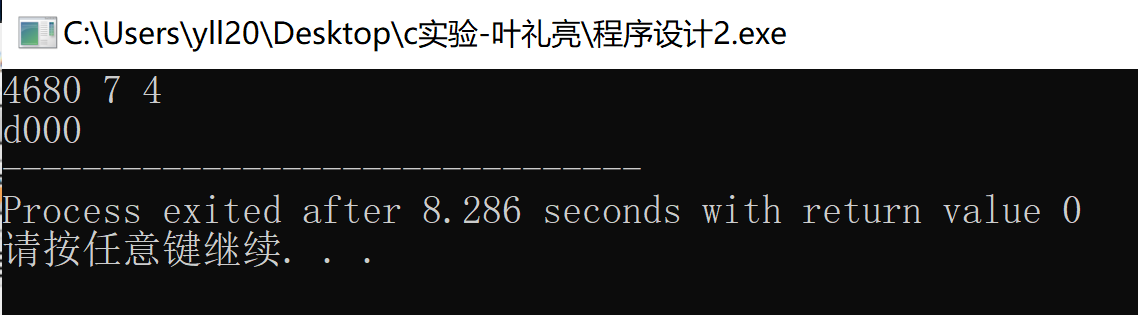


图1-5 编程题3的测试用例一的运行结果

对应测试测试用例2的运行结果如图1-6所示。

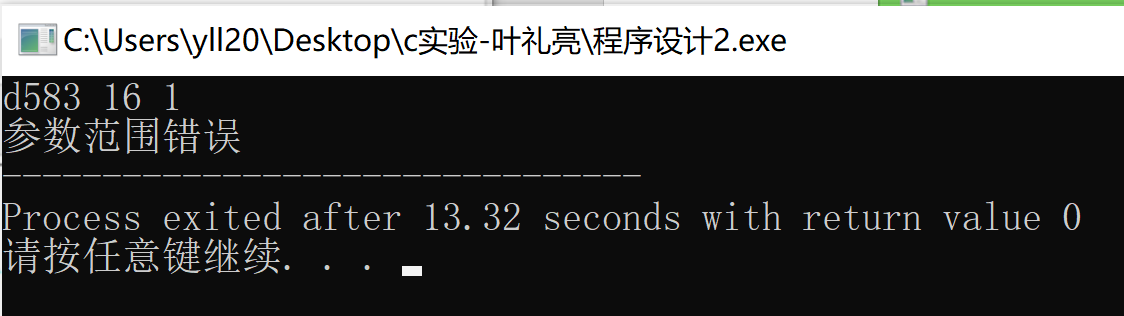


图1-6 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图1-7所示。

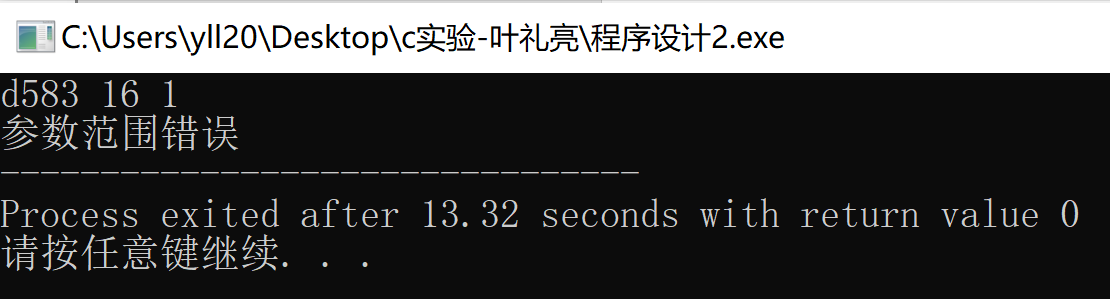


图1-7 编程题3的测试用例三的运行结果

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

实验过程中遇到了一些问题，比如很多程序都多多少少地写出了一些bug,但是通过一步一步调试解决了。通过这次实验掌握了基本的调试方法。同时学会了不使用中间变量进行交换的办法。通过对改错题的修改也帮助提醒自己不要犯类似的错误。

# 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

**2.2.1 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) {

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);

14 }

15 return 0;

16 }

**解答：**

（1）错误修改：

1) 第8行的判断内运算优先级有错，正确形式为：

if (!(x%i)) {

2) 第12行的判断相等应该用==，正确形式为：

if(flag==1) printf("%d是合数", x);

3) flag每次循环都应该复位，正确形式为：

添加flag = 0;

（2）错误修改后运行结果：

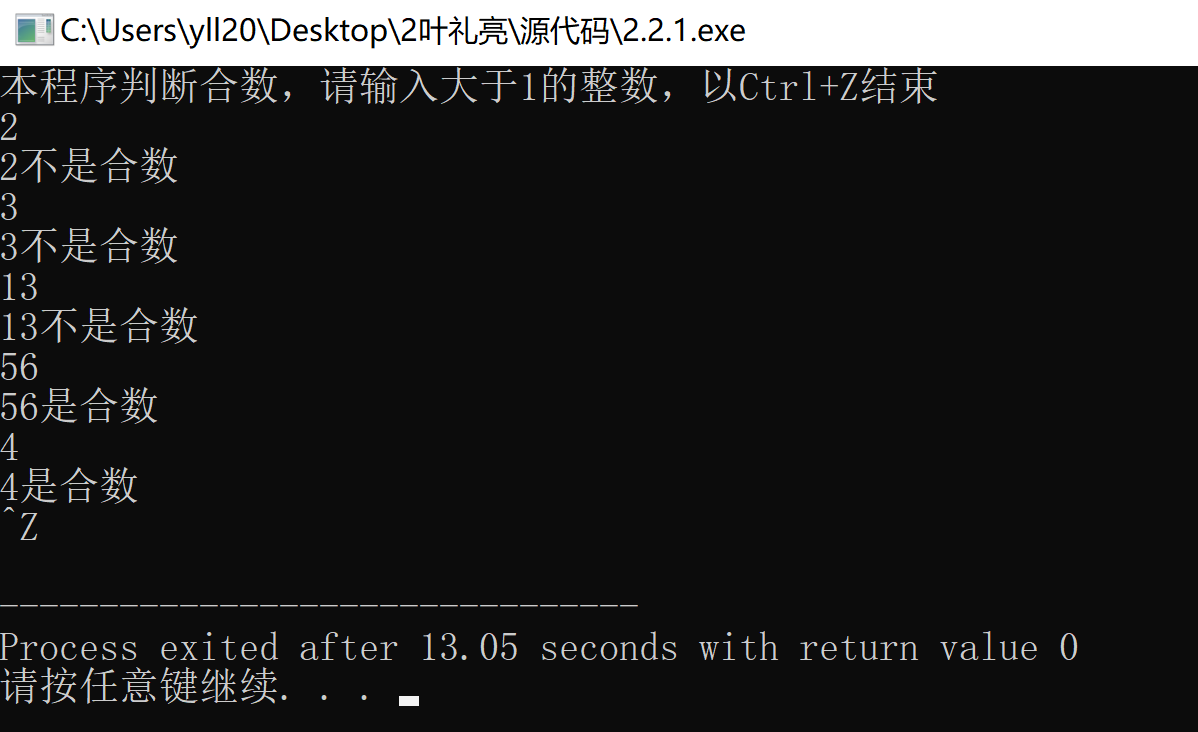


图2-1 程序改错题1的运行结果

**2.2.2 源程序修改替换**

（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。

**解答：**

修改后的程序如下所示：

/\* 实验2-1改错题程序：合数判断器\*/

#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&&flag==0;i++)//加了一句&&flag==0

{

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;

}



图2-2 源程序修改替换题1的运行结果

（2）修改实验2-1程序，将for循环改用do-while循环。

修改后的程序如下所示:

/\* 实验2-1改错题程序：合数判断器\*/

#include <stdio.h>

#include<iostream>

using namespace std;

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)

{

i=2,k=x>>1;

do

{

if (i>k) break;

if (!(x%i))

{

flag = 1;

}

i++;

}while(i<=k&&flag==0);

if(flag==1)

{

printf("%d是合数", x);

}

else

{

printf("%d不是合数", x);

}

flag = 0;

}

return 0;

}



图2-3 程序改错题2的运行结果

（3）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。

修改后的程序如下所示:

#include <stdio.h>

int unit\_num(int n)

{

int i;

for(i=2;i<=n>>1;i++)

{

if(n%i==0)

return 1;

}

return 0;

}

int main()

{

int i,a,b;

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

{

a=i/100;

b=i/10;

if(unit\_num(i)&&unit\_num(a)&&unit\_num(b)&&a>1)

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

}

return 0;

}

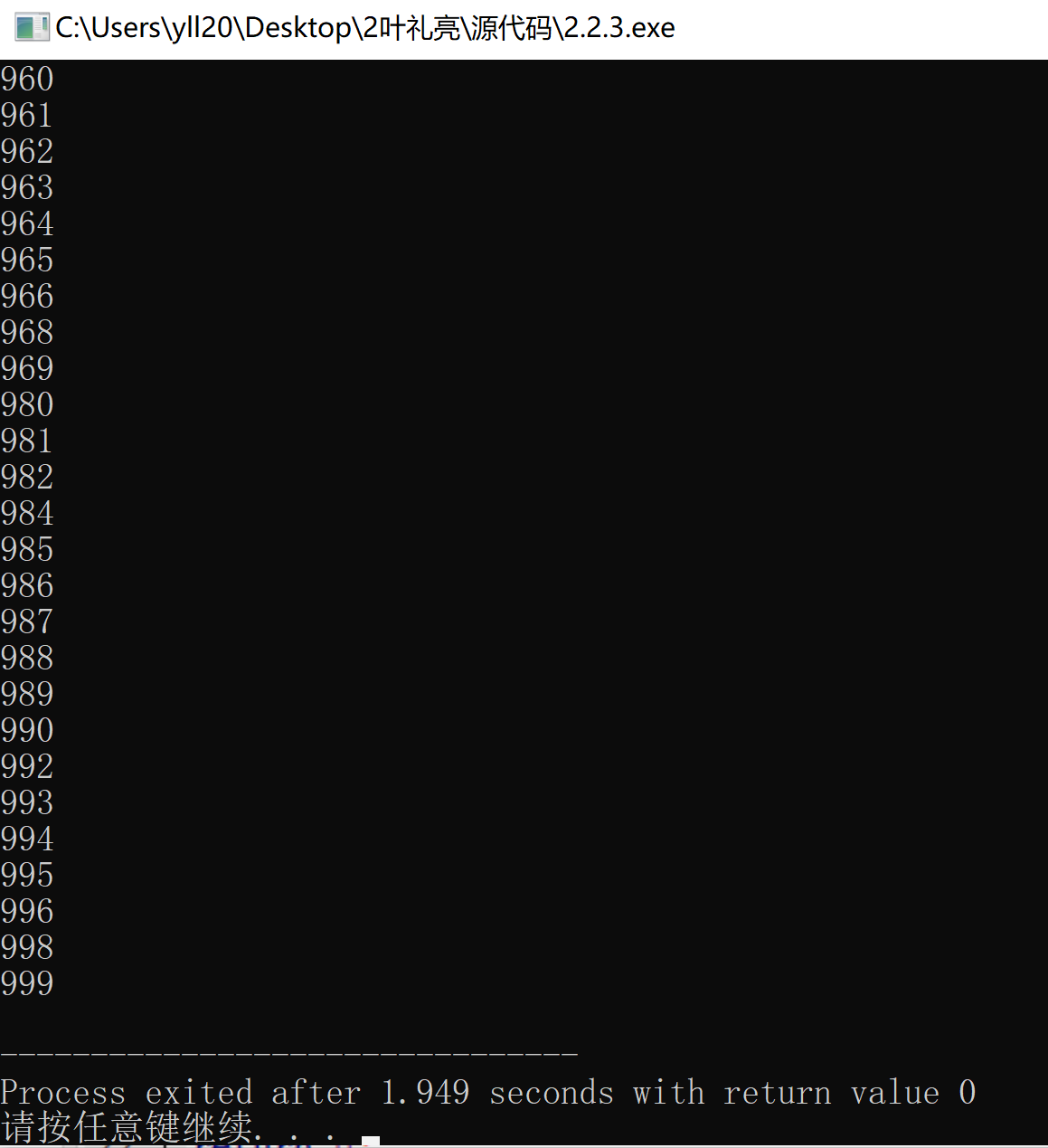


图2-4 程序改错题3的运行结果

**2.2.3 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；x＞5000，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。

**解答：**

1. 算法流程如图2-5所示。

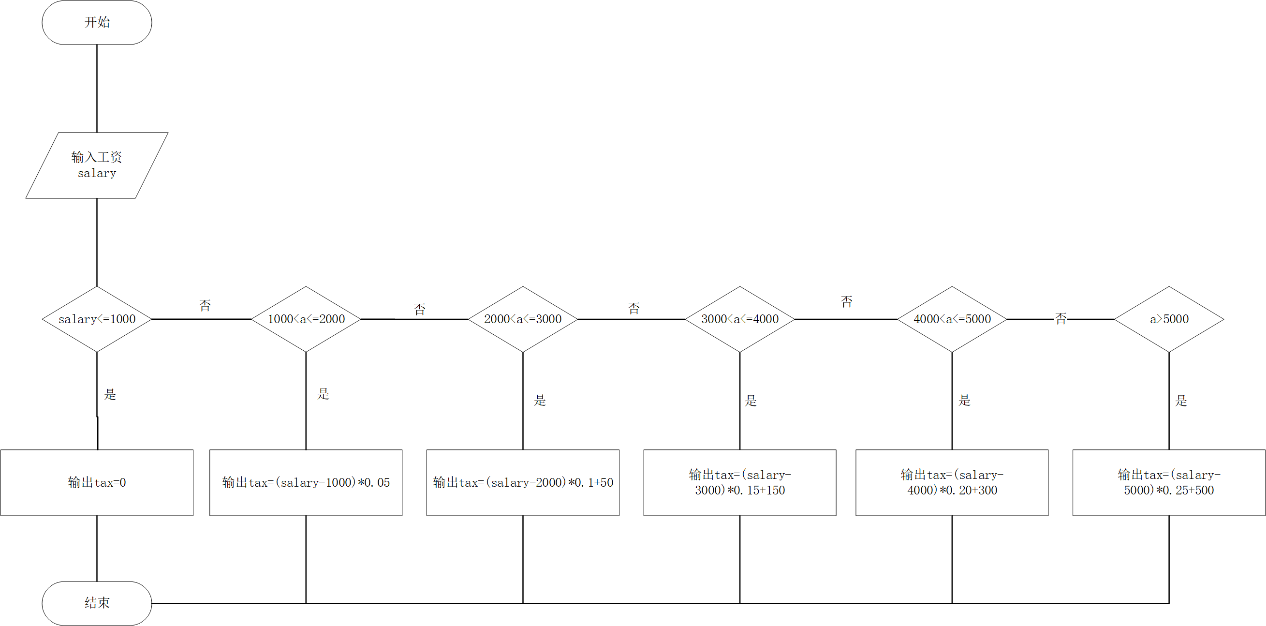


图2-5 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

int main()

{

int salary, tax;

int a;

scanf("%d",&salary);

a=salary/1000;

if(a==0) tax=0;

else if(a==1) tax=(salary-1000)\*0.05;

else if(a==2) tax=(salary-2000)\*0.1+50;

else if(a==3) tax=(salary-3000)\*0.15+150;

else if(a==4) tax=(salary-4000)\*0.20+300;

else tax=(salary-5000)\*0.25+500;

printf("%d",tax);

return 0;

