

华中科技大学

# 课程实验报告

课程名称: C 语言程序设计实验

专业班级: \_\_\_\_\_

学 号: \_\_\_\_\_

姓 名: \_\_\_\_\_

指导教师: \_\_\_\_\_

报告日期: 2019.01.01

# 目 录

1	表达式和标准输入与输出实验.....	1
1.1	实验目的.....	1
1.2	实验内容.....	1
1.3	实验小结.....	10
2	流程控制实验 .....	12
2.1	实验目的.....	12
2.2	实验内容.....	12
2.3	实验小结.....	26
3	函数与程序结构实验.....	27
3.1	实验目的.....	27
3.2	实验内容.....	27
3.3	实验小结.....	37
4	编译预处理实验 .....	39
4.1	实验目的.....	39
4.2	实验内容.....	39
4.3	实验小结.....	47
5	数组实验 .....	48
5.1	实验目的.....	48
5.2	实验内容.....	48
5.3	实验小结.....	64
6	指针实验 .....	66
6.1	实验目的.....	66
6.2	实验内容.....	66
6.3	实验小结.....	86
7	结构与联合实验 .....	87
7.1	实验目的.....	87
7.2	实验内容.....	87
7.3	实验小结.....	112
8	文件实验 .....	113
8.1	实验目的.....	113
8.2	实验内容.....	113
8.3	实验小结.....	118

# 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 3 行的 main 函数与之前的 void 之间缺少空格, 正确形式为:

```
void main(void)
```

3) 第 6 行缺少 newint 的声明, 正确格式为:

```
short p, k, newint;
```

4) 第 9, 10, 12 行 printf 函数内应用英文格式的双引号, 正确格式为:

```
9 printf("Input Fahrenheit: ")
```

```
10 scanf("%d",&f)
```

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

5) 第 10 行缺少&, 正确格式为:

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

6) 第 11 行 5/9 的小数部分被舍去, 正确格式为:

```
c=5.0/9*(f-32);
```

7) 第 15 行 r 的转换说明错误, 正确格式为:

```
scanf("%lf",&r);
```

8) 第 17 行 printf 函数中不用&, 正确格式为:

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

9) 第 20 行 k, p 的转换说明错误, 正确格式为:

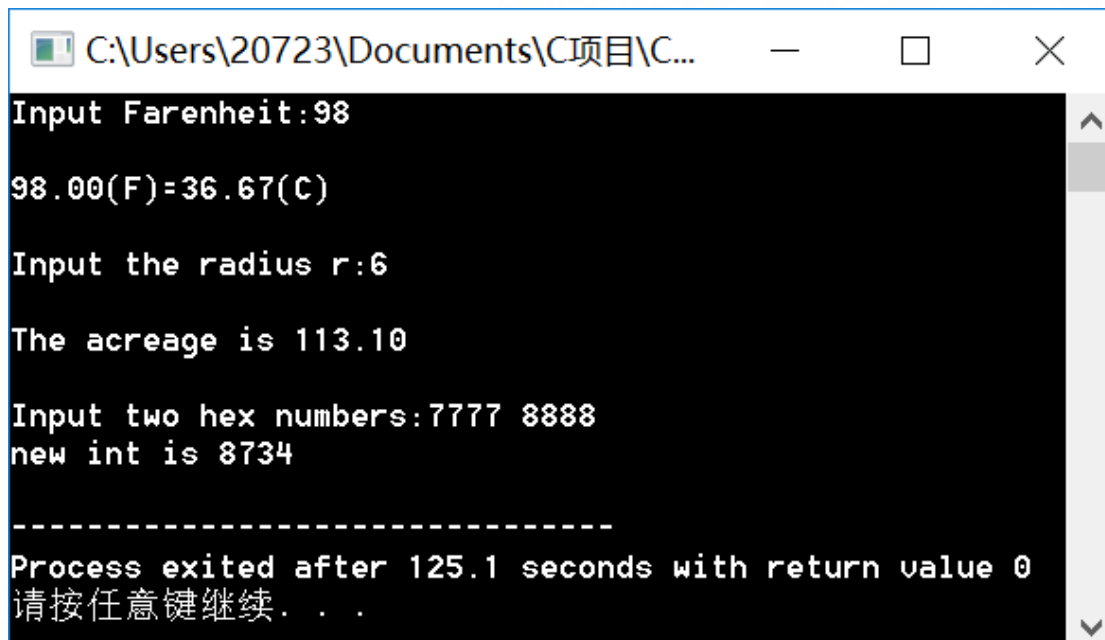
```
printf("%hu %hu",k,p);
```



10) 第 21 行应该用右移符，正确格式为：

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

(2) 错误修改后运行结果如图 1-1 所示。



```

C:\Users\20723\Documents\C项目\C...
Input Farenheit:98
98.00(F)=36.67(C)
Input the radius r:6
The acreage is 113.10
Input two hex numbers:7777 8888
newint is 8734
-----
Process exited after 125.1 seconds with return value 0
请按任意键继续...
    
```

图 1-1 程序改错题运行结果截图

### 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;
    printf("\na=%d,b=%d",a,b);
}
    
```

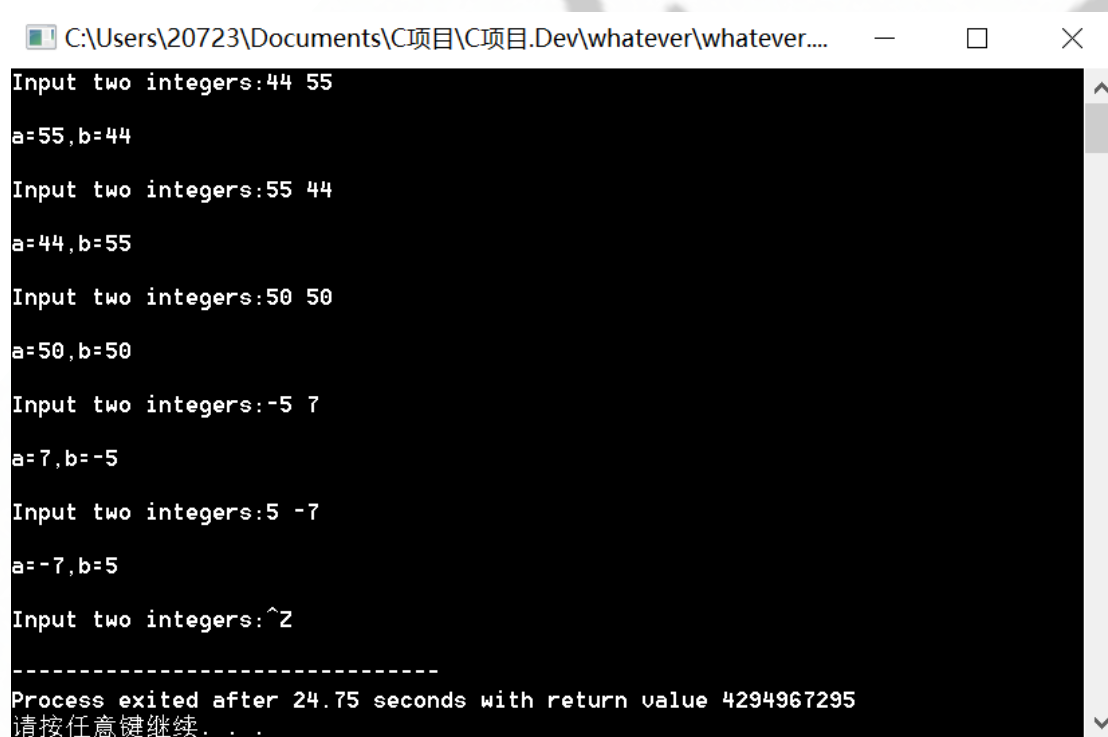
**解答：**

将上述语句替换为  $a=a^b$ ;  $b=a^b$ ;  $a=a^b$ ;，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);
}
```

修改后的程序运行结果如图 1-2 所示。



```
C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
Input two integers:44 55
a=55,b=44
Input two integers:55 44
a=44,b=55
Input two integers:50 50
a=50,b=50
Input two integers:-5 7
a=7,b=-5
Input two integers:5 -7
a=-7,b=5
Input two integers:^Z
-----
Process exited after 24.75 seconds with return value 4294967295
请按任意键继续. . .
```

图 1-2 源程序修改替换运行结果截图

### 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。

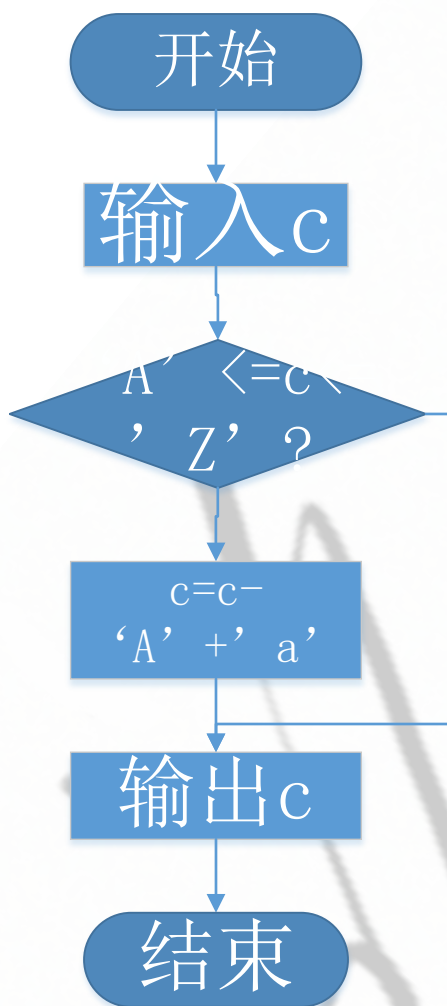


图 1-3 编程题 1 的程序流程图

## 2) 源程序清单

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    /*for task (1)*/
    char c;
    c=getchar();
    if(c>='A'&&c<='Z')c=c-'A'+'a';
    printf("%c\n",c);
    return 0;
}
    
```

### 3) 测试

#### (a) 测试数据:

D (大写字母)

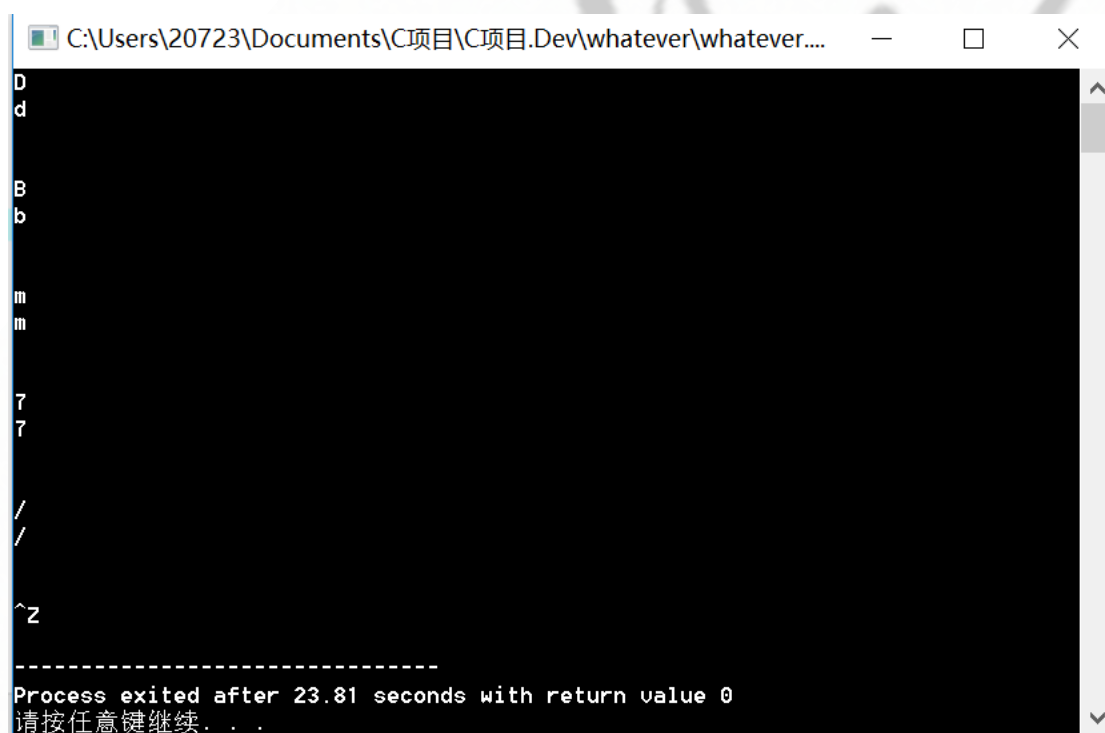
B

m (小写字母)

7 (数字)

/ (符号)

#### (b) 对应测试数据的运行结果如图 1-4 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
D
d
B
b
m
m
7
7
/
/
^Z
-----
Process exited after 23.81 seconds with return value 0
请按任意键继续. . .
    
```

图 1-4 程序设计题 1 运行结果截图

(2) 编写一个程序，输入无符号短整数  $x$ ,  $m$ ,  $n$  ( $0 \leq m \leq 15$ ,  $1 \leq n \leq 16-m$ ), 取出  $x$  从第  $m$  位开始向左的  $n$  位 ( $m$  从右至左编号为  $0 \sim 15$ ), 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  $0 \leq m \leq 15$ ,  $1 \leq n \leq 16-m$ , 转 2.1, 否则转 3.

2.1 首先  $x \gg m$ , 将要处理的  $n$  位移动到最右;



2.2 再将上一步的结果左移 16-n 位，即； $x \ll (16-n)$

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>
#include <stdlib.h>
int main()
{
    /*for task (2)*/
    unsigned short x,m,n;
    printf("Input three integrals(m between 0 and 15, n between 1 and 16-m):");
    scanf("%hx %hu %hu",&x,&m,&n);
    if((m>=0)&&(m<=15)&&(n>0)&&(n<=(16-m))) {
        x=x>>m<<(16-n);    //将 x 向左移 m 位后再向右移 16-n 位（即向左对齐）
        printf("%hx\n",x);
    }
    else printf("Error");
    return 0;
}
```

3) 测试

(a) 测试数据：

叙述选择测试数据的方法 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	0001 0001 0001 0001 (1111)	8	8	计算结果 0001 0001 0000 0000 即 1100
用例 2	1101 0101 1000 0011 (aaaa)	16	1	Error (m 值超范围)
用例 3	1010 1011 1100 1101 (abcd)	8	9	Error (n 值超范围)

(b) 对应测试用例 1 的运行结果如图 1-5 所示。

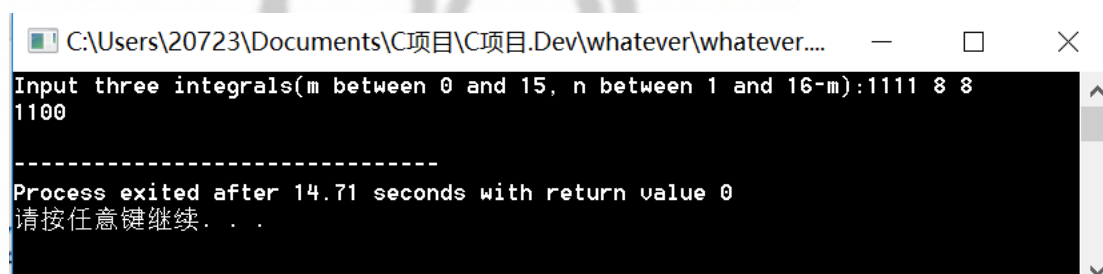
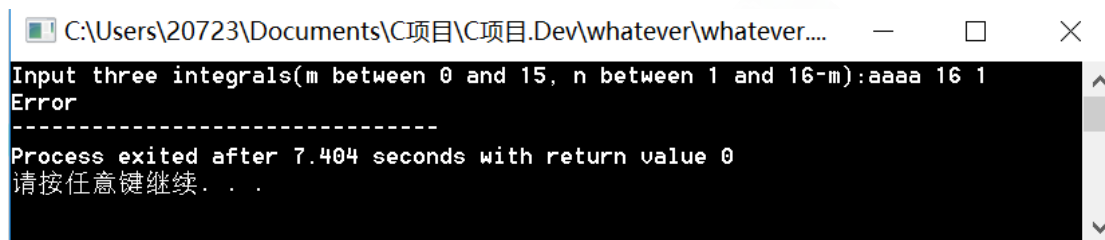


图 1-5 编程题 3 的测试用例一的运行结果

对应测试用例 2 的运行结果如图 1-6 所示。

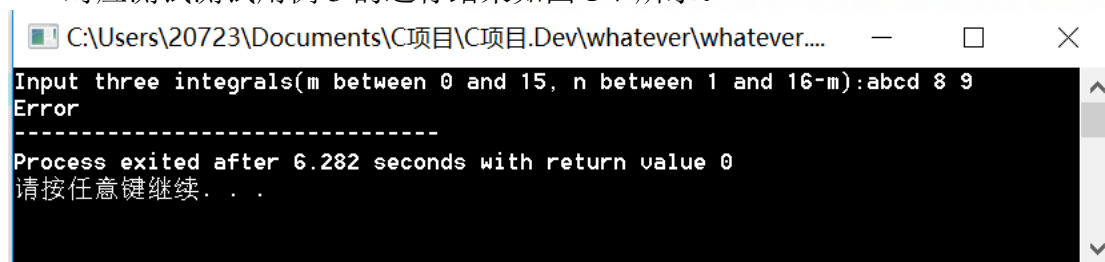


```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
Input three integrals(m between 0 and 15, n between 1 and 16-m):aaaa 16 1
Error
-----
Process exited after 7.404 seconds with return value 0
请按任意键继续. . .
    