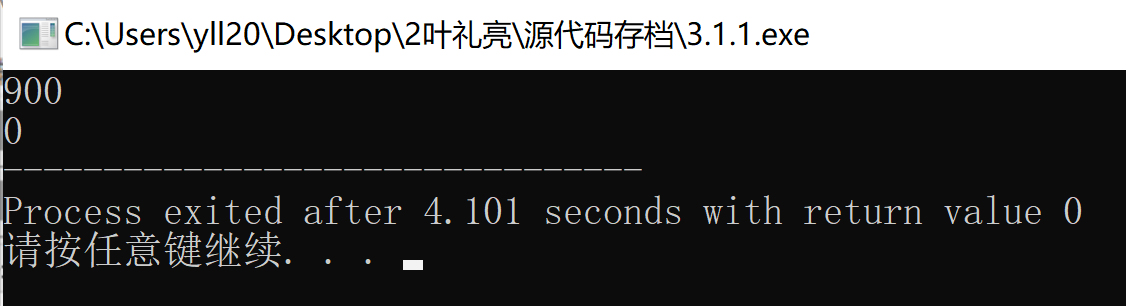
}

3）测试

（a） 测试数据：

900,4500

（b） 对应测试数据的运行结果截图



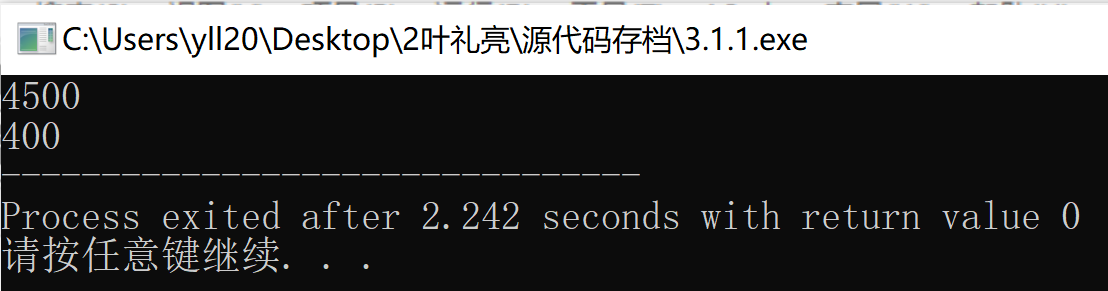


图2-6 程序设计题1的运行结果

（2）将输入的正文复制到输出，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） 算法流程如图1.1所示。

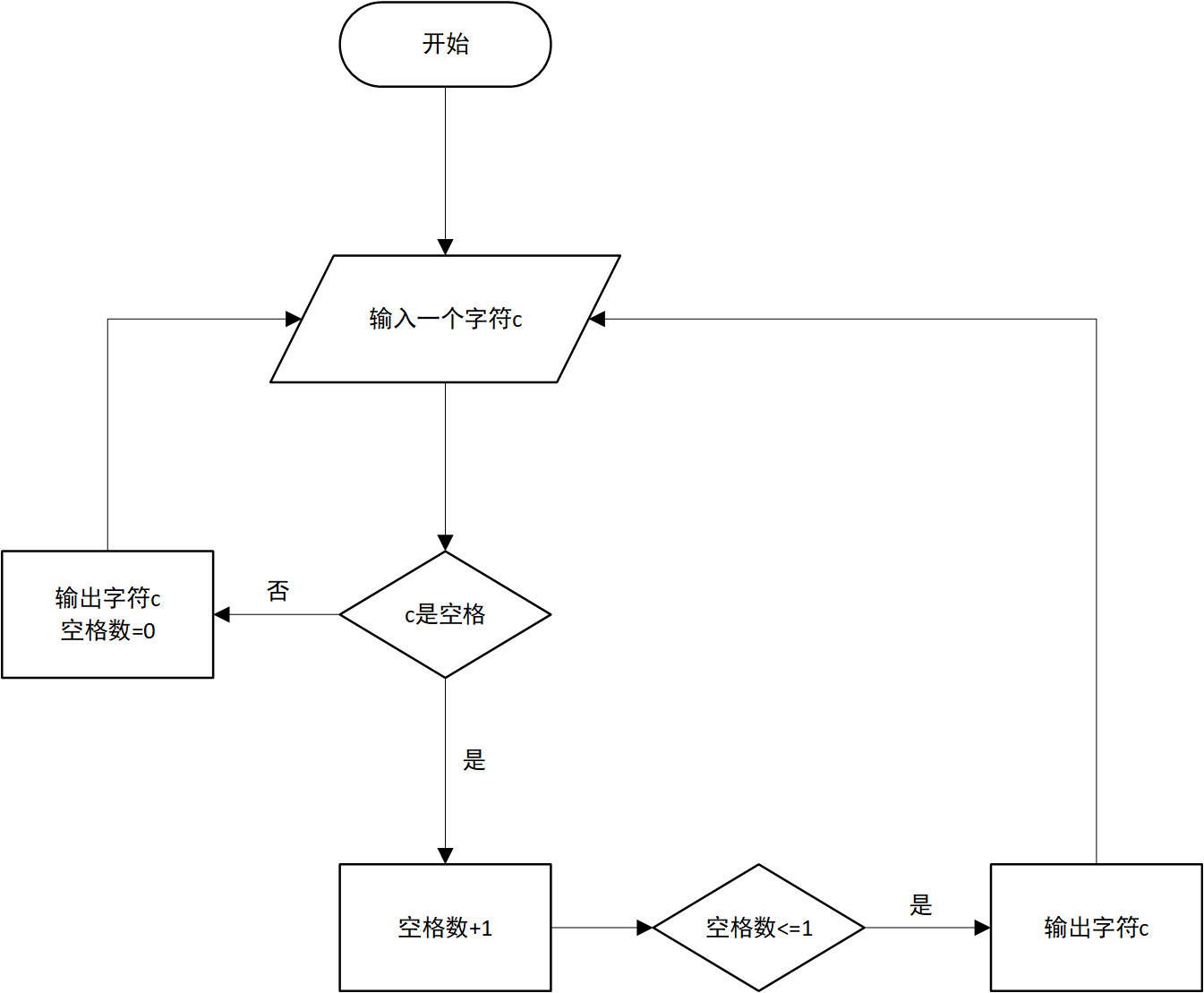


图2-7 程序设计题2的程序流程图

2）源程序清单

#include <stdio.h>

int main()

{

int c, space\_num;

space\_num=0;

while((c=getchar())!=EOF)

{

if (c!=' ')

{

putchar(c);

space\_num=0;

}

else

{

space\_num++;

if (space\_num<=1) putchar(c);

}

}

return 0;

}

3）测试

（a） 测试数据：

1.a b c12 2. Sc d6

（b） 对应测试数据的运行结果截图

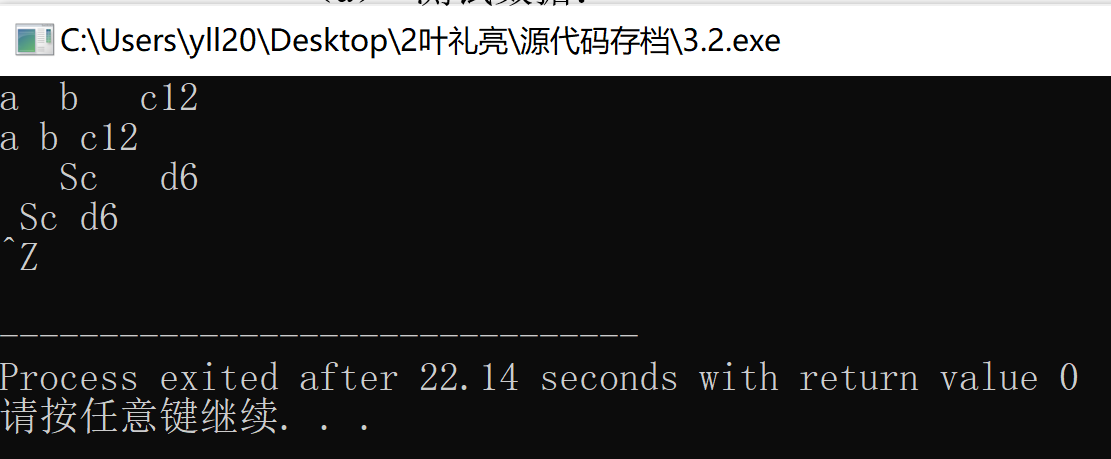


图2-8 程序设计题2的运行结果

（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。

1. 算法流程如下所示。
2. 输入需要输出的行数
3. 每一层开头输出行数减去几个数字+1
4. 调用获取值的函数,四格向左对齐输出
5. 源程序清单

#include<stdio.h>

int number(int i, int j);

int main(void)

{

int i, j, m;

int howmany\_lines;

scanf("%d",&howmany\_lines);

for(i = 0; i < howmany\_lines; i++)

{

for(m = howmany\_lines-i-1; m > 0; m--)

printf(" ");

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

printf("%-4d", number(i, j));

printf("\n");

}

return 0;

}

int number(int i, int j)

{

int sum;

if (j==0)

sum = 1;

else

sum = number(i, j-1)\*(i - j + 1)/j;

return sum;

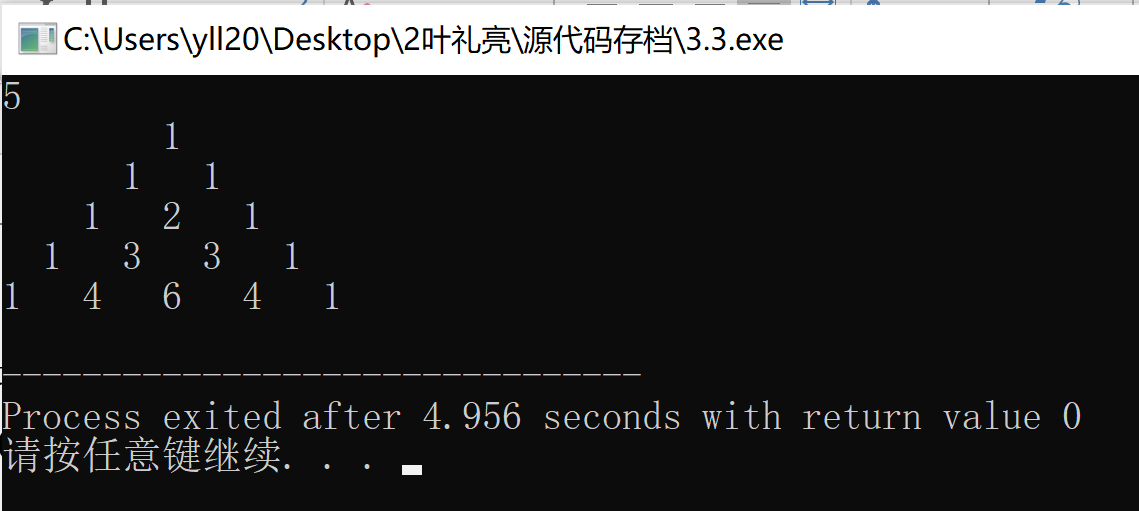
}

3）测试

（a） 测试数据：

5,10

（b） 对应测试数据的运行结果截图



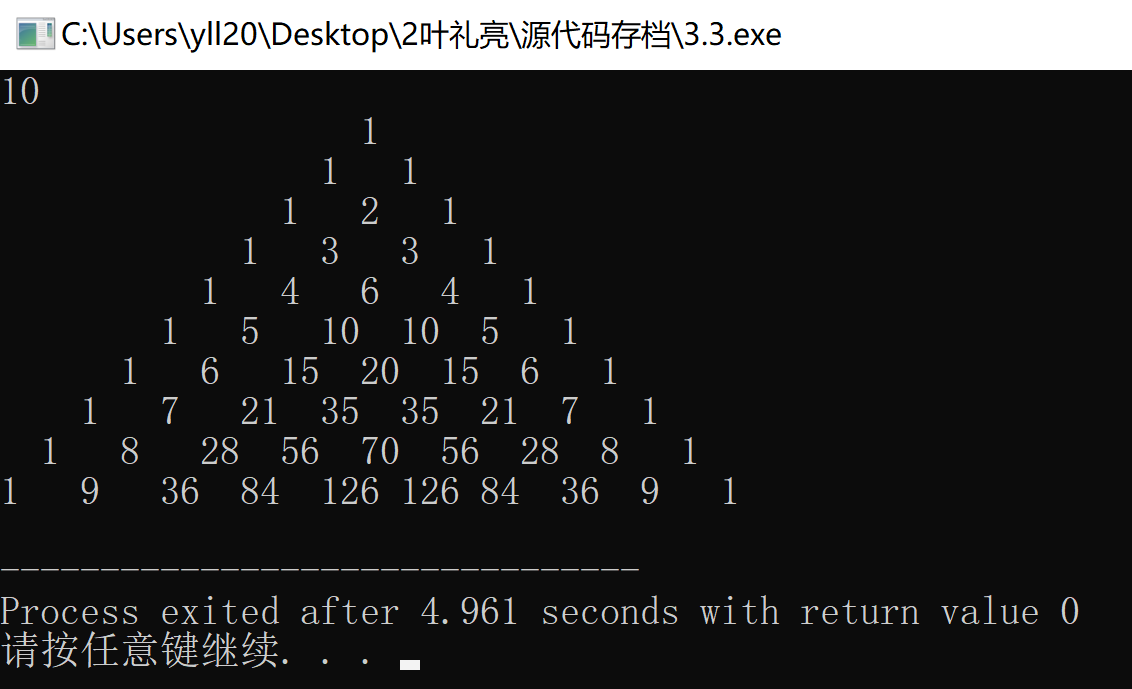


图2-9 程序设计题3的运行结果

（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。

1） 算法流程如图4.1所示。

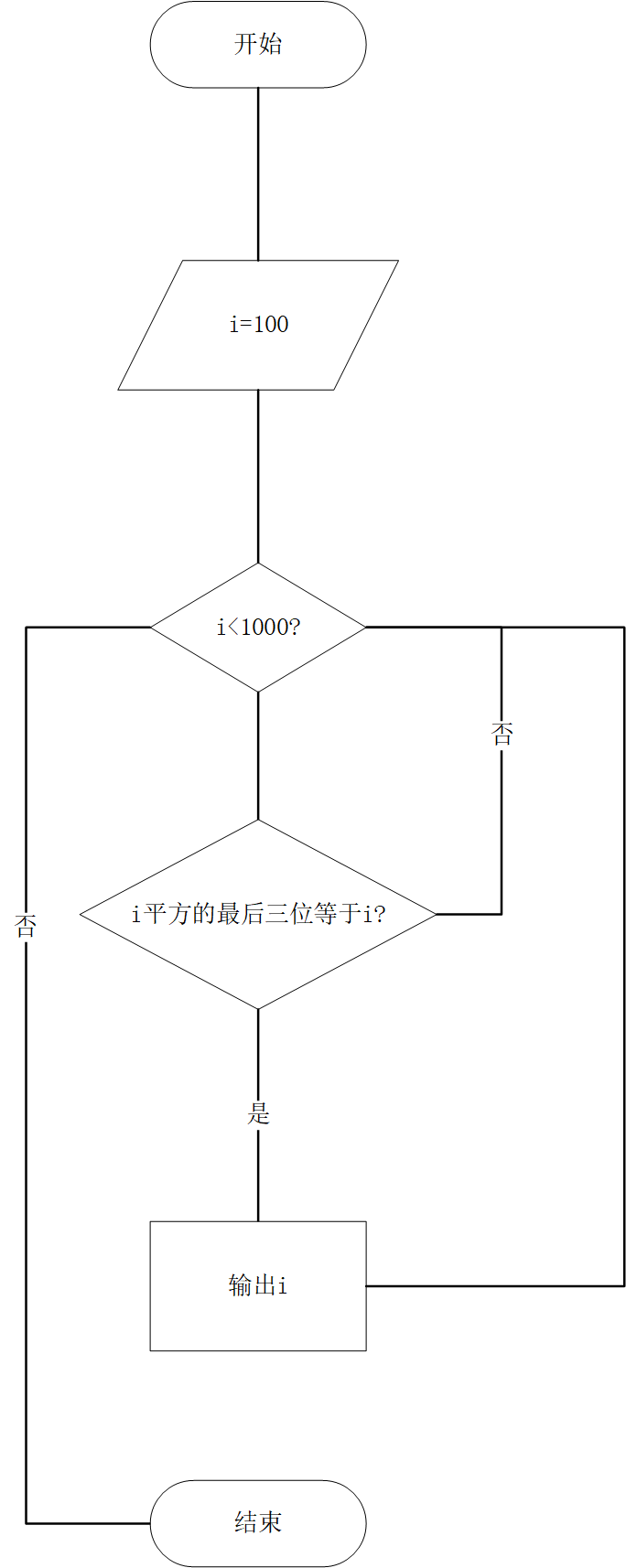


图2-10 程序设计题4的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

int i, i\_2, last\_3;