```

图 1-6 编程题 3 的测试用例二的运行结果

对应测试用例 3 的运行结果如图 1-7 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
Input three integrals(m between 0 and 15, n between 1 and 16-m):abcd 8 9
Error
-----
Process exited after 6.282 seconds with return value 0
请按任意键继续. . .
    
```

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。

(3) IP 地址通常是 4 个用句点分隔的小整数，如 32.55.1.102。这些地址在机器中用无符号长整形表示。编写一个程序，以机器存储的形式读入一个 32 位的互联网 IP 地址，对其译码，然后用常见的句点分隔的 4 部分的形式输出。

解答：

- 1) 算法流程如图 1-8 所示。

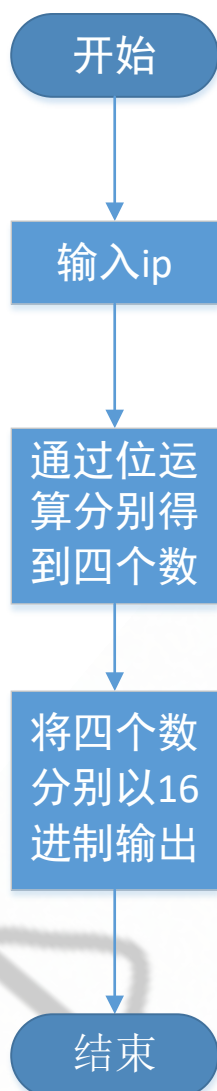


图 1-8 编程题 3 的程序流程图

## 2) 源程序清单

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    /*for task (3)*/
    unsigned int ip,_1,_2,_3,_4;
    scanf("%u",&ip);
    _1=ip>>24;
    _2=ip<<8>>24;
    _3=ip<<16>>24;
    _4=ip<<24>>24;
    printf("%hu.%hu.%hu.%hu\n",_1,_2,_3,_4);

    return 0;
}
    
```

## 3) 测试

(a) 测试数据:

4294967295 (理论结果 255.255.255.255)

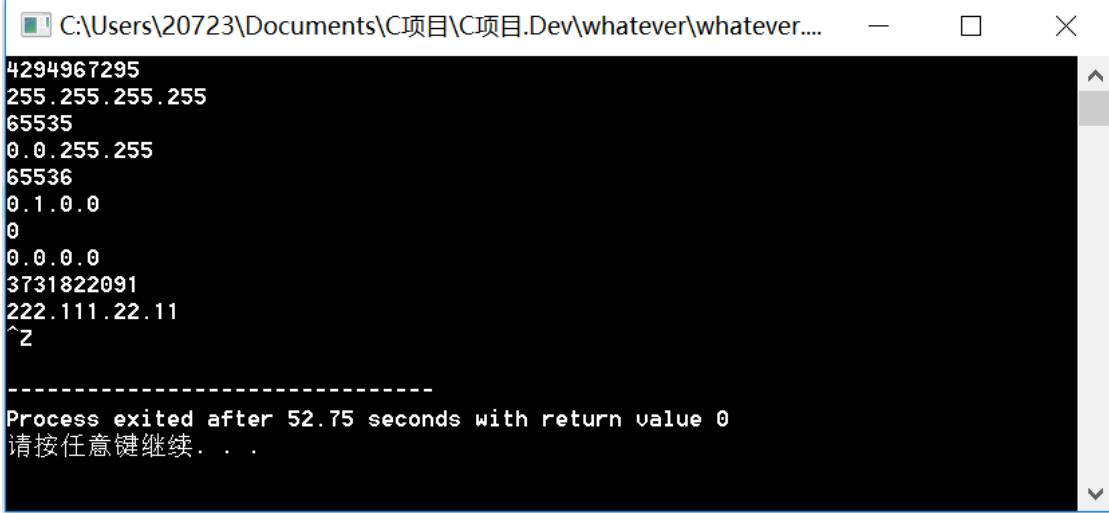
65535 (理论结果 0.0.1.1)

65536 (理论结果 0.1.0.0)

0 (理论结果 0.0.0.0)

3731822091 (理论结果 222.111.22.11)

(b) 对应测试数据的运行结果如图 1-9 所示



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
4294967295
255.255.255.255
65535
0.0.255.255
65536
0.1.0.0
0
0.0.0.0
3731822091
222.111.22.11
^Z
-----
Process exited after 52.75 seconds with return value 0
请按任意键继续. . .
    
```

图 1-9 程序设计题 3 运行结果截图

### 1.3 实验小结

1. 改错题中有一些引号不是英文格式, 刚开始没发现, 后来仔细观察许久, 才找到了这个错误。以后编程的过程中打引号时需要格外注意。
2. 如果在编译预处理处定义常量时多加了分号, 编译报错将会出现在使用了相关常量的表达式处。一方面, 用 `define` 定义常量时千万不要多加分号, 另一方面, 如果编译显示某表达式有错时可以检查一下常量定义。
3. 改错题中华氏温度和摄氏温度转换的表达式那里的  $5/9$  的计算结果是 0, 导致输出永远是 0。我们要牢记 C 语言中整数和整数的运算结果永远是整数, 这点对除法运算的影响最大, 所以我们在调用除法运算时要注意操作数是否为整型。
4. 改错题有一处 `scanf` 中缺少地址符 `&`, 编译不会报错, 但输入函数不能正常运行, 程序也会崩溃, 检查时也难以找到问题。所以使用 `scanf` 函数时要记得加上取地址符。

5. 程序设计题中有一处定义了 unsigned short 变量，但在调用输入函数时错误的使用了 %u 的转换说明，导致输入时只有第一个数据是正确的，后经过老师的提醒发现应用 %hu 的转换说明。这告诉我每每进行转换说明时都要仔细想想它到底是哪种类型的变量，一定要使用正确的转换说明。





## 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 Turbo C 2.0 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

#### 1.2.1 源程序改错

下面是计算  $s=n!$  的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上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。例如， $8!=40320$ 。

```
1  #include <stdio.h>
2  void main(void)
3  {
4      int i,n,s=1;
5      printf("Please enter n:");
6      scanf("%d",n);
7      for(i=1,i<=n,i++)
8          s=s*i;
9      printf("%d! = %d",n,s);
10 }
```

解答：

(1) 错误修改：

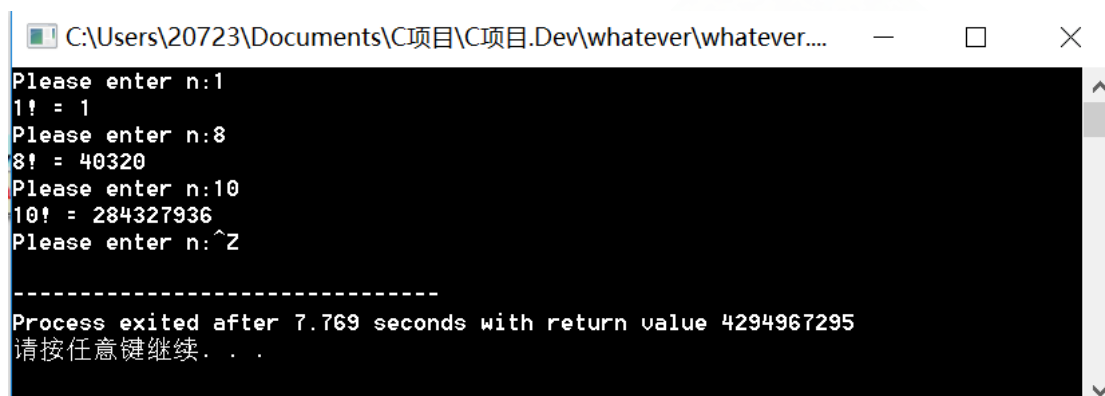
1) 第 6 行 scanf 函数中缺少地址符&，正确形式为：

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

2) for 循环中本应使用分号但错误的使用了逗号，正确形式为：

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

(2) 错误修改后运行结果如图 2-1 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
Please enter n:1
1! = 1
Please enter n:8
8! = 40320
Please enter n:10
10! = 284327936
Please enter n:^Z
-----
Process exited after 7.769 seconds with return value 4294967295
请按任意键继续...
    