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

{

i\_2=i\*i;

last\_3=i\_2%1000;

if(last\_3==i)

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

}

return 0;

}

3）测试

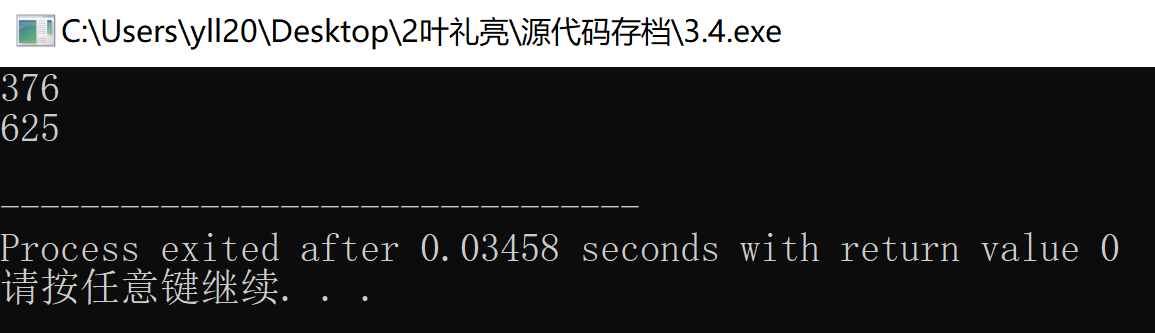


图2-11 程序设计题4的运行结果

## 2.3 实验小结

实验中很多地方的算法有累赘或者效率低的地方,修修补补才弄好。

这提醒了我今后写代码要提前做好规划，不要边想边写，这样会产生很多不必要的空间和时间浪费。

# 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

**3.2.1 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：

1) 第6行的函数使用前应该先声明，正确形式为在开头添加：

long long sum\_fac(int n);

2) 第9行的函数数字过大应该用long long，正确形式为：

long long sum\_fac(int n)

3) 第14行的fac变量应该先初始化为1，正确形式为在循环前添加：

fac = 1;

4) 第15行的s+=fac;应该放在循环内部，正确形式为在循环内添加：

s+=fac;

（2）错误修改后运行结果：

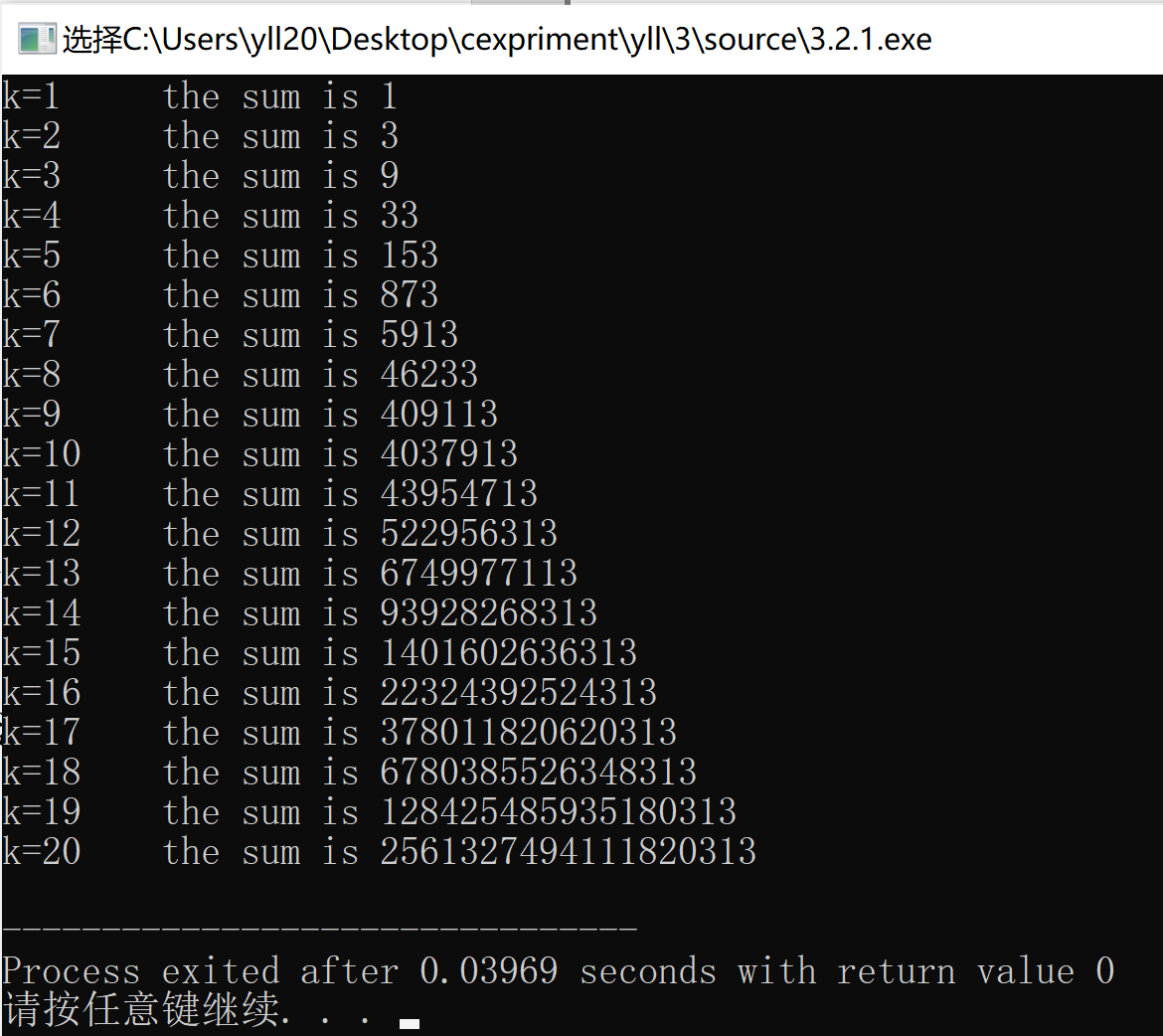


图3-1 改错题1的运行结果

**3.2.2 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。

**解答：**

替换后的程序如下所示：

#include <stdio.h>

long long sum\_fac(int);

int main()

{

int k;

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

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

return 0;

}

long long sum\_fac(int n)

{

long long s=0;

long long fac;

int i;

fac = 1;

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

{

fac \*=i;

}

if(n == 1) return 1;

else s =sum\_fac(n-1) + fac;

return s;

}

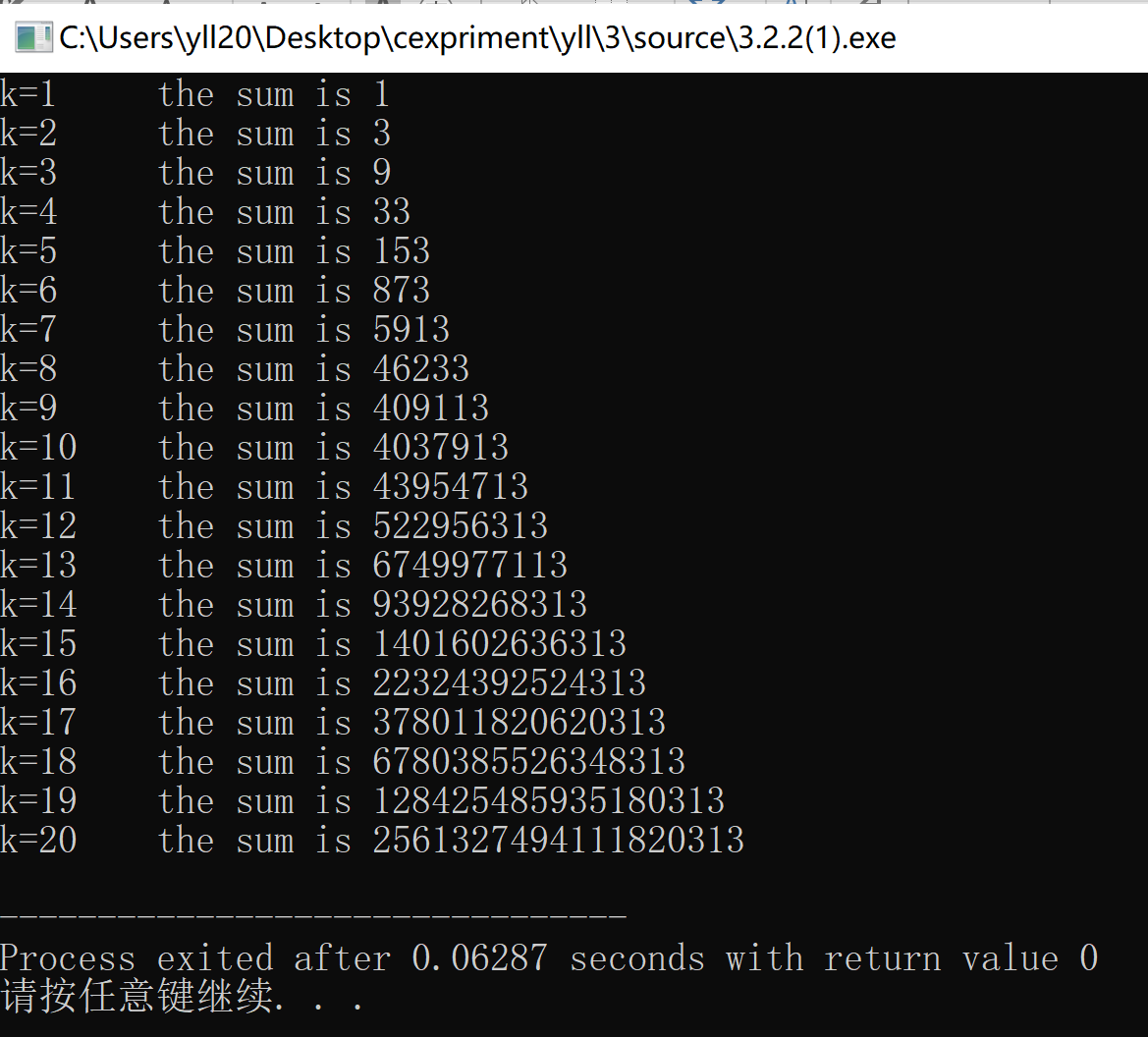


图3-2 程序替换题1的运行结果

（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;

}

**解答：**

利用上次的计算结果来简化计算,替换后的程序如下所示：

#include<stdio.h>

double sum(double x,int n)

{

int i;

double z=1.0;

double last\_mulx=1;

long long last\_fac=1;

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

{

z=z+(last\_mulx\*x)/(last\_fac\*i);

last\_mulx\*=x;

last\_fac\*=i; //利用上次的值

}

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;