```

图 2-1 源程序改错运行结果截图

### 1.2.2 源程序修改替换

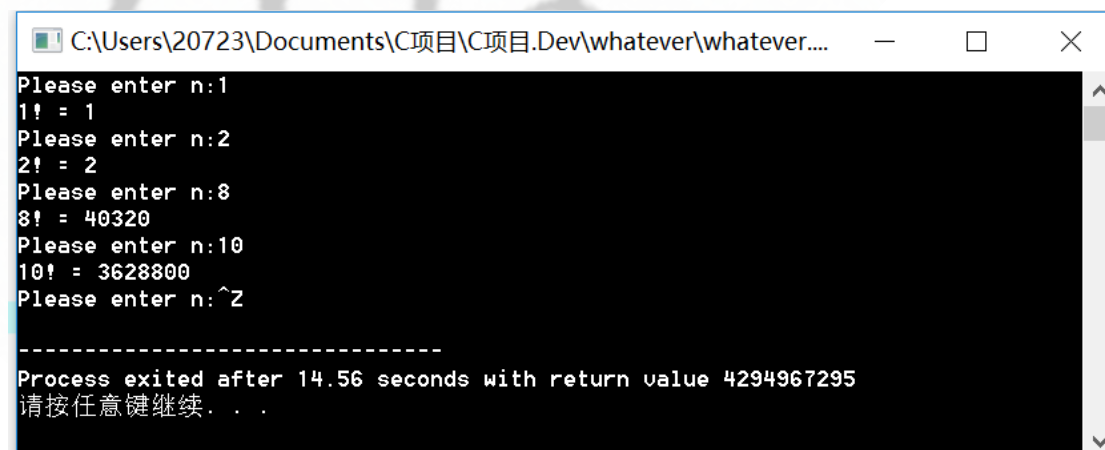
(1) 修改第 1 题，分别用 while 和 do-while 语句替换 for 语句。

解答：

```

#include <stdio.h>
#include <stdlib.h>
void main(void) {
    int i=1,n,s=1;
    printf("Please enter n:");
    scanf("%d",&n);
    while(i<=n){
        s=s*i;
        i++;
    }
    printf("%d! = %d",n,s);
}
    
```

用 while 语句替换 for 语句后的程序运行结果如图 2-2 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
Please enter n:1
1! = 1
Please enter n:2
2! = 2
Please enter n:8
8! = 40320
Please enter n:10
10! = 3628800
Please enter n:^Z
-----
Process exited after 14.56 seconds with return value 4294967295
请按任意键继续...
    
```

图 2-2 用 while 语句替换 for 语句后运行结果截图

```

#include <stdio.h>
    
```

```
#include <stdlib.h>
void main(void) {
    int i=0,n,s=1;
    printf("Please enter n:");
    scanf("%d",&n);
    do{
        i++;
        s=s*i;
    }while(i<n);
    printf("%d! = %d",n,s);
}
```

用 do-while 语句替换 for 语句后程序运行结果如图 2-3 所示。



```
Please enter n:10
10! = 3628800
-----
Process exited after 3.663 seconds with return value 13
请按任意键继续...
```

图 2-3 用 do-while 语句替换 for 语句运行结果截图

(2)修改第 1 题,输入改为“整数 S”,输出改为“满足  $n! \geq S$  的最小整数 n”。

例如输入整数 40310, 输出结果为 n=8。

**解答:**

```
#include <stdio.h>
#include <stdlib.h>

void main(void) {
    int S,n=1,t=1;
    scanf("%d",&S);
    while(t<S){
        n++;
        t=t*n;
    }
    printf("%d",n);
}
```

替换后的程序运行结果如图 2-4 所示。

```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever...
Please enter n:2
2! = 2
Please enter n:4
4! = 24
Please enter n:8
8! = 40320
Please enter n:12
12! = 479001600
Please enter n:^Z

-----
Process exited after 15.51 seconds with return value 4294967295
请按任意键继续. . .
    
```

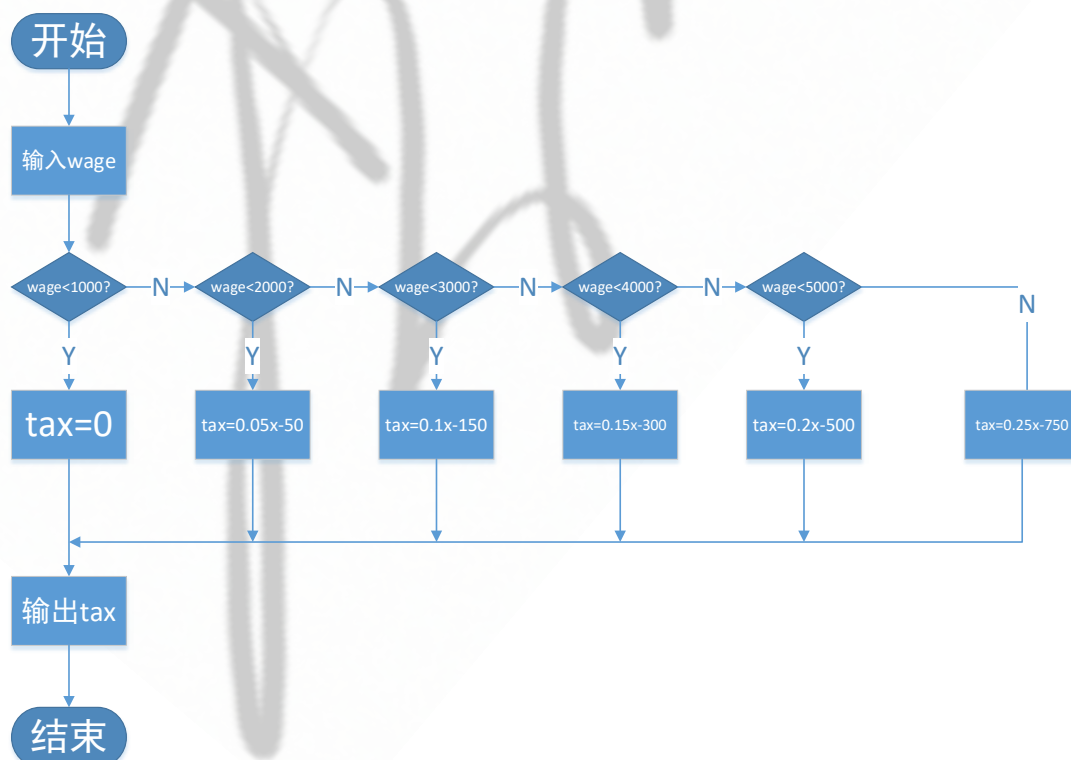
图 2-4 替换后的程序运行结果截图

### 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； $1000 \leq x < 2000$ ，收取 5% 的税金； $2000 \leq x < 3000$ ，收取 10% 的税金； $3000 \leq x < 4000$ ，收取 15% 的税金； $4000 \leq x < 5000$ ，收取 20% 的税金； $x \geq 5000$ ，收取 25% 的税金。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。