}

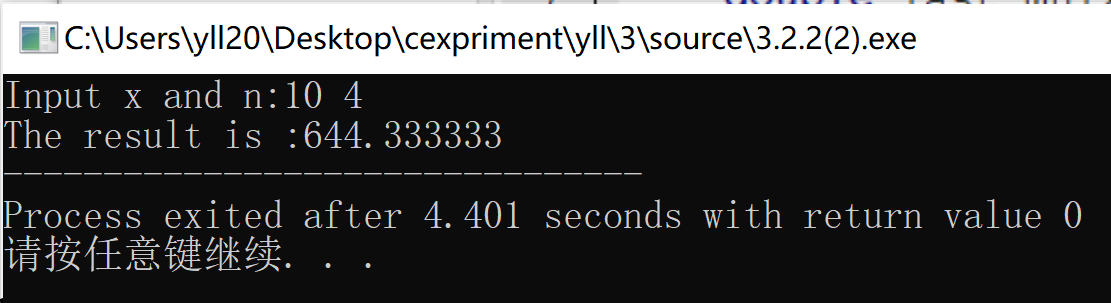


图3-3 程序替换题2的运行结果

**3.2.3 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);

}

**解答：**

1. 如图3-4所示。

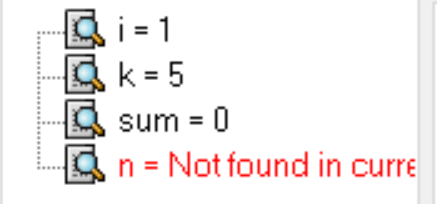


图3-4 跟踪调试题的变量查看

1. 变量值如图3-4所示。返回后停留在printf("i=%d\tthe sum is %ld\n",i,sum);如图3-5所示。

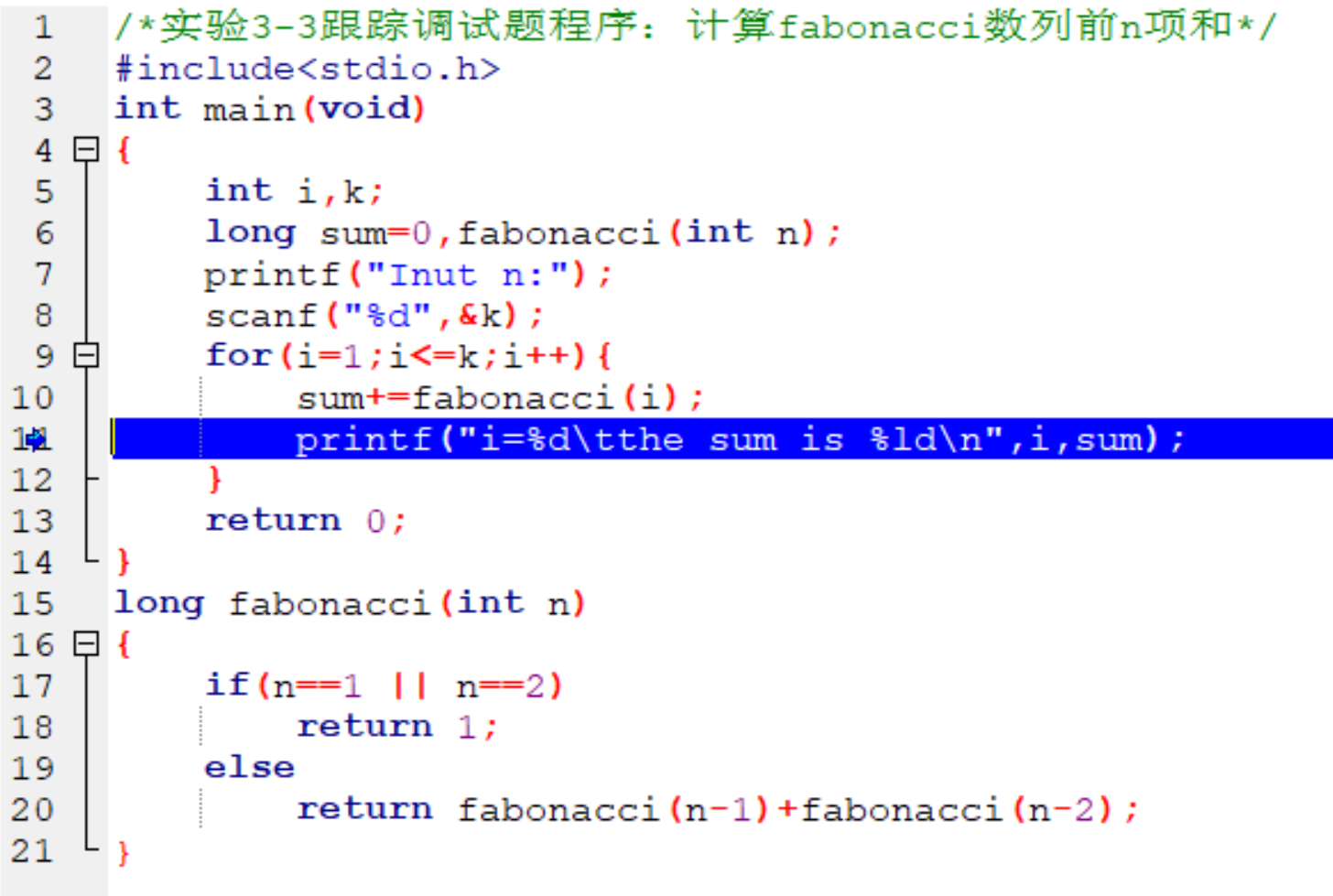


图3-5 跟踪调试题的单步调试

1. 跳转到fabonacci函数的开头。如图3-6所示。

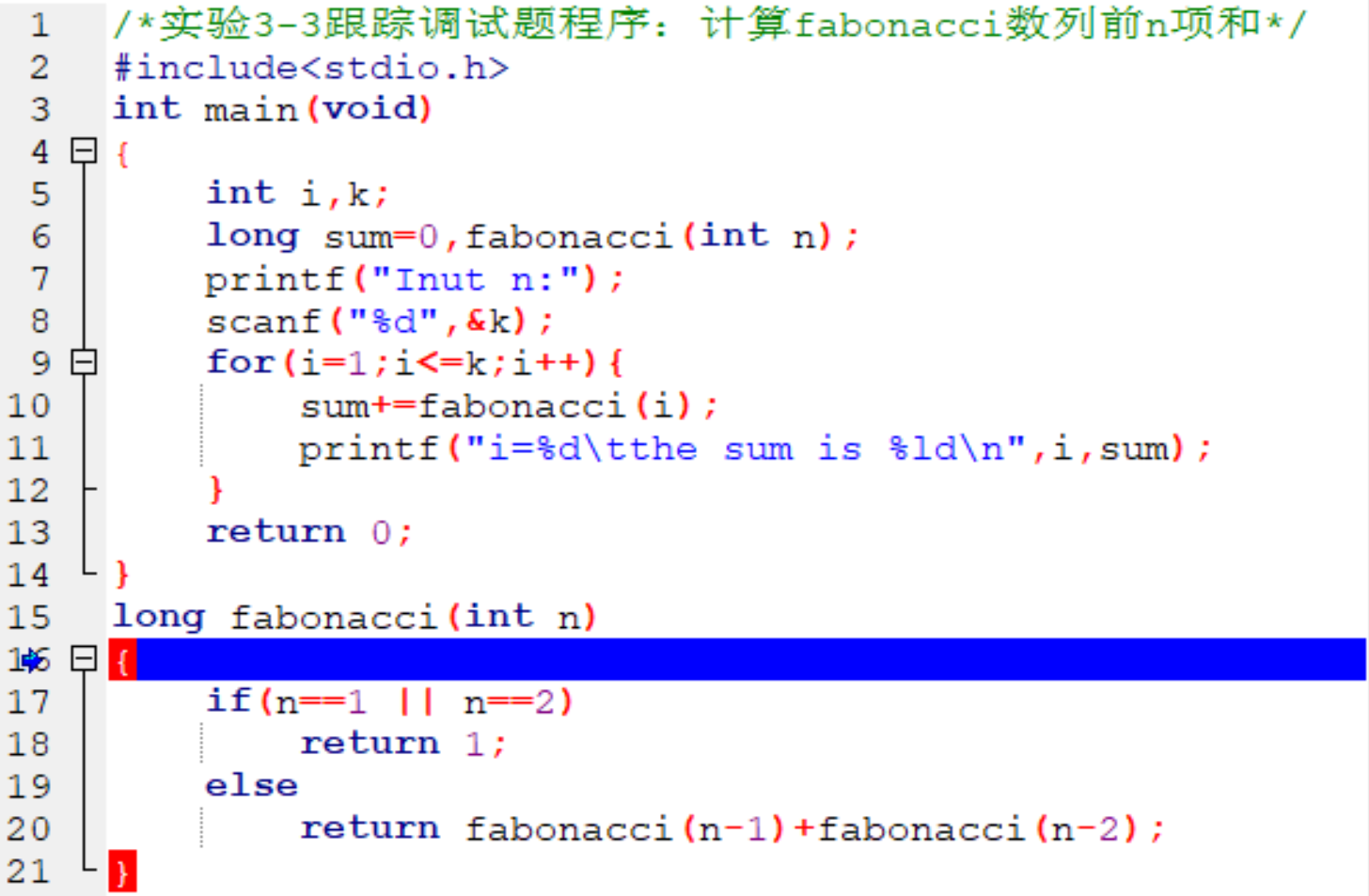


图3-6 跟踪调试题的单步调试

1. 递归函数不断调用自本身并等待最后一层调用返回结果后一层一层返回。最后直接返回到下一行printf。
2. sum和k是main函数内声明的变量,可见域只在main函数内。

**3.2.4 程序设计**

（1）编程验证歌德巴赫猜想：一个大于等于4的偶数都是两个素数之和。要求设计一个函数对其形参n验证哥德巴赫猜想，并以“n=n1+n2”的形式输出结果。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。

**解答：**

1） 算法流程如图3-6所示。



图3-6 程序设计题1的程序流程图

2）源程序清单

#include <stdio.h>

#include <math.h>

int is\_veg(int n);

int main(void)

{

int num, i = 2;

scanf("%d", &num);

while(!(is\_veg(i) && is\_veg(num-i))) //两个都要是质数,否则下一个

i++;

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

return 0;

}

int is\_veg(int n)

{

int i, j;

for (i = 2, j = (int)sqrt(n); i <= j; i++)

if (n % i) //不能整除就继续

continue;

else

return 0;

return 1;

}

3）测试

（a） 测试数据：

56,74

（b） 对应测试数据的运行结果截图

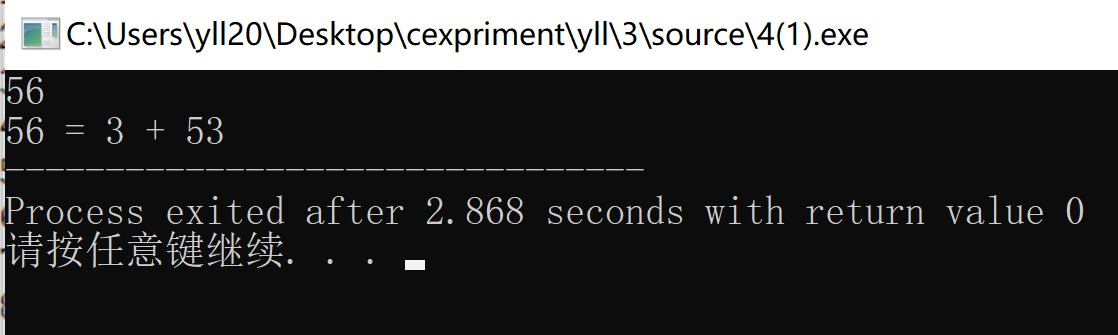


图3-7 程序设计题1的运行结果图

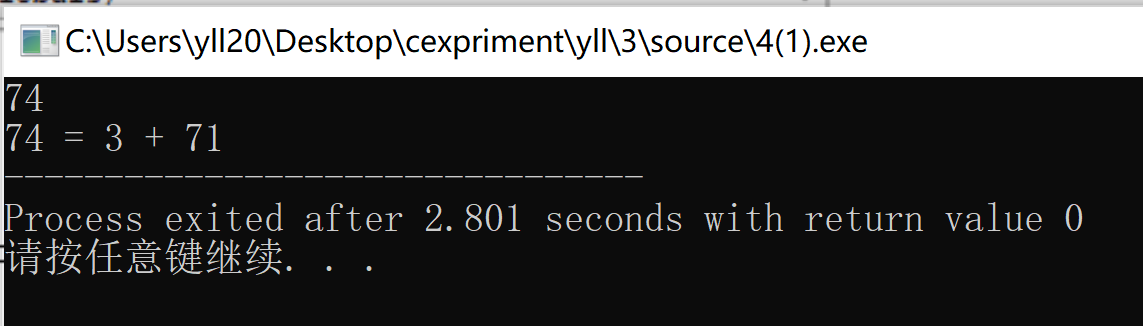


图3-8 程序设计题1的运行结果图

（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）之和恰好等一它本身。例如6=1+2+3，28=1+2+4+7+14等。编程寻找108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，则以n的真因子之和的形式输出结果，例如“6=1+2+3”；否则，输出“not a perfect number”，例如“5 is not a perfect number”。

在main函数中调用该函数求108以内的所有完全数。

**解答：**

1） 算法流程如图3-9所示。



图3-9 程序设计题2的程序流程图

2）源程序清单

#include <stdio.h>

#include <math.h>

int isPrime(int n);

void quickSort(int \*array, int left, int right);

int isPrime(int n)

{

if(n==2 || n==3) return 1;

if(n%6!=1 && n%6!=5) return 0;

for(int i=2;i<=sqrt(n)+1;i+=1)

{

if(n%i==0) return 0;

}

return 1;

}

void quickSort(long long \*array, int left, int right)

{

if(left < right)

{

long long pivot = array[left];

int low = left, high = right;

while(low < high)

{

while(array[high] >= pivot && low < high)

high--;

array[low] = array[high];

while(array[low] <= pivot && low < high)

low++;

array[high] = array[low];

}

array[low] = pivot;

quickSort(array, left, low - 1);

quickSort(array, low + 1, right);

}

}

int main()

{

long long mp, n = 0, i,shit;

long long b[100005];

for (i = 2; i <= 20; i++)

{

mp = pow(2,i) - 1;

if (isPrime(mp))

{

shit = (mp\*(mp+1))>>1;

if(shit<=1e8)

{

printf("%lld=",shit);

int z=0;

for(long long i=1;i\*i<=shit;i++)

{

if (shit%i==0)

{

b[z++]=i;

if(i\*i!=shit)

{

b[z++]=shit/i;

}

}

}

quickSort(b,0,z-1);

for (long i=0;i<z;i++)

{

if (i==z-1)

{

printf("\n");

break;

}

if (i==z-2)

{

printf("%lld",b[i]);

continue;

}

printf("%lld+",b[i]);

}

printf("\n");

}

}

}

return 0;

}

3）测试

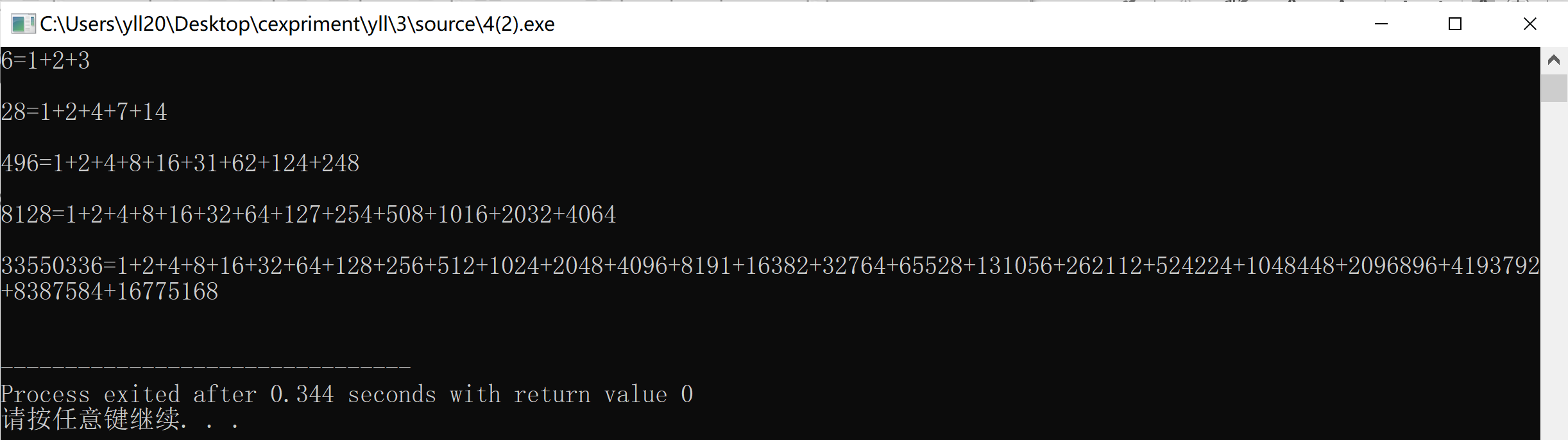


图3-10 程序设计题2的运行结果图

（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，则返回1；否则，返回0。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。

**解答：**

1. 算法流程如图1.1所示。



图3-11 程序设计题3的程序流程图

2）源程序清单

#include<stdio.h>

#include<math.h>

int main()

{

int cache[100];

int n;

scanf("%d",&n);

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

{

cache[i]=pow(i,n);

}

int x = 0;

for(x=pow(10,n-1);x<pow(10,n);x++)

{

int temp = x;

int sum = 0, a = 0, i;

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

{

a = temp%10; //最后一位

temp /= 10; //去掉后一位

sum += pow(a,n);

}

if(sum == x)

{

printf("%10d",x);

}

}

printf("\n");

return 0;

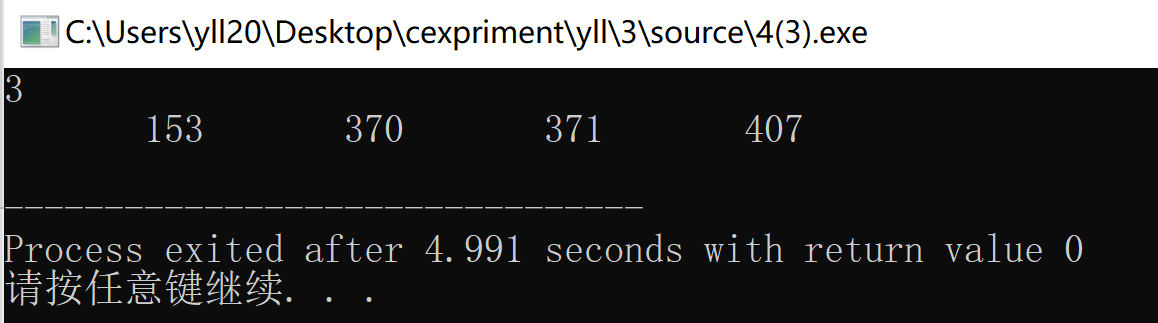
}

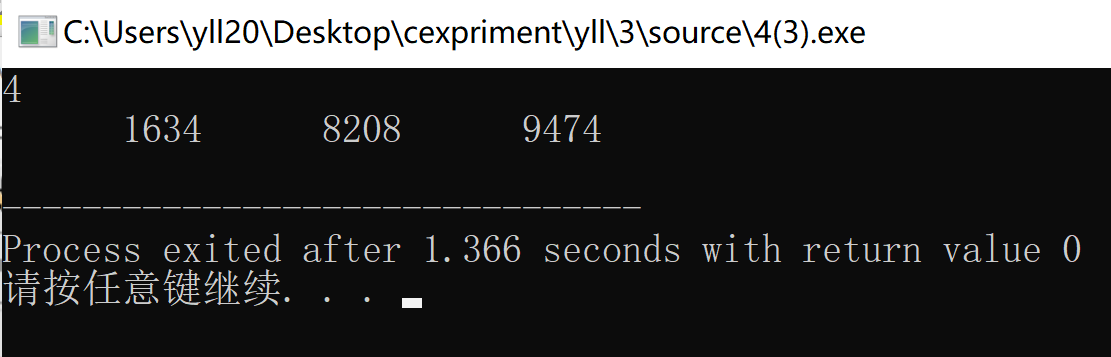
3）测试

（a） 测试数据：

3,4,5

（b） 对应测试数据的运行结果截图





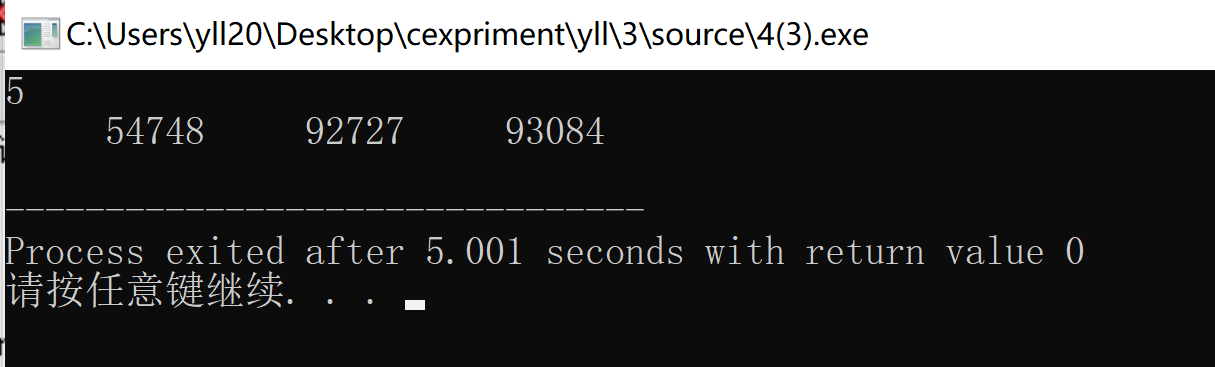


图3-12 程序设计题3的运行结果图

## 3.3 实验小结

实验中体会到了递归函数自己调用自己的思路，同时体会到了递归函数在开销上的局限性。对计算完全数的算法进行了细致的优化，采用了筛素数的方法，捅死提高了自身规划算法的能力。

# 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

**4.2.1 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、将换两数\*/

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：

1) 第4行不能直接交换两个数，正确形式为：

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

（2）错误修改后运行结果：

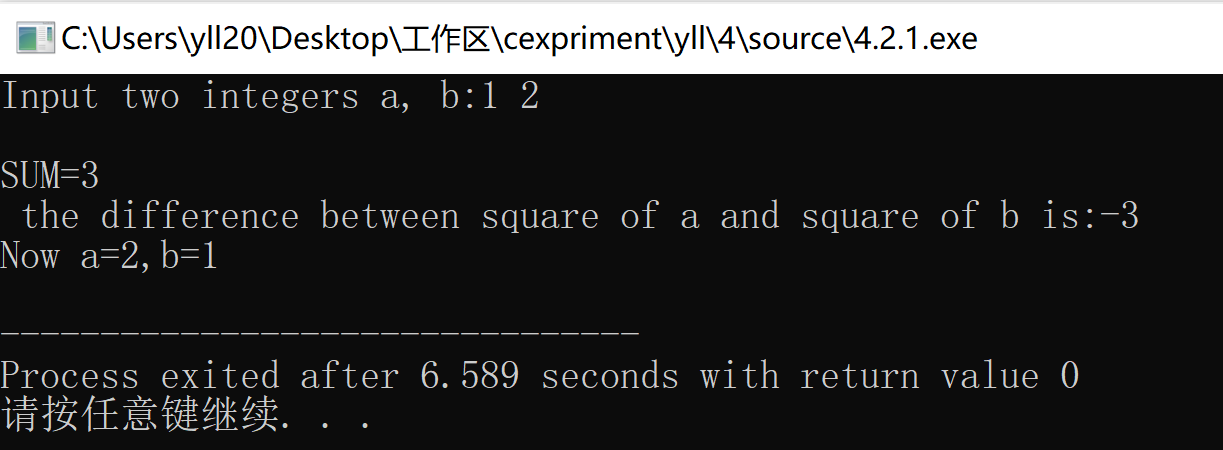
****

图4-1 改错题1的运行结果

**3.2.2 程序修改替换题**

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：

/\*实验4-2程序修改替换题程序\*/

#include<stdio.h>

#define max(a,b,c) a>b?(a>c?a:c):(b>c?b:c) //求三个数的最大值

float sum(float x, float y); //函数在使用之前应该先声明

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;

}

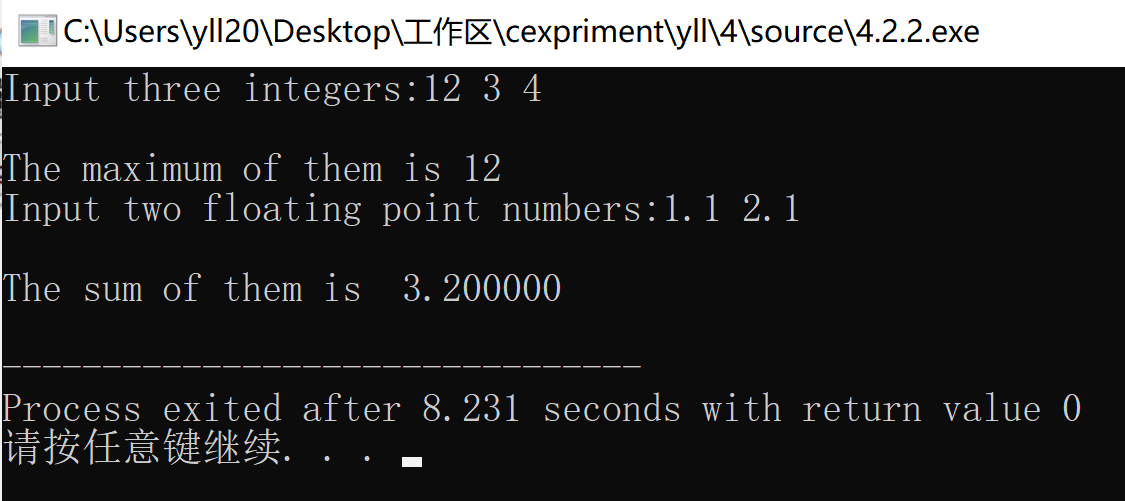


图4-2 程序替换题1的运行结果

**4.2.3 跟踪调试题**

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)

{

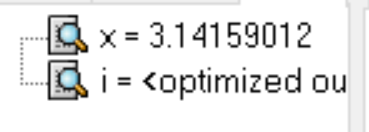
int i=x;

return i;

}

**解答：**

（1）如图4-3所示。

****

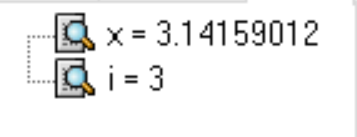
****

图4-3 跟踪调试题的变量查看

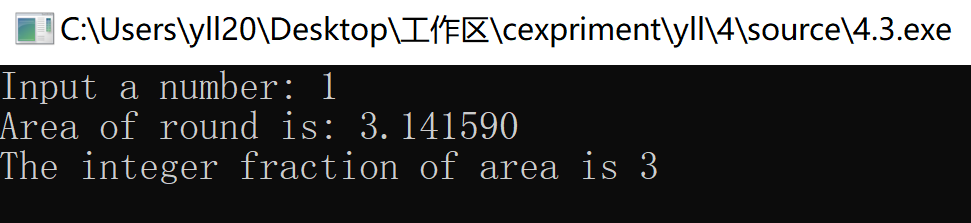


图4-4 跟踪调试题的运行结果

**4.2.4 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。

**解答：**

1） 算法流程如图4-5所示。



图4-5 程序设计题1的程序流程图

2）源程序清单

#include<stdio.h>

#include<math.h>

#define get\_s(a,b,c) ((a+b+c)/2)

#define get\_area(s,x,y,z) (sqrt(s\*(s-x)\*(s-y)\*(s-z)))

int main()

{

double a,b,c;

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

double s=get\_s(a,b,c);

double area=get\_area(s,a,b,c);

printf("%lf",area);

return 0;

}

3）测试

（a） 测试数据：

3 4 5

（b） 对应测试数据的运行结果截图

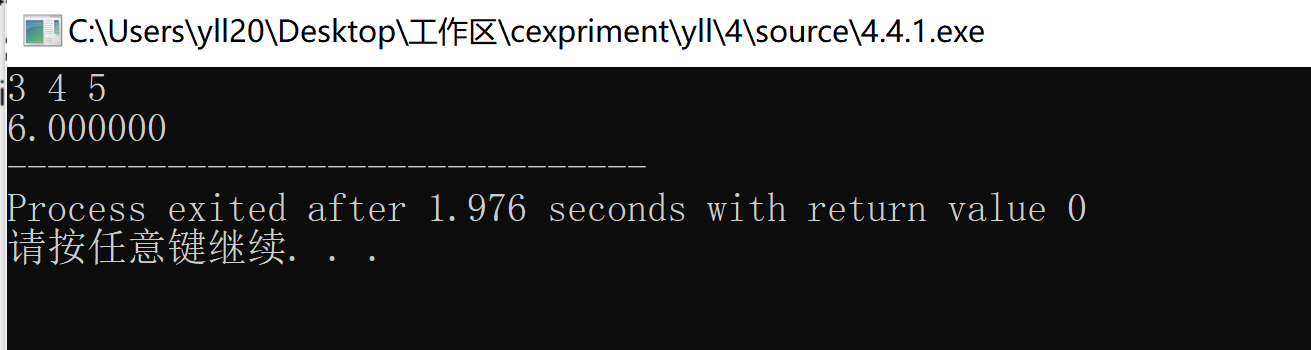


图4-6 程序设计题1的运行结果图

（2）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。

1） 算法流程如图4-7所示。



图4-7 程序设计题2的程序流程图

2）源程序清单

#include<stdio.h>

#define CHANGE 1

int main()

{

char a[100];

#if CHANGE==0

scanf("%s",a);

printf("%s",a);

#elif CHANGE==1

scanf("%s",a);

int i=0;

while(a[i])

{

if ((a[i]<='z')&&(a[i]>='a'))

{

a[i]=a[i]-32;

}

else if ((a[i]<='Z')&&(a[i]>='A'))

{

a[i]=a[i]+32;

}

i++;

}

printf("%s",a);

#endif

return 0;

}

3）测试

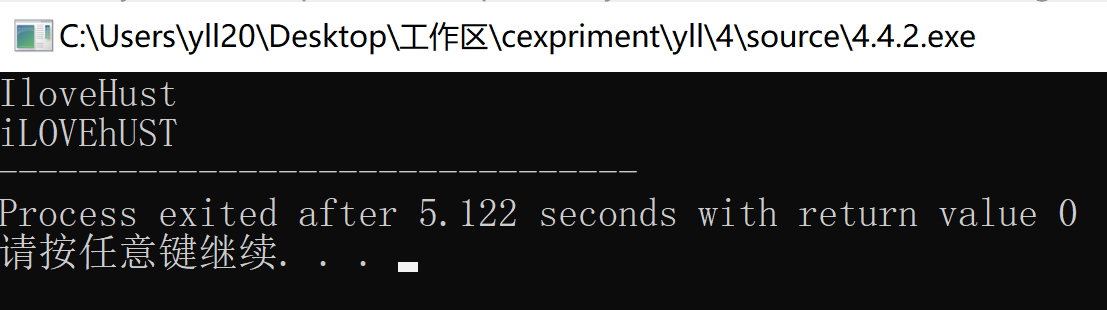


图4-8 程序设计题2的运行结果图

## 4.3 实验小结

实验中体会到了宏定义对计算的方便性带来的好处，同时也体会到可其容易出错的缺点，替换的时候只是单纯地替换内容，定义的时候要注意优先级和打上括号。

# 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

**5.2.1 源程序改错与跟踪调试**

下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。

（2）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。

/\*实验5-1程序改错与跟踪调试题程序\*/

1 #include<stdio.h>

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

3 void strdelc(char [],char );

4 int main(void)

5 {

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

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

8 strcate(b,a);

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

10 strdelc(b, 'a');

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

12 return 0;

13 }

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

15 {

16 int i = 0, j = 0;

17 while(t[i++]) ;

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

19 }

20 void strdelc(char s[], char c)

21 {

22 int j,k;

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

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

25 }

**解答：**

（1）错误修改：

1) 第17行最后统计的i会多一位，正确形式为：

while(t[i]) i++;

2) 第23行应该判断s[k]不为\0，正确形式为：

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

3) 第24行应该在字符串最后添加\0，正确形式为在24行后添加：

s[k]='\0';

（2）错误修改后运行结果：

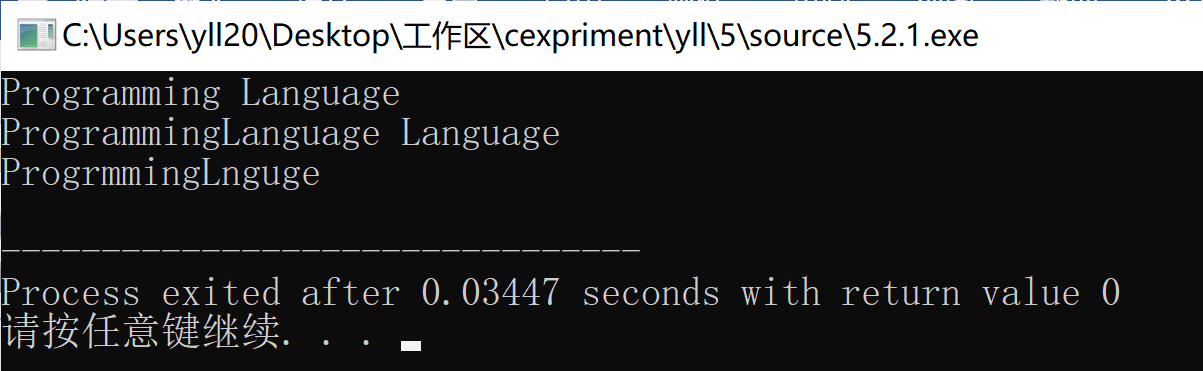
****

图5-1 改错题1的运行结果

**5.2.2 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;

}

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：**

1. 替换后的程序如下所示：

#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[i-1]; /\* 将报数为N的人的编号存入数组b j=0的时候存最后一个人\*/

if(j)

{

for(k = --j; k < i; k++)

{

a[k]=a[k+1]; //后面的人向前

} /\* 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;

}

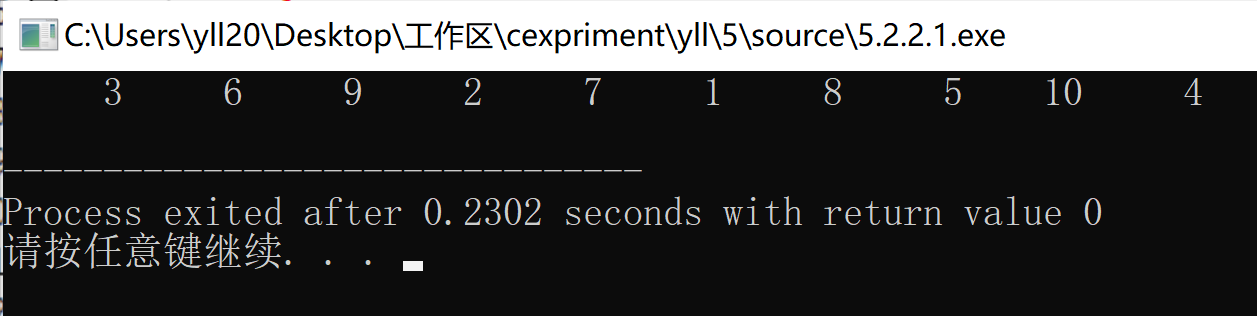


图5-2 程序替换题1的运行结果

1. 替换后的程序如下所示：

#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>0; i--)

{

for(k=0;;)

{

k=a[j]?k+1:k; //a[j]=0则k不增加，相当于没报数

if(k==N)

break;

if(++j > M-1)

j = 0;

}

b[M-i] = a[j];

a[j]=0;

}

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

{

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

}

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

return 0;

}

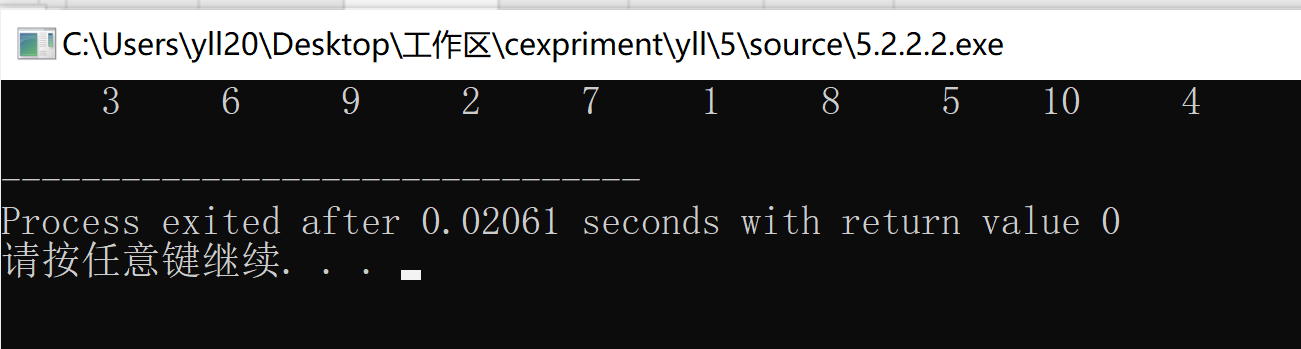


图5-3 程序替换题1的运行结果

**5.2.3 程序设计**

（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。

**解答：**

1） 算法流程如图5-4所示。



图5-4 程序设计题1的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

int n;

char a[100];

scanf("%d",&n);

int i;

for(i=0;i<16;i++)