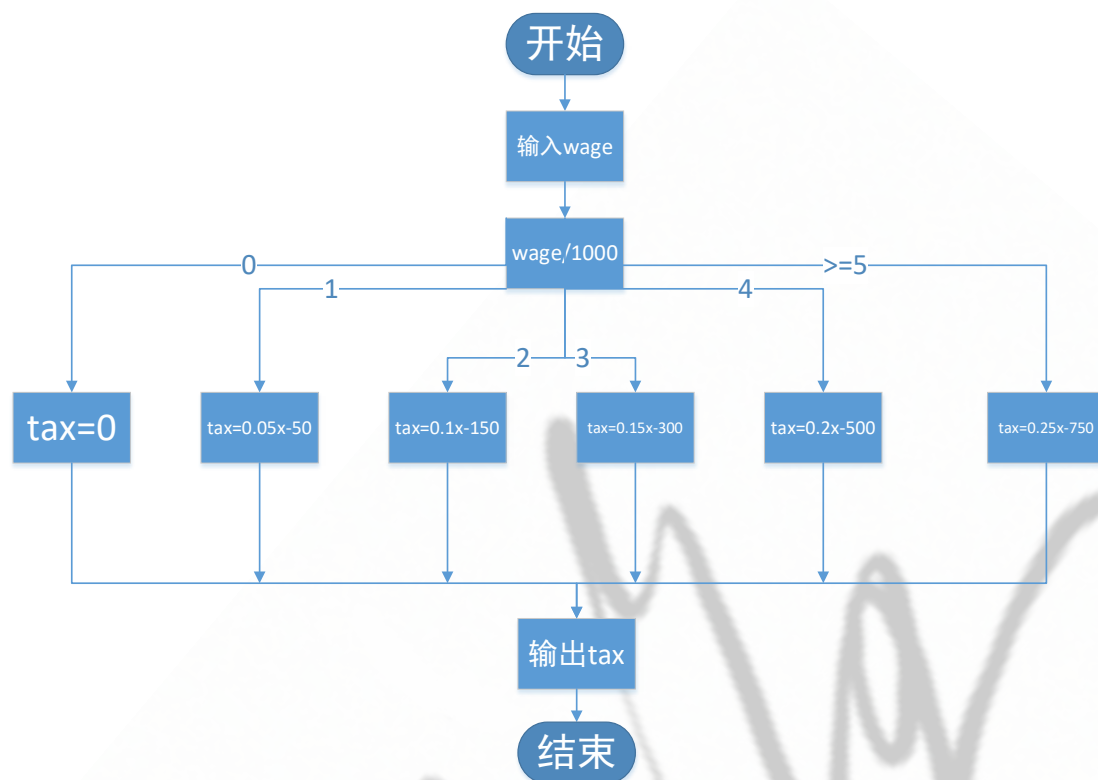


图 2-5 程序设计题 1 算法示意图

## 2) 源程序清单

```

#include <stdio.h>
#include <stdlib.h>
int main(void){
    double wage,tax;
    scanf("%lf",&wage);
    if(wage<1000)tax=0;
    else if(wage<2000)tax=0.05*(wage-1000);
    else if(wage<3000)tax=0.1*(wage-2000)+50;
    else if(wage<4000)tax=0.15*(wage-3000)+150;
    else if(wage<5000)tax=0.2*(wage-4000)+300;
    else tax=0.25*(wage-5000)+500;
    printf("%.2lf(if-else)\n",tax);
    switch((int)wage/1000){
        case 0:tax=0;break;
        case 1:tax=0.05*(wage-1000);break;
        case 2:tax=0.1*(wage-2000)+50;break;
        case 3:tax=0.15*(wage-3000)+150;break;
        case 4:tax=0.2*(wage-4000)+300;break;
        default :tax=0.25*(wage-5000)+500;break;
    }
    printf("%.2lf(switch)\n",tax);
    return 0;
}
    
```



}

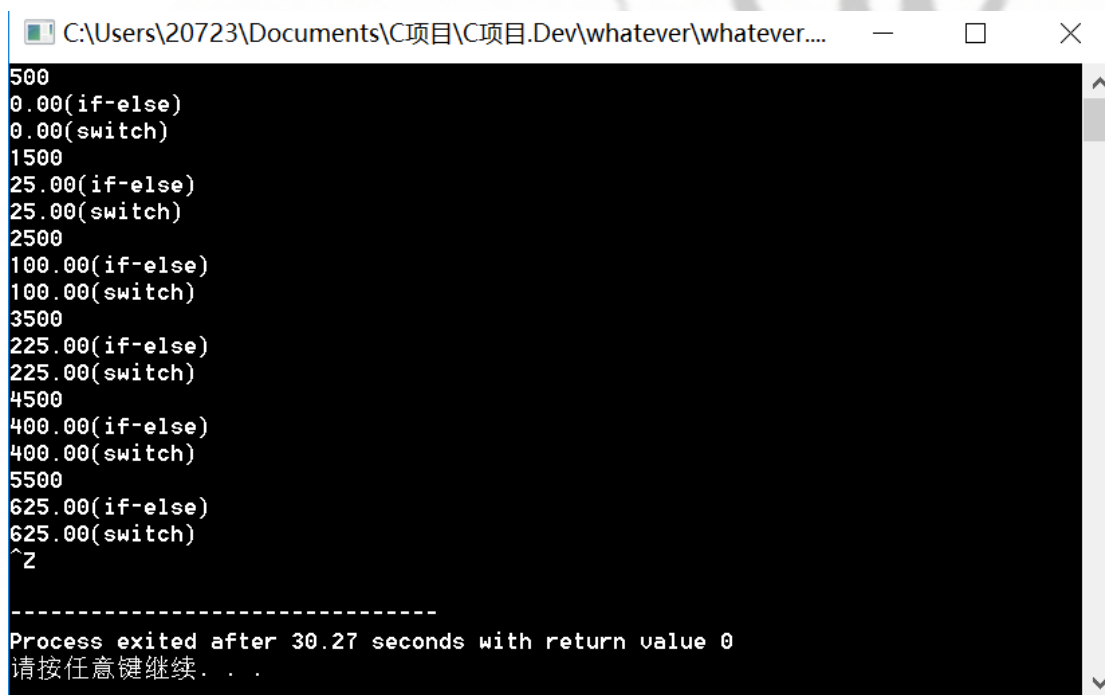
### 3) 测试

(a) 测试数据:

500  
1500  
2500  
3500  
4500  
5500

在各范围区间分别选取一个值进行测试

(b) 对应测试数据的运行结果截图如图 2-6 所示



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
500
0.00(if-else)
0.00(switch)
1500
25.00(if-else)
25.00(switch)
2500
100.00(if-else)
100.00(switch)
3500
225.00(if-else)
225.00(switch)
4500
400.00(if-else)
400.00(switch)
5500
625.00(if-else)
625.00(switch)
^Z

-----
Process exited after 30.27 seconds with return value 0
请按任意键继续. . .
    
```

图 2-6 程序设计题 1 的运行结果截图

(2) 编写一个程序, 将输入的一行字符复制到输出, 复制过程中将一个以上的空格字符用一个空格代替。

**解答:**

1) 算法流程如图 2-7 所示。

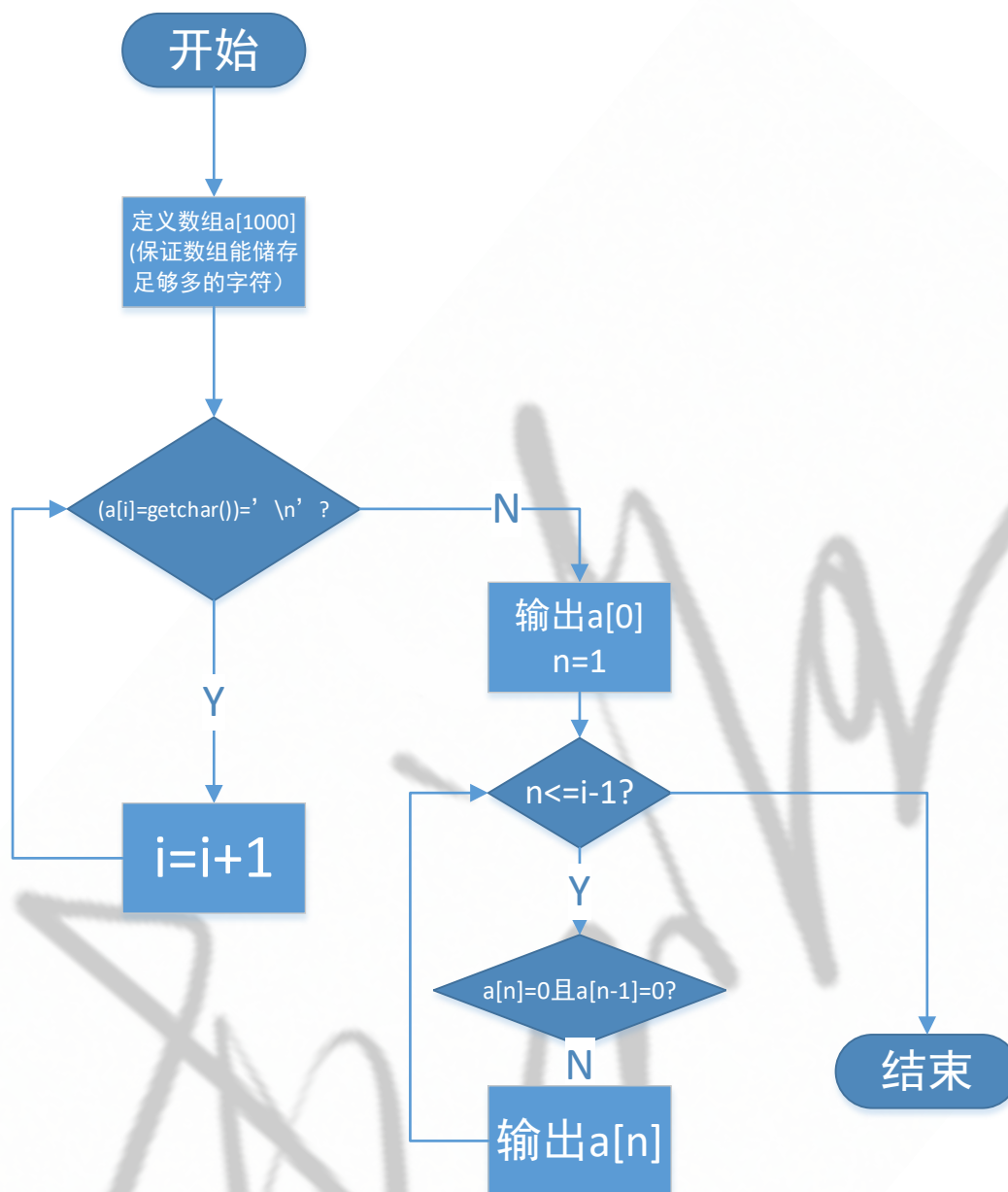


图 2-7 程序设计题 2 算法示意图

## 2) 源程序清单