{

a[i]=(n&(1<<(16-i-1)))?'1':'0'; //如果还有值说明那一位是1，否则是0

}

for(i=0;i<16;i++)

{

printf("%c",a[i]);

}

return 0;

}

3）测试

（a） 测试数据：

256

（b） 对应测试数据的运行结果截图

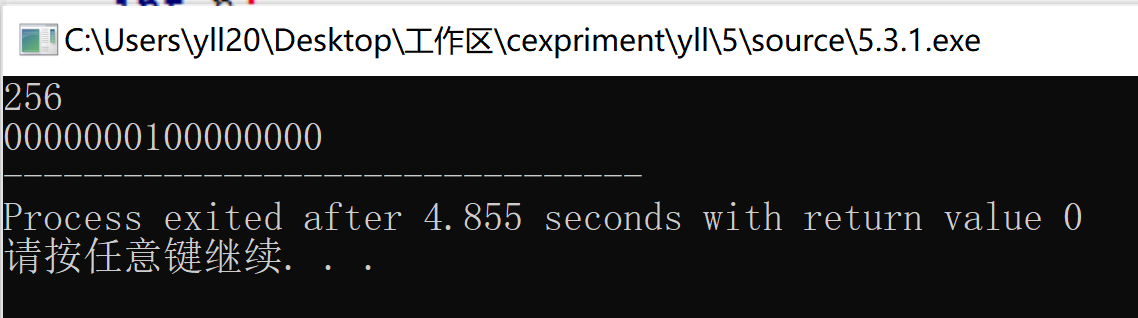


图5-5 程序设计题1的运行结果图

（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输入n个学生的姓名和C语言课程的成绩。

②将成绩按从高到低的次序排序，姓名同时进行相应调整。

③输出所有学生的姓名和C语言课程的成绩。

1） 算法流程如图5-6所示。



图5-6 程序设计题2的程序流程图

2）源程序清单

#include<stdio.h>

#include<windows.h>

#include <string.h>

char name[100][10]={};

char name\_temp[100][10]={};

int score[100]={0};

int score\_temp[100]={0};

void input();

void sort();

void output();

void search();

void sort2();

int main()

{

static char temp;

system("title 学生成绩管理系统 by yll");

start:

system("Cls");

printf("-----------------------\n");

printf("| 学生成绩管理系统 |\n");

printf("-----------------------\n");

printf("| 1 |输入学生成绩|\n");

printf("-----------------------\n");

printf("| 2 | 按成绩排序 |\n");

printf("-----------------------\n");

printf("| 3 |输出所有成绩|\n");

printf("-----------------------\n");

printf("| 4 | 查找成绩 |\n");

printf("-----------------------\n");

printf("| 5 | 退出系统 |\n");

printf("-----------------------\n");

scanf("%c",&temp);

if (temp=='1')

{

input();

}

if (temp=='2')

{

sort();

}

if (temp=='3')

{

output();

}

if (temp=='4')

{

search();

}

if (temp=='5')

{

return 0;

}

else

{

printf("输入错误！\n");

}

goto start;

return 0;

}

void input()

{

system("Cls");

int i=0;

printf("-----------------------------------\n");

printf("|请这样输入学生成绩： |\n");

printf("|张三 96 |\n|李四 98 |\n");

printf("|以ctrl+Z结束录入 |\n");

printf("-----------------------------------\n");

while(scanf("%s",&name[i][0])!=EOF)

{

scanf("%d",&score[i]);

i++;

}

}

void output()

{

system("Cls");

int i=0;

printf("-----------------\n");

printf("|姓名\t|成绩\t|\n");

printf("-----------------\n");

while(score[i])

{

printf("|%s\t|%d\t|\n",name[i],score[i]);

printf("-----------------\n");

i++;

}

printf("请按任意键返回......\n");

getchar();

getchar();

}

void sort()

{

int len=0;

while(score[len])

{

len++;

}

int i, j;

int v;

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

{

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

{

if(score[i] < score[j])//如前面的比后面的小，则交换。

{

v = score[i];

score[i] = score[j];

score[j] = v;

char temp[10];

strcpy(temp,name[i]);

strcpy(name[i],name[j]);

strcpy(name[j],temp);

}

}

}

output();

}

void sort2()

{

int len=0;

while(score[len])

{

len++;

}

int i, j;

int v;

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

{

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

{

if(score\_temp[i] < score\_temp[j])//如前面的比后面的小，则交换。

{

v = score\_temp[i];

score\_temp[i] = score\_temp[j];

score\_temp[j] = v;

char temp[10];

strcpy(temp,name\_temp[i]);

strcpy(name\_temp[i],name\_temp[j]);

strcpy(name\_temp[j],temp);

}

}

}

}

void search()

{

system("Cls");

printf("-----------------------\n");

printf("| 输入要查找的成绩 |\n");

printf("-----------------------\n");

int target = 0;

scanf("%d",&target);

int len=0;

while(score[len])

{

score\_temp[len] = score[len];

strcpy(name\_temp[len],name[len]);

len++;

}

int left = 0;

int right = len - 1;

int flag=1;

sort2();

while(flag)

{

while(left <= right)

{

int mid = left+(right-left)/2; //防止溢出

if(score\_temp[mid] == target)

{

printf("-----------------\n");

printf("|%s\t|%d\t|\n",name[mid],score[mid]);

printf("-----------------\n");

score\_temp[mid] = -2; //不会真有人考-2分吧

}

else if (score\_temp[mid] < target)

right = mid - 1;

else if (score\_temp[mid] > target)

left = mid + 1;

}

flag = 0;

printf("查找结束......\n");

}

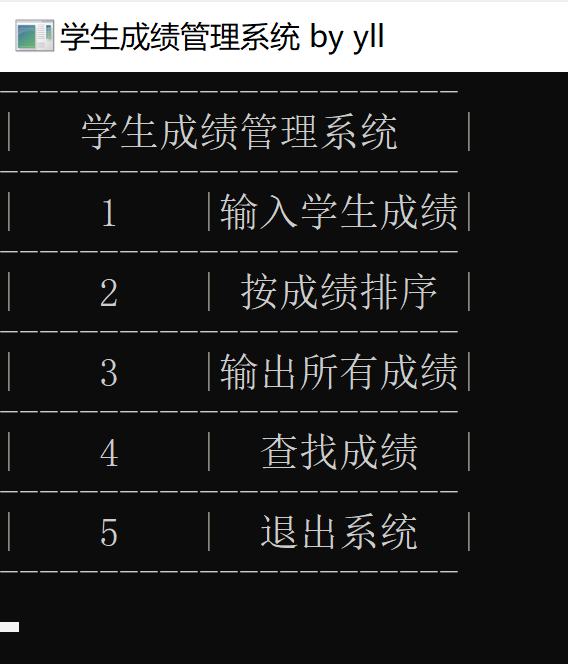
printf("请按任意键返回......\n");

getchar();

getchar();

}

3）测试



****



图5-7 程序设计题2的运行结果图

**5.2.4 程序设计选做题**

编写并上机调试运行能实现以下功能的函数和程序。

编写函数strnins(s,t,n),其功能是：可将字符数组 t中的字符串插入到字符数组 s中字符串的第n个字符的后面。

**解答：**

1） 算法流程如图5-9所示。



图5-8 选做题的程序流程图

2）源程序清单

#include<stdio.h>

void strnins(char a[],char b[],int n);

int main()

{

char a[100],b[100];

int n=0;

scanf("%s %s %d",&a,&b,&n);

strnins(a,b,n);

printf("%s",a);

return 0;

}

void strnins(char a[],char b[],int n)

{

int i=n;

int j=0;

while(b[j])

{

a[i] = b[j];

i++;

j++;

}

a[i+1]='\0';

}

3）测试

（a） 测试数据：

ilovechina ilovehust 5

（b） 对应测试数据的运行结果截图

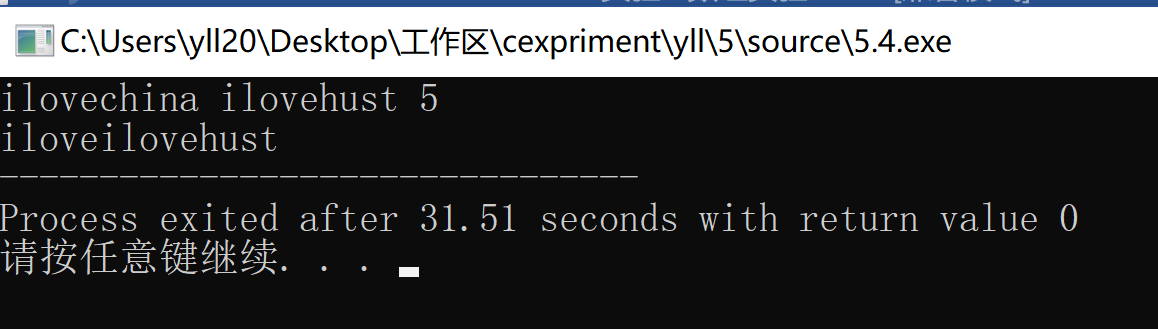


图5-9 选做题的运行结果图

## 5.4 实验小结

实验中体会到了数组在存储数据上的方便性，掌握了了数组的说明、初始化和使用，同时也发现使用数组的时候一定要小心访问非法内存的问题，否则会产生很多问题。还掌握了字符串的常见处理方法和二分查找的思想以及选择排序的办法。

# 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 实验内容

**6.2.1 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

1 #include<stdio.h>

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

3 int main(void)

4 {

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

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

7 scanf("%s", s2);

8 strcopy(s1, s2);

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

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

11 scanf("%s", s2);

12 s3 = strcopy(s1, s2);

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

14 return 0;

15 }

16

17 /\*将字符串s复制给字符串t，并且返回串t的首地址\*/

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

19 {

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

21 return (t);

22 }

**解答：**

（1）错误修改：

1) 第5行指针应当指向具体的内存，正确形式为：

char str1[20],str2[20],str3[20];

s1=str1,s2=str2,s3=str3;

2) 第6行和12行printf不用s2，正确形式为：

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

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

3) 第21行应该返回字符串原始的开始地址，正确形式为：

char \*tmp=t; //先暂存串t开头的位置来返回

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

return (tmp);

（2）错误修改后运行结果：

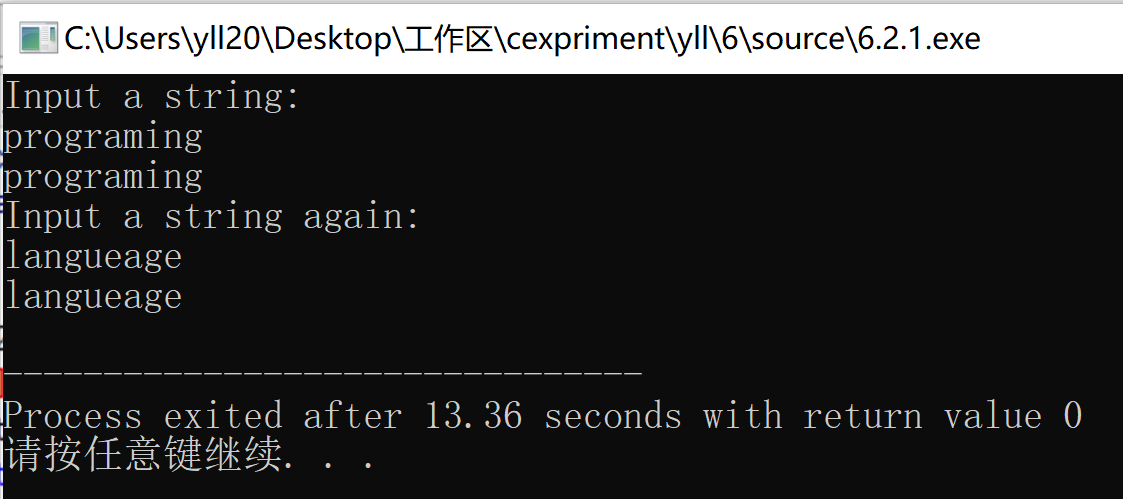
****

图6-1 改错题1的运行结果

**6.2.2 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-j-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]);

return 0;

}

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：**

1. 替换后的程序如下所示：

#include<stdio.h>

#include<stdlib.h> //malloc在stdlib里面

#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; //从后面来看temp是指针

int i, j;

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

{

for (j=0; j<size-i-1; j++) //此处应为j<size-i-1;

if (strcmp(s[j],s[j+1])>0) //升序排序

{

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); //此处将t存入新开辟的空间

}

strsort(s,i); //此处调用排序函数

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

return 0;

}

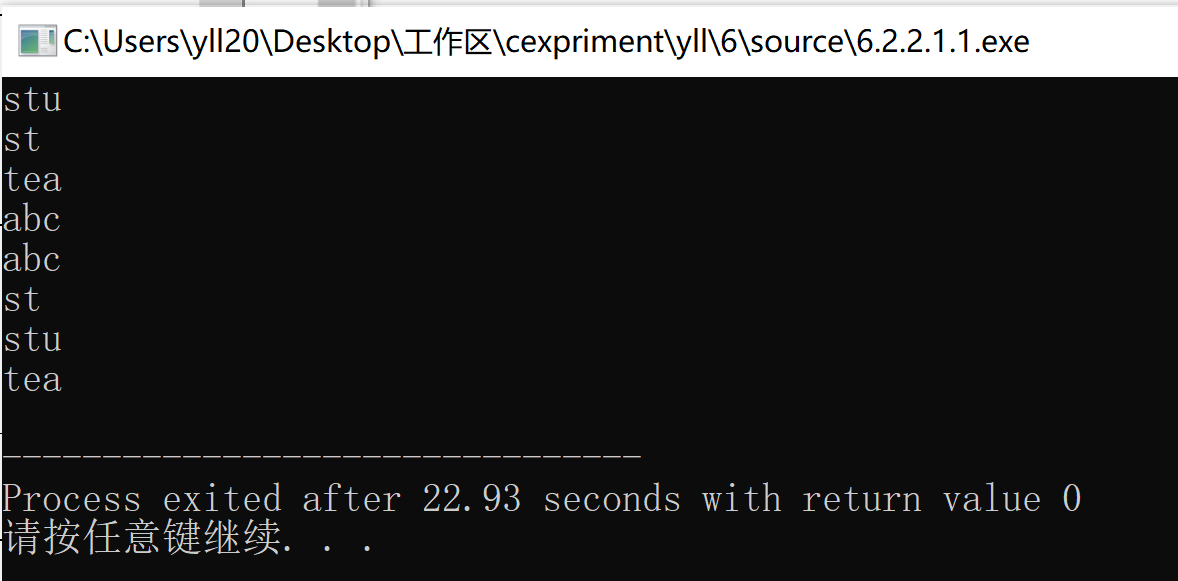


图6-2 程序替换题1的运行结果

1. 替换后的程序如下所示：

#include<stdio.h>

#include<stdlib.h> //malloc在stdlib里面

#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-j-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); //此处将t存入新开辟的空间

}

strsort(s,i+2); //此处调用排序函数

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

return 0;

}

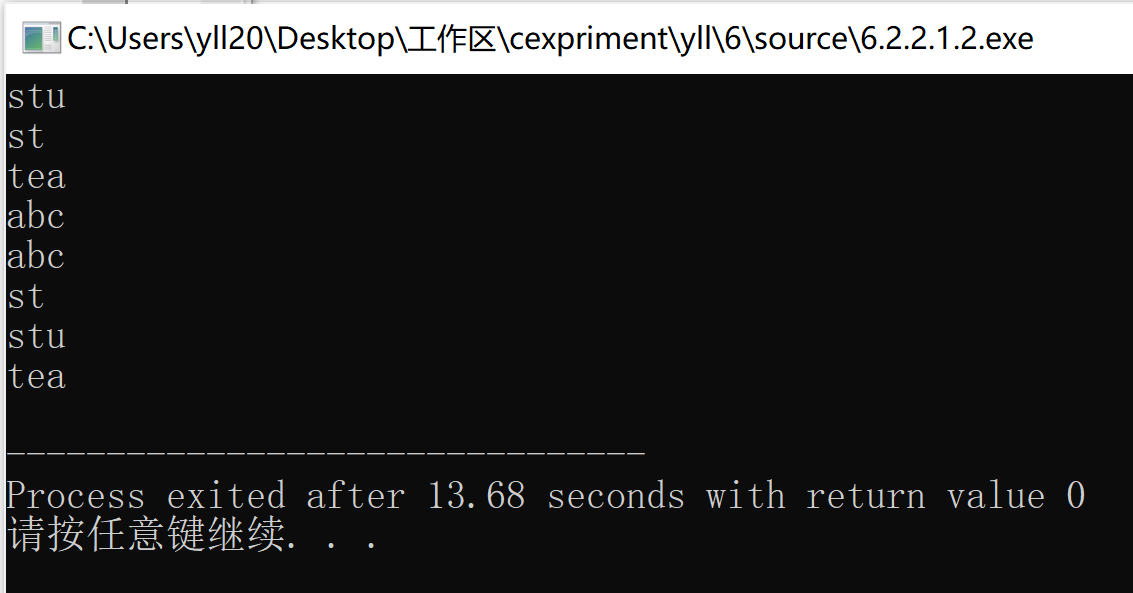


图6-3 程序替换题1的运行结果

（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.

# include<stdio.h>

# include<string.h>

int main (void)

{

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

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 = strok; break;

case 4: p = goto down;

}

getchar();

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

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

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

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

result = \_\_\_\_\_\_\_\_\_\_\_(a, b);

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

}

down:

return 0;

}

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：**

1. 替换后的程序如下所示：

# include<stdio.h>

# include<string.h>

int main (void)

{

char \*(\*p)(char \*a,char \*b);

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");

gets(a); //读a

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

gets(b); //读b

result = p(a, b); //干活

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

}

down:

return 0;

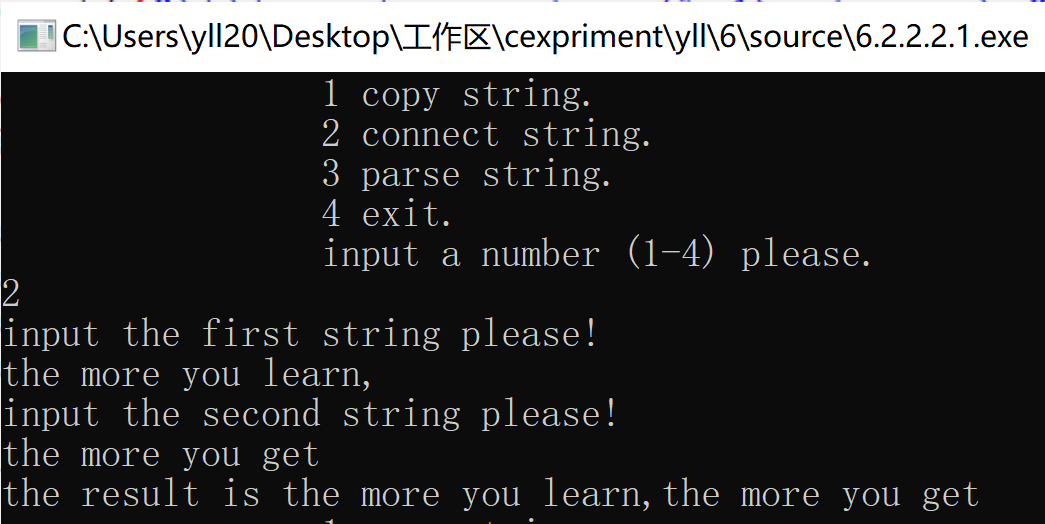
} 

图6-4 程序替换题2的运行结果

1. 替换后的程序如下所示：

# include<stdio.h>

# include<string.h>

int main (void)

{

char \*(\*p[4])(char \*a,char \*b);

p[1]=strcpy;

p[2]=strcat;

p[3]=strtok;

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);

if(choice==4) goto down;

getchar();

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

gets(a); //读a

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

gets(b); //读b

result = p[choice](a, b); //干活

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

}

down:

return 0;

}

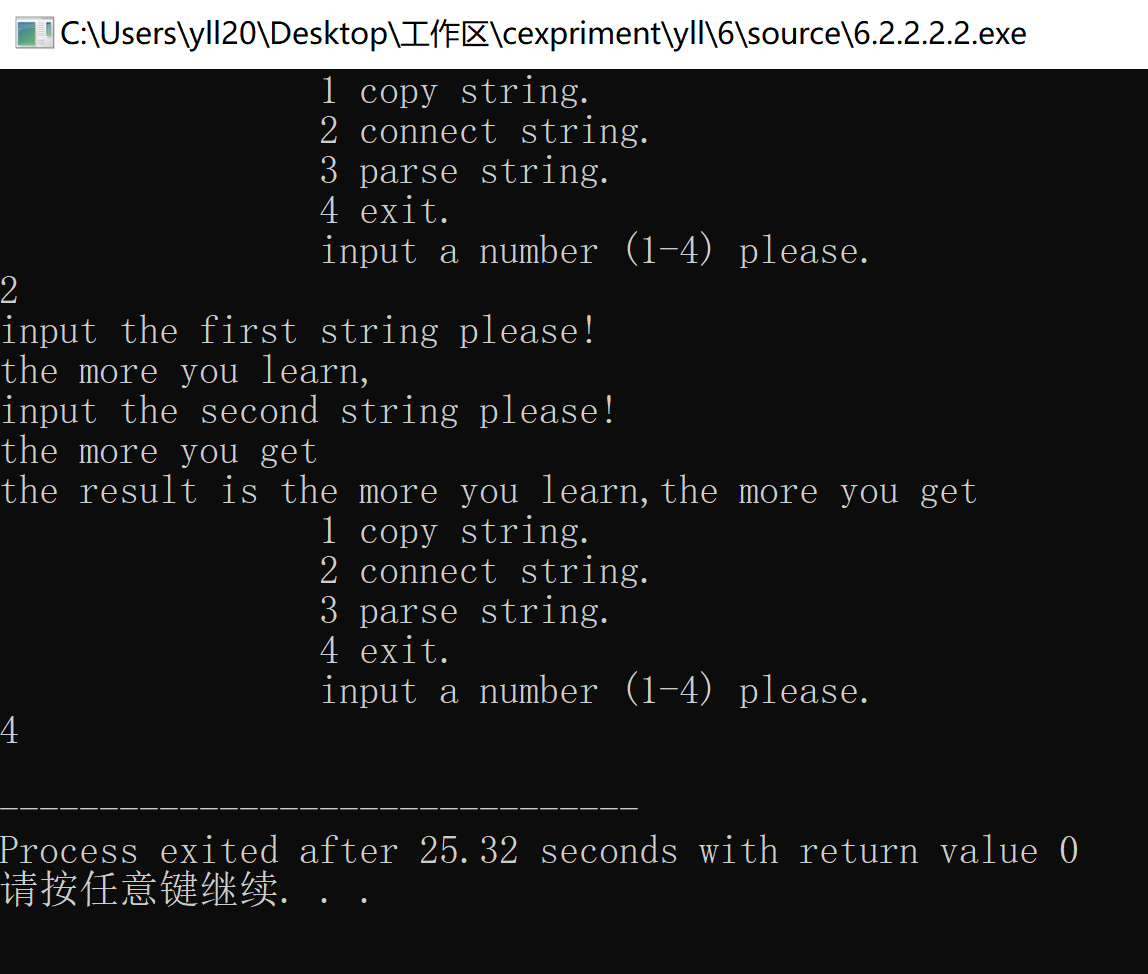


图6-5 程序替换题1的运行结果

**6.2.3 跟踪调试题**

请按下面的要求对源程序进行操作，并回答问题和排除错误。

（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（2）排除错误，使程序输出结果为：there is a boat on the lake.

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

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

}

char \*strcpy(char \*s,char \*t)

{

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

return (s);

}

**解答：**

1）修改后程序如下

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[60],b[60]="there is a boat on the lake."; //a长度小了

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

}

char \*strcpy(char \*s,char \*t)

{

char \*tmp=s;

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

return (tmp);//返回初始的指针

}

3）测试

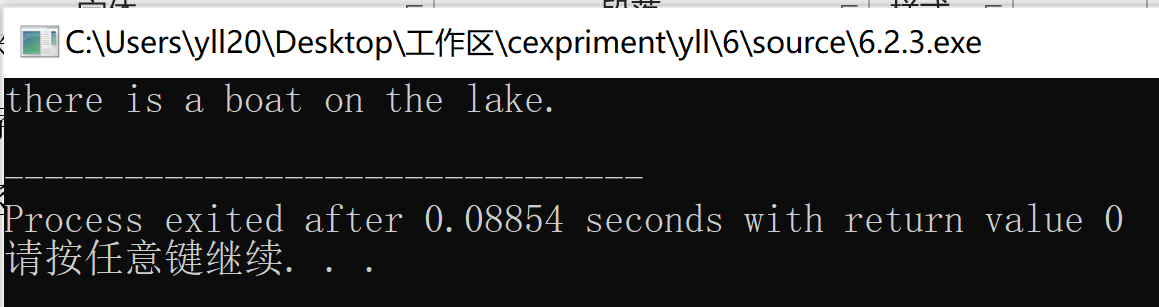


图6-6 程序设计题1的运行结果图

**6.2.4 编程设计题**

（1）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。试从该长整型变量的高字节开始，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，要求通过指针取出每字节。

**解答：**

1） 算法流程如图5-4所示。



图6-7 编程设计题1的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

int num=0;

scanf("%d",&num);

char \*p=&num;

for(int i=3;i>=0;i--)

{

printf("%d:high:%x low:%x\n",4-i,(\*(p+i)>>4)&0x000F,(\*(p+i))&0x000F);

}

return 0;

}

3）测试

（a） 测试数据：

1012

（b） 对应测试数据的运行结果截图

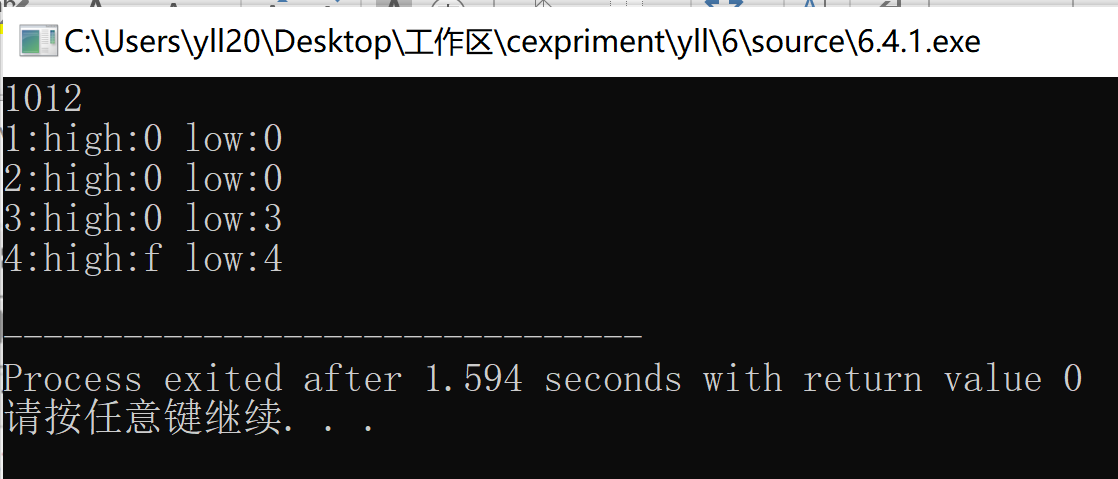


图6-8 程序设计题1的运行结果图

（2）旋转是图像处理的基本操作，编程实现一个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。例如，输入：

2 3

1 5 3

3 2 4

则输出：

3 4

5 2

1 3

**解答：**

1） 算法流程如图6-9所示。



图6-9 编程设计题2的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

int n,m;

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

int num[m][n];

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

{

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

{

scanf("%d",&num[m-1-j][i]);

}

}

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

{

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

{

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

}

printf("\n");

}

return 0;

}

3）测试

（a） 测试数据：

2 3

1 2 3

4 5 6

（b） 对应测试数据的运行结果截图

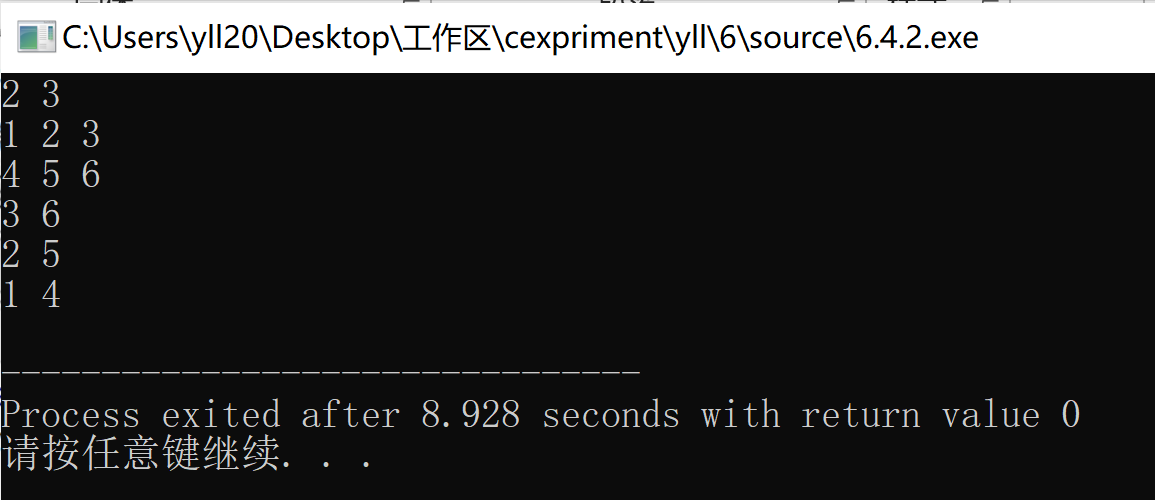


图6-10 程序设计题2的运行结果图

（3）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。

**解答：**

       1） 算法流程如图6-11所示。



图6-11 编程设计题3的程序流程图

2）源程序清单

#include<stdio.h>

#include<stdlib.h> //malloc在stdlib里面

#include<string.h>

#define N 3

char \*delete\_bank(char \*s)

{

while(\*s==' '||\*s=='\t')

{

s++;

}

return s;

}

int main()

{

int i;

char \*s[N], t[80];

char \*s1[N];

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

{

gets(t);

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

s1[i] = s[i];

strcpy(s[i],t); //此处将t存入新开辟的空间

s[i]=delete\_bank(s[i]);

}

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

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

{

free(s1[i]);

}

return 0;

}

3）测试

  （a） 测试数据：

        abcd bcv

qieiu ij

idwif iejf

  （b） 对应测试数据的运行结果截图

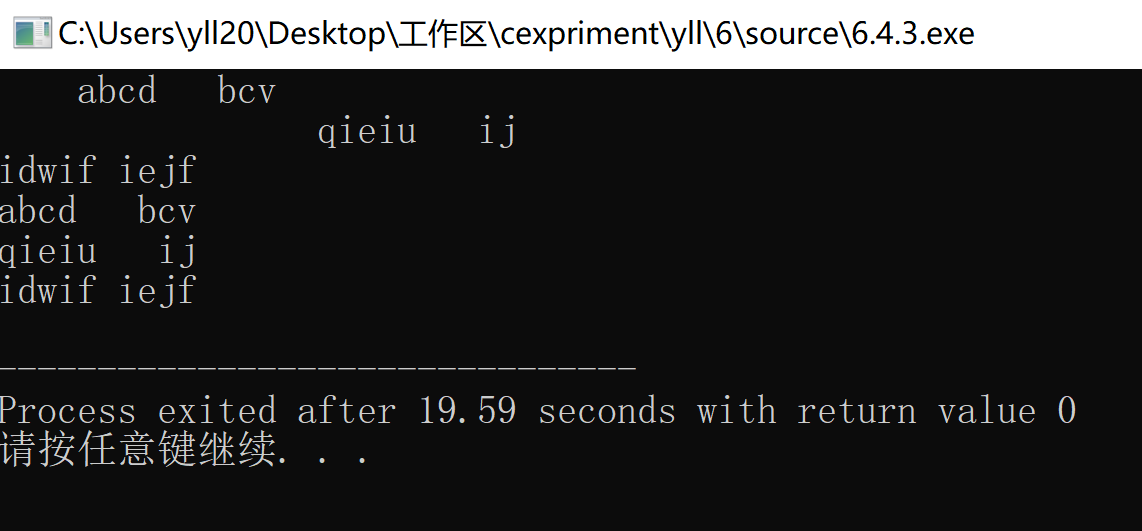


图6-12 程序设计题3的运行结果图

（4）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!