```

#include <stdio.h>
#include <stdlib.h>
int main(void){
    char c[1000];
    int i=0,cnt;
    while((c[i]=getchar())!='\n'){
        i=i+1;
    }
    i--;
    putchar(c[0]);
    for(cnt=1;cnt<=i;cnt++){

```

```

        if(c[cnt]!=' '&& c[cnt-1]!=' ');
        else putchar(c[cnt]);
    }

    return 0;
}

```

### 3) 测试

(a) 测试数据:

123	456	789	10000 (数字)
Abc	def	ghi	jklmn (字母)
~!@	#\$%	^&*	()_+= (符号)
-[]	{ }	\ ; ' , < > ? /	(符号)

(同时检测字母和数据, 有空格数量为 1, 有多于 1, 保证正确运行)

(b) 对应测试数据的运行结果如图 2-8 所示。

```

C:\Users\20723\Documents\C项目\C项目.Dev\C语言作业\合并空格...
123 456 789 10000
12345678910000
abc def ghi jklmn
abcdefghijklmn
~!@ #$% ^&* ()_+=
~!@#$%^&*()_+=
-[] {} | \ ; ' , < > ? /
-[]{}|\;'\,<>?/
-----
Process exited after 105.9 seconds with return value 0
请按任意键继续...

```

图 2-8 程序设计题 2 运行结果截图

(3) 编写一个程序, 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

每个数据值可以由组合  $C_i^j$  计算（表示第  $i$  行第  $j$  列位置的值），而  $C_i^j$  的计算如下：

$$C_i^0 = 1 \quad (i=0, 1, 2, \dots)$$

$$C_i^j = C_i^{j-1} * (i - j + 1) / j \quad (j=0, 1, 2, 3, \dots, i)$$

本程序中为了打印出金字塔效果，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，程序编制过程中要注意区分。

**解答：**

1) 算法流程如图 2-9 所示。

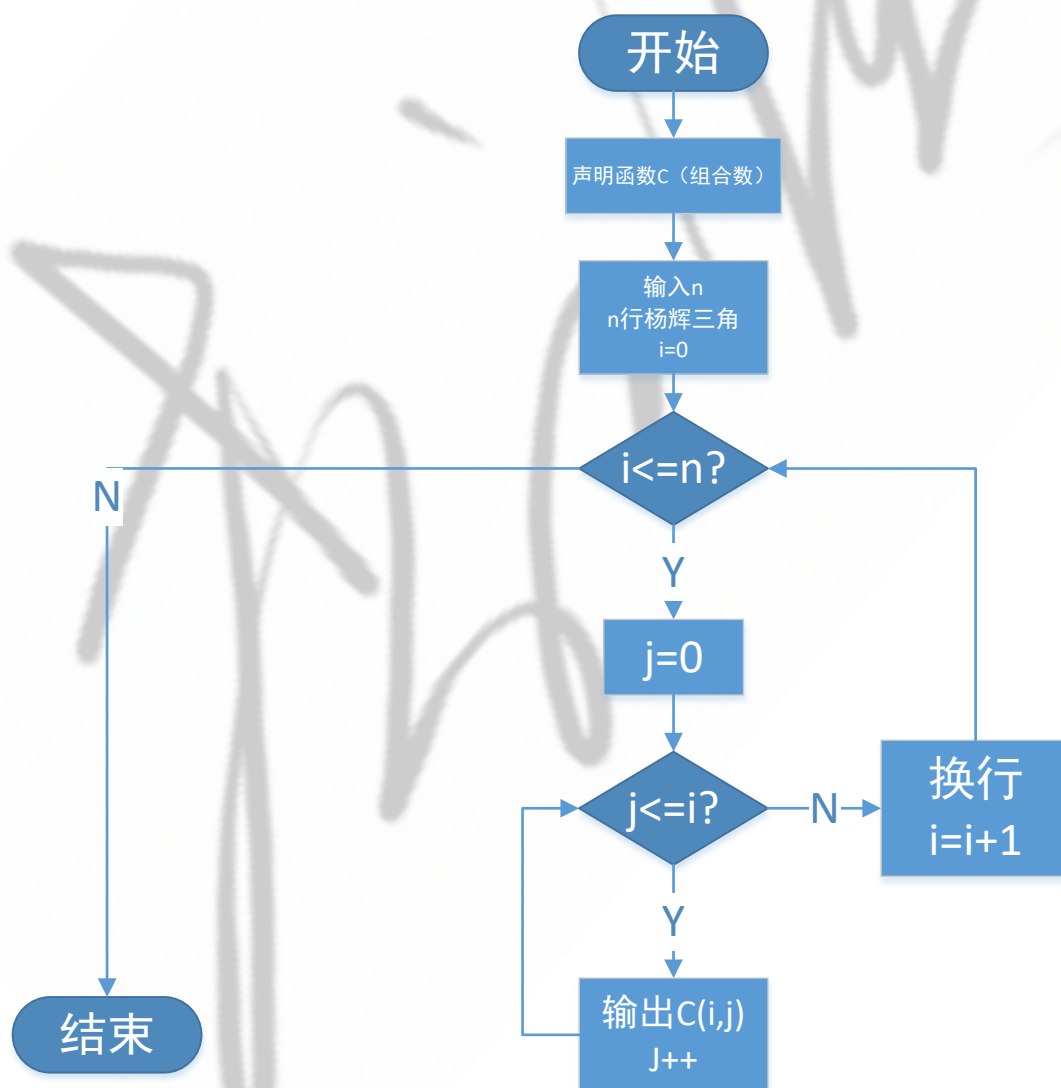


图 2-9 程序设计题 3 算法示意图

## 2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>
int combination(int ,int )
int main(void){
    int n,i,j,icnt,jcnt,space;
    scanf("%d",&n);
    for(i=0;i<=n;i++){
        for(space=2*n-2*i;space>0;space--)printf(" ");
        for(j=0;j<=i;j++)printf("%-4d",combination(i,j));
        printf("\n");
    }
    return 0;
}
int combination(int i,int j){
    int comb[j+1],n;
    comb[0]=1;
    for(n=1;n<=j;n++){
        comb[n]=comb[n-1]*(i-n+1)/n;
    }
    return comb[j];
}
```

## 3) 测试

### (a) 测试数据:

12(在当前数据类型的限制下选取较大的数据,保证小于这个数的测试数据都能正确运行)

### (b) 对应测试数据的运行结果如图 2-10 所示。



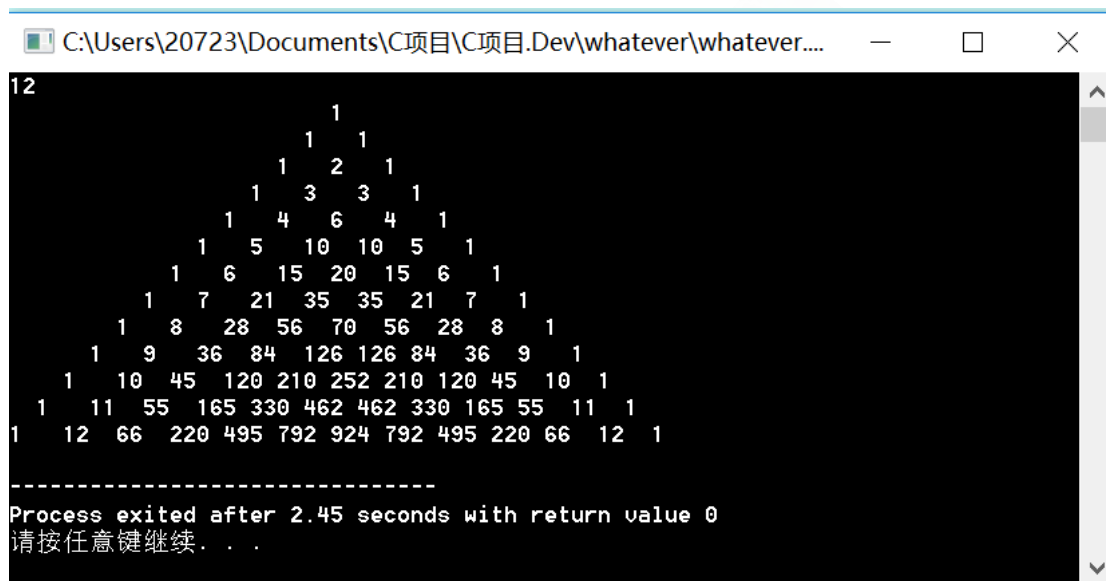


图 2-10 程序设计题 3 运行结果截图

(4)编写一个程序,将用户输入的任意正整数逆转,例如,输入 1234,输出 4321。

解答:

1) 算法流程如图 2-11 所示。

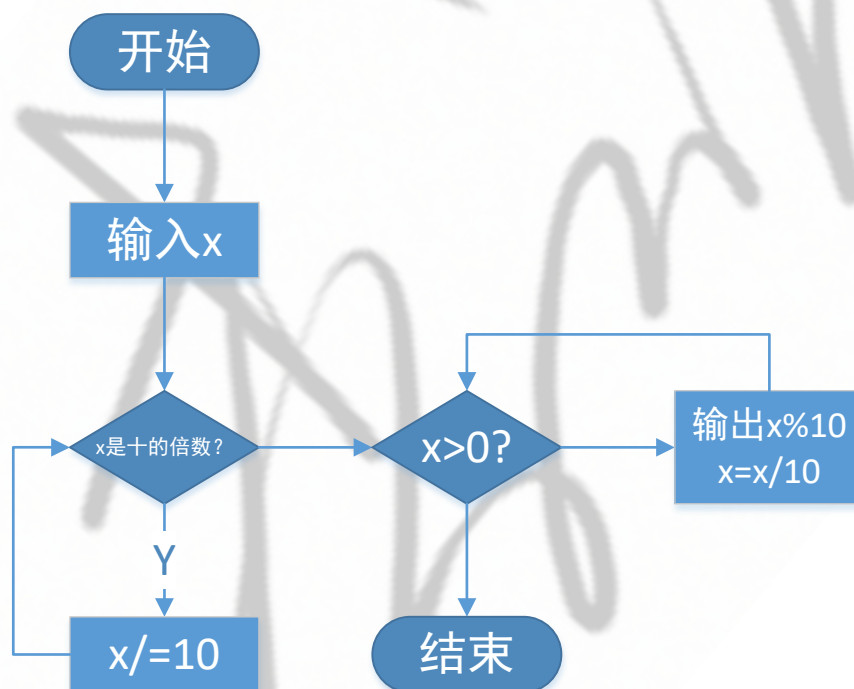


图 2-11 程序设计题 4 算法示意图

2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
int main(void) {
```

```

int x;
scanf("%d",&x);
while(x%10==0){
    x/=10;
}
while(x>0){
    printf("%d",x%10);
    x/=10;
}
printf("\n");
return 0;
}

```

### 3) 测试

(a) 测试数据:

9 (一位数)

10000 (末尾多位 0)

00001 (前导多位 0)

12345 (无 0)

12045 (中间有 0)

10009 (中间多位 0)

(b) 对应测试数据的运行结果如图 2-12 所示。

```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
9
9
10000
1
00001
1
12345
54321
12045
54021
10009
90001
^Z
-----
Process exited after 30.77 seconds with return value 0
请按任意键继续. . .

```

图 2-12 程序设计题 4 运行结果截图

### 2.2.4 选做题

编写并上机调试运行能实现以下功能的程序。

编写一个程序，用牛顿迭代法求方程  $f(x)=3x^3-4x^2-5x+13=0$  满足精度  $e=10^{-6}$  的一个近似根，并在屏幕上输出所求近似根。

牛顿迭代法求方程近似根的迭代公式为：

$$\begin{cases} x_0 = a \\ x_{k+1} = x_k - f(x_k) / f'(x_k) \end{cases},$$

其中， $f'(x)$  是函数  $f(x)$  的导函数。牛顿迭代法首先任意设定的一个实数  $a$  来作为近似根的迭代初值  $x_0$ ，然后用迭代公式计算下一个近似根  $x_1$ 。如此继续迭代计算  $x_2, x_3, \dots, x_n$ ，直到  $|x_n - x_{n-1}| \leq \text{精度 } e$ ，此时值  $x_n$  即为所求的近似根。

解答：

1) 算法流程如图 2-13 所示。

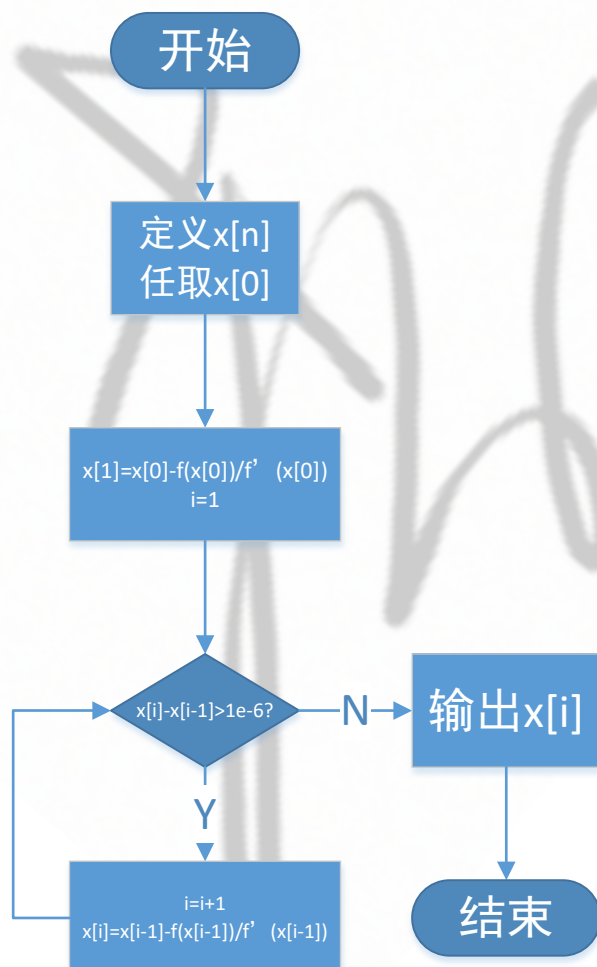


图 2-13 选做题算法示意图

## 2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double f(double);
double fdao(double);
int main(void) {
    double x[10000];
    x[1]=x[0]-f(x[0])/fdao(x[0]);
    int i=1;
    while(fabs(x[i]-x[i-1])>1e-6){
        i++;
        x[i]=x[i-1]-f(x[i-1])/fdao(x[i-1]);
    }
    printf("%lf\n",x[i]);
    return 0;
}
double f(double x){
    double f;
    f=((3*x-4)*x-5)*x+13;
    return f;
}
double fdao(double x){
    double fdao;
    fdao=(9*x-8)*x-5;
    return fdao;
}
```

## 3) 测试

(a) 测试数据:

本题无

(b) 对应测试数据的运行结果如图 2-14 所示。

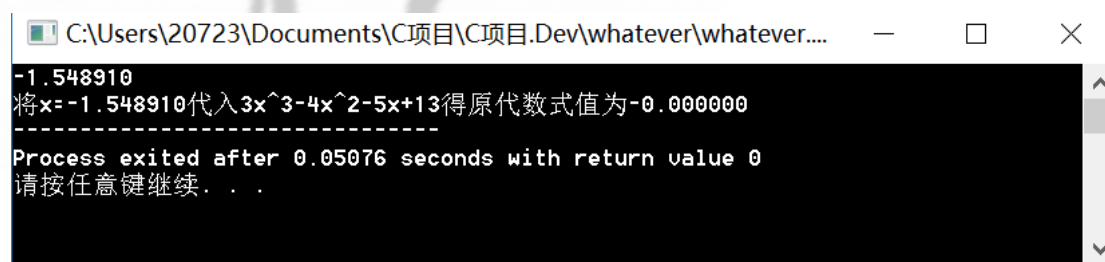


图 2-14 选做题运行结果截图

## 2.3 实验小结

1. 完成“打印杨辉三角”过程中，在迭代计算组合数混淆了变量  $j$  与循环变量  $n$ ，导致了错误的输出。故在写循环的过程中应该分清循环变量和循环条件变量，尤其是在迭代循环中。

2. 完成“输出整数逆序”过程中，考虑不周全，没有考虑到整数末尾有若干个 0 的情况。发现问题时也没有能够立即想到解决问题的办法，原因是一直在思考怎么修改循环内的语句，后发现，在对整数做处理前去掉整数末尾的 0 就能轻松解决问题。

3. 完成“牛顿迭代求方程解”过程中，在计算函数值和导函数值的表达式中沿用了数学中的书写习惯，即因子之间省去了乘号 $*$ ，导致错误的计算结果。一开始还怀疑是使用秦九韶算法时的错误，将函数计算的表达式换回了原始表达式，突然想起 C 语言中没有计算幂的运算符，因而高次项得一一展开，这才发现了问题。故在书写表达式的过程中，要牢记 C 语言的语法规则。



## 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方法；以及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) 熟悉多文件编译技术。