**解答：**

1）源程序清单

#include<stdio.h>

void scheduler(void);

void task0(void);

void task1(void);

void task2(void);

void task3(void);

void task4(void);

void task5(void);

void task6(void);

void task7(void);

void task0(void)

{

printf("task0 is called\n");

}

void task1(void)

{

printf("task1 is called\n");

}

void task2(void)

{

printf("task2 is called\n");

}

void task3(void)

{

printf("task3 is called\n");

}

void task4(void)

{

printf("task4 is called\n");

}

void task5(void)

{

printf("task5 is called\n");

}

void task6(void)

{

printf("task6 is called\n");

}

void task7(void)

{

printf("task7 is called\n");

}

void execute(void (\*p2[8])(), int i)

{

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

{

p2[j]();

}

}

void scheduler(void)

{

void (\*p[8])(void);

p[0]=task0;

p[1]=task1;

p[2]=task2;

p[3]=task3;

p[4]=task4;

p[5]=task5;

p[6]=task6;

p[7]=task7;

char cmd[10];

gets(cmd);

int i=0;

void (\*p2[8])(void);

while(cmd[i])

{

p2[i]=p[cmd[i]-'0'];

i++;

}

execute(p2,i);

}

int main()

{

scheduler();

return 0;

}

2）测试

  （a） 测试数据：

        1433223

  （b） 对应测试数据的运行结果截图

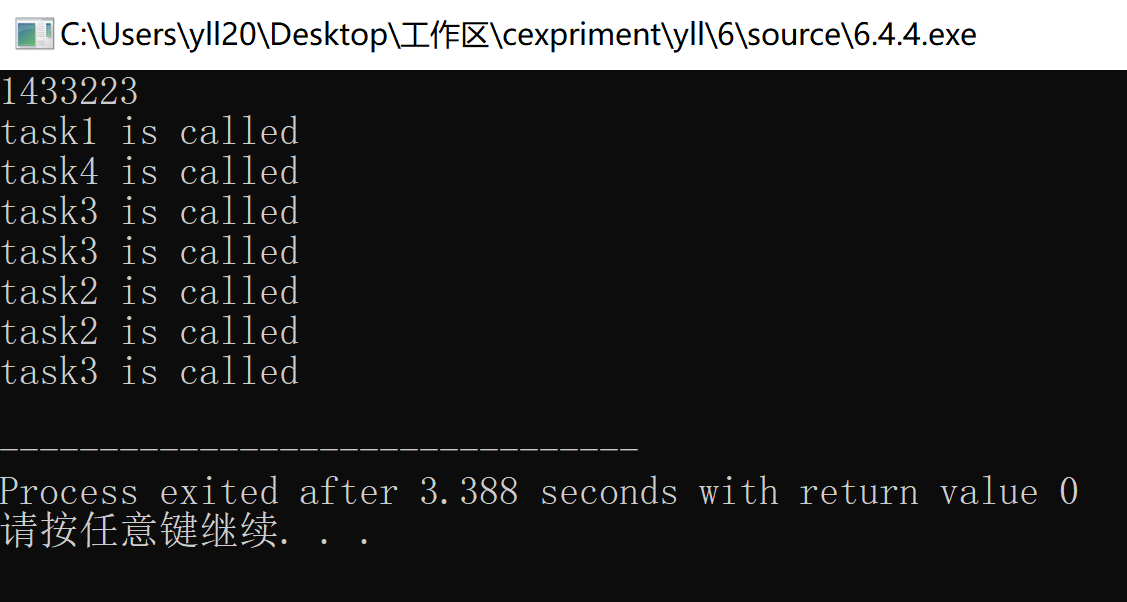


图6-13 程序设计题4的运行结果图

**6.2.5 程序设计选做题**

（1）设有N位整数和M位小数（N=20，M=10）的数据a,b。编程计算a+b并输出结果。

如：12345678912345678912.1234567891 + 98765432109876543210.0123456789

**解答：**

1）源程序清单

#include<stdio.h>

#include<string.h>

#include<math.h>

int main()

{

char s1[100],s2[100];

int a[100]={0},b[100]={0},c[100]={0};

scanf("%s%s",&s1,&s2);

int la,lb,lc;

//字符串转为数字，并且转置

la=strlen(s1);

lb=strlen(s2);

for(int i=0,j=0;i<la;i++)

{

if(s2[i]=='.')

{

continue;

}

a[la-i] = s1[i]-'0';

}

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

{

if(s2[i]=='.')

{

continue;

}

b[lb-i] = s2[i]-'0';

}

lc = 31;

for(int i=1;i<=lc;i++)

{

c[i] += a[i] + b[i];

c[i+1] = c[i]/10;

c[i] = c[i]%10;

}

if(c[11])

{

c[12]+=c[11];

for(int i=12;i<=lc;i++)

{

c[i+1] += c[i]/10;

c[i] = c[i]%10;

}

}

if(c[lc]==0&&lc>0) lc--;

for(int i=lc;i>11;i--)

{

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

}

printf(".");

for(int i=10;i>0;i--)

{

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

}

return 0;

}

2）测试

（a） 测试数据：

12345678912345678912.1234567891 98765432109876543210.0123456789

（b） 对应测试数据的运行结果截图

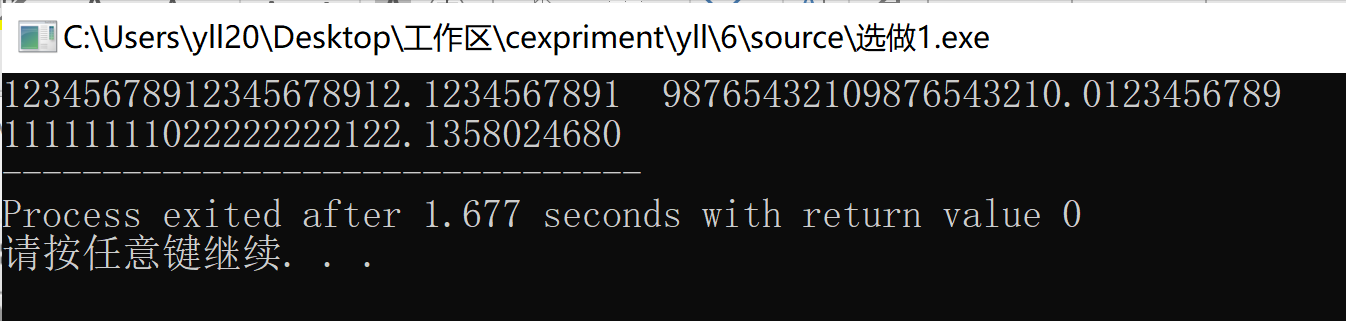


图6-14 选做题1的运行结果图

（2）编写使用复杂声明char \*(\*p[2])(const char \*,const char \*);的程序。

提示：p中元素可为strcmp、strstr等函数名。

**解答：**

1）源程序清单

#include<stdio.h>

#include<stdlib.h> //malloc在stdlib里面

#include<string.h>

#define N 4

char \*(\*p[2])(const char \*,const char \*);

/\*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; //从后面来看temp是指针

int i, j;

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

{

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

if (p[1](s[j],s[j+1])>0) //升序排序

{

temp = s[j];

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

s[j+1] = temp;

}

}

}

int main()

{

int i;

p[0]=strcpy;

p[1]=strcmp;

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

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

{

gets(t);

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

p[0](s[i],t); //此处将t存入新开辟的空间

}

strsort(s,i); //此处调用排序函数

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

return 0;

}

2）测试

  （a） 测试数据：

        stu

st abc

csc

cdc

  （b） 对应测试数据的运行结果截图

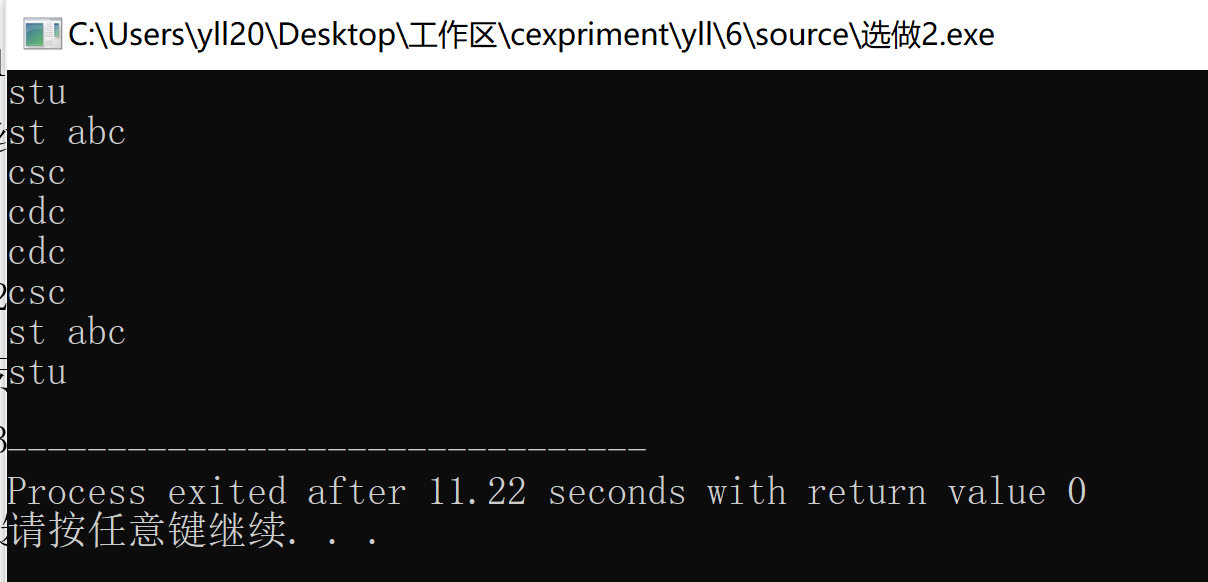


图6-15 选做题2的运行结果图

（3）指定main函数的参数

选择“Project/Settings/Debug”菜单命令，即可打开图6-16所示的对话框，在“Program arguments”文本框中输入main函数的参数arg1 arg2 arg3，编写程序在命令行面板重输出这三个参数。

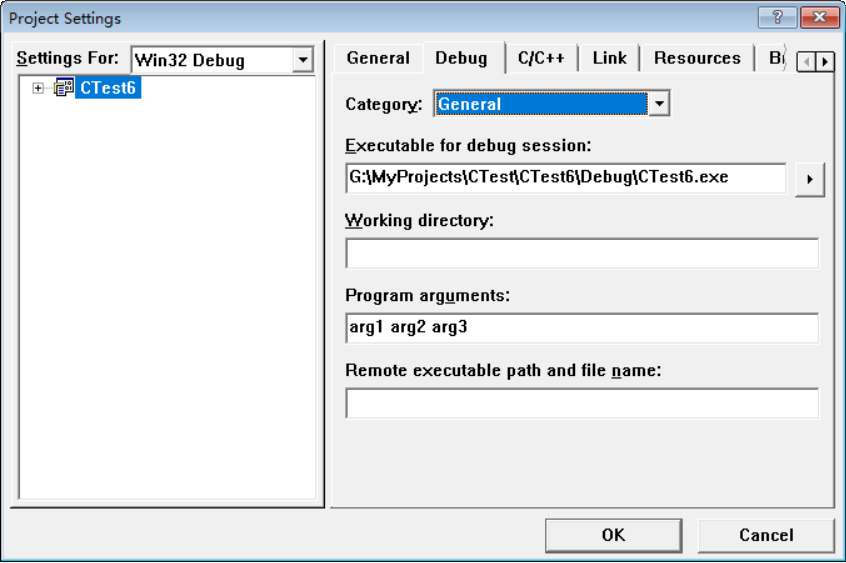


图6-16 输入main函数的参数

**解答：**

1）源程序清单

#include<stdio.h>

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

{

for(int i=1;i<argv;i++)

{

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

}

return 0;

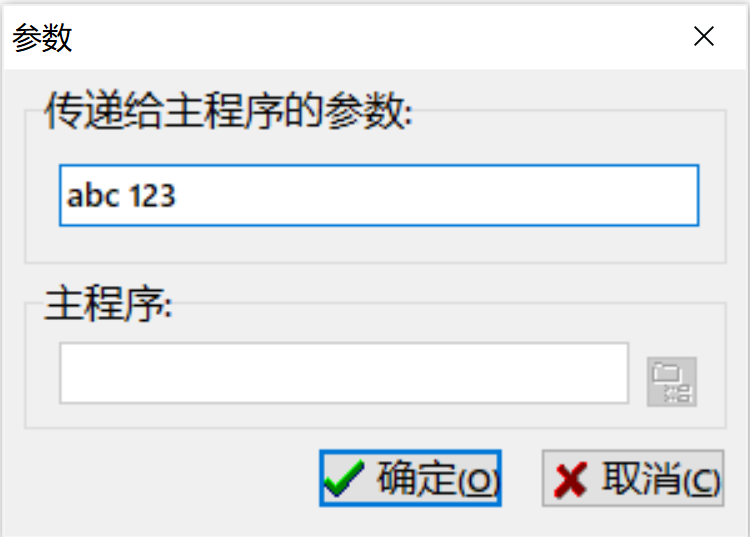
}

2）测试

  （a） 测试数据：

        abc 123

  （b） 对应测试数据的运行结果截图



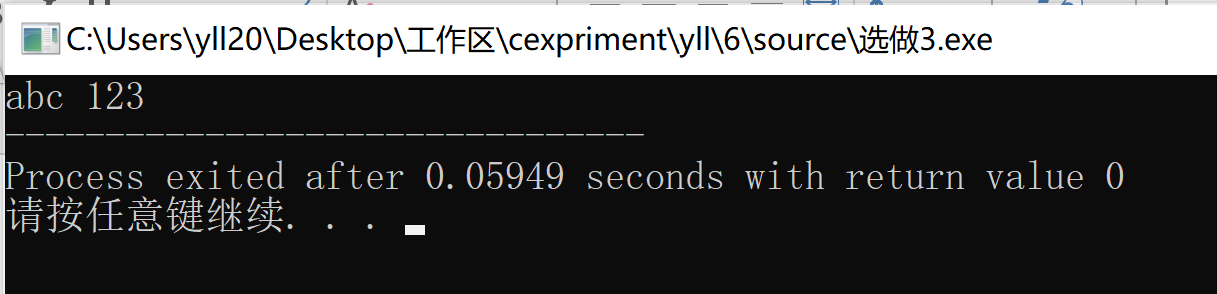


图6-17 选做题3的运行结果图

## 6.4 实验小结

实验中体会到了指针在访问数据上的方便性，同时也发现使用指针的时候一定要小心访问非法内存的问题。同时掌握了高精度计算将数字转化为字符存储于数组中的思想。