### 3.2 实验内容

#### 3.2.1. 源程序改错题

下面是计算  $s=1!+2!+3!+\dots+n!$  的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上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。

```
1  #include "stdio.h"
2  void main(void)
3  {
4      int k;
5      for(k=1;k<6;k++)
6          printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));//使用前缺少声明
7  }
8  long sum_fac(int n)
9  {
10     long s=0;
11     int i;
12     long 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 行调用 sum\_fac 函数时未进行声明或定义，正确形式为：第 2 行  
前：long sum\_fac(int );

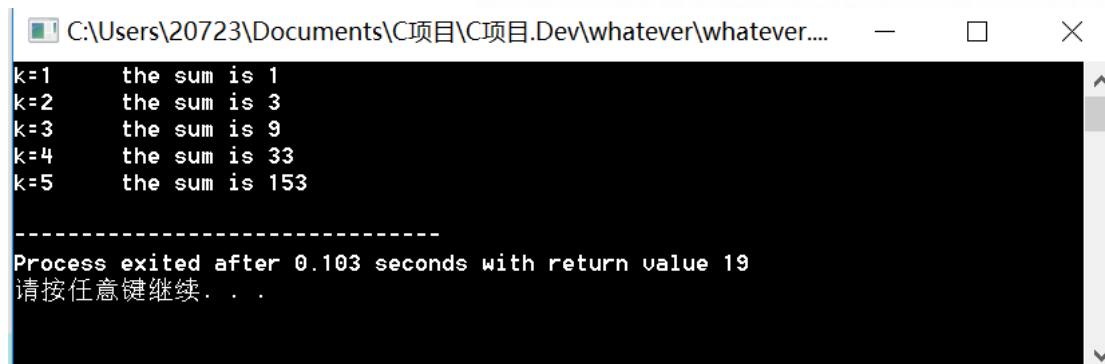
2) 第 10 行定义的 `s` 缺少累加器功能，正确形式为：

```
static long s;
```

3) 第 12 行定义的 `fac` 未初始化，正确形式为：

```
long fac=1;
```

(2) 错误修改后运行结果如图 3-1 所示。



```
C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
k=1    the sum is 1
k=2    the sum is 3
k=3    the sum is 9
k=4    the sum is 33
k=5    the sum is 153

-----
Process exited after 0.103 seconds with return value 19
请按任意键继续...
```

图 3-1 错误修改题程序运行结果截图

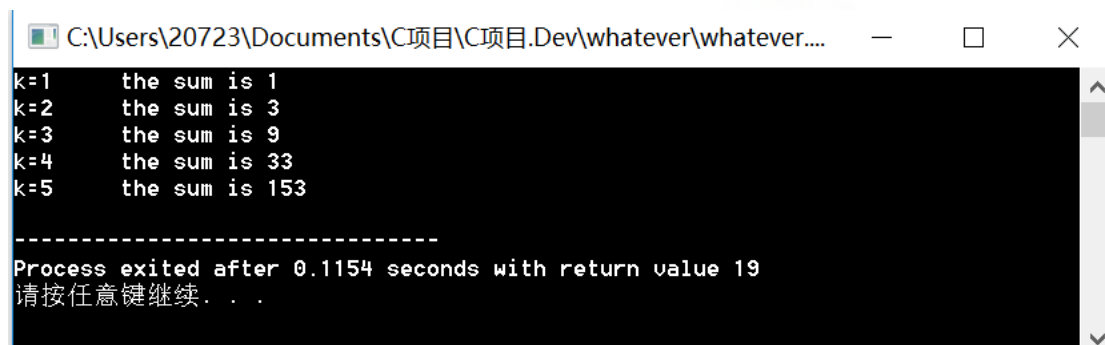
### 3.3.2. 源程序修改替换题

(1) 修改第 1 题中 `sum_fac` 函数，使其计算量最小。

解答：

```
#include "stdio.h"
long sum_fac(int);
void main(void)
{
    int k;
    for(k=1;k<6;k++)
        printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
}
long sum_fac(int n)
{
    static unsigned long s=0;
    int i;
    static long fac=1;
    fac*=n;
    s+=fac;
    return s;
}
```

修改后的程序运行结果如图 3-2 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
k=1    the sum is 1
k=2    the sum is 3
k=3    the sum is 9
k=4    the sum is 33
k=5    the sum is 153

-----
Process exited after 0.1154 seconds with return value 19
请按任意键继续. . .

```

图 3-2 源程序修改替换题（1）程序运行结果截图

(2) 修改第 1 题中 sum\_fac 函数，计算  $s = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$ 。

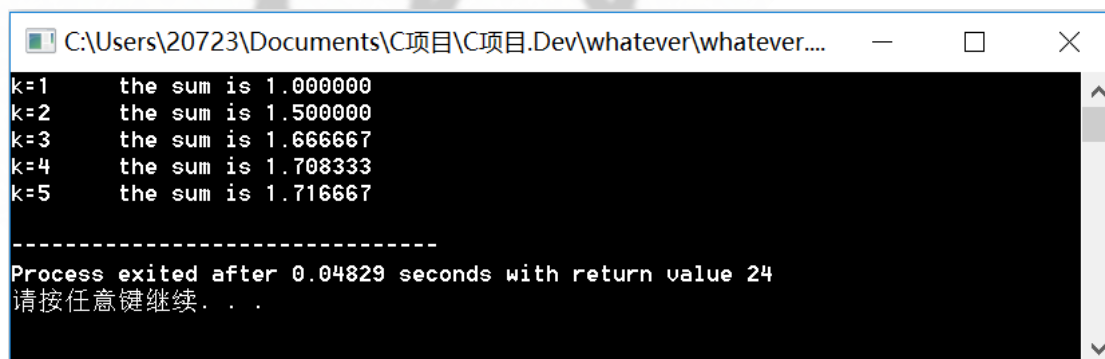
解答：

```

#include "stdio.h"
double sum_fac(int);
void main(void)
{
    int k;
    for(k=1;k<6;k++)
        printf("k=%d\tthe sum is %lf\n",k,sum_fac(k));
}
double sum_fac(int n)
{
    static double s=0;
    int i;
    static long fac=1;
    fac*=n;
    s+=1.0/fac;
    return s;
}

```

修改后的程序运行结果如图 3-3 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
k=1    the sum is 1.000000
k=2    the sum is 1.500000
k=3    the sum is 1.666667
k=4    the sum is 1.708333
k=5    the sum is 1.716667

-----
Process exited after 0.04829 seconds with return value 24
请按任意键继续. . .

```

图 3-3 源程序修改替换题（2）程序运行结果截图

### 3.2.3. 跟踪调试题

下面是计算 fibonacci 数列前 n 项和的源程序，现要求单步执行该程序，观察 p, i, sum, n 值，即：

(1) 刚执行完 `scanf("%d",&k);` 语句，p, i 值是多少？

p=0x22fe38(sum 的地址) i=0

(2) 从 fibonacci 函数返回后光条停留在哪个语句上？

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

(3) 进入 fibonacci 函数时，watch 窗口显示的是什么？

进入 fibonacci 函数后 watch 窗口显示如图 3-4 所示。

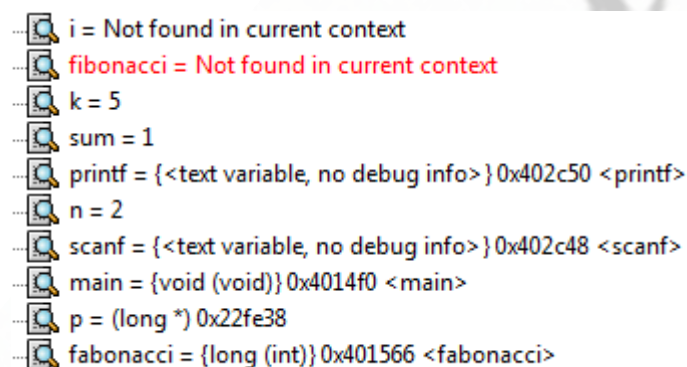


图 3-4 进入 fibonacci 函数后 watch 窗口显示截图

(4) 当 i=3 时，从调用 fibonacci 函数到返回，n 值如何变化？

Not found in current txt---4210691---3

### 3.2.4. 编程设计题

(1) 编程让用户输入两个整数，计算两个数的最大公约数并且输出之（要求用递归函数实现求最大公约数）。同时以单步方式执行该程序，观察递归过程。

解答：

1) 算法流程解释如下。

- ① 开始
- ② 输入 a, b
- ③ 调用 gcd (a, b)
- ④ 若 b=0, 返回 a
- ⑤ 否则，递归返回 gcd (a, a%b)

⑥ 输出返回值

⑦ 结束

2) 源程序清单

```
#include "stdio.h"
int gcd (int ,int );

int main(void){
    int a,b;
    scanf("%d%d",&a,&b);
    printf("%d",gcd(a,b));
}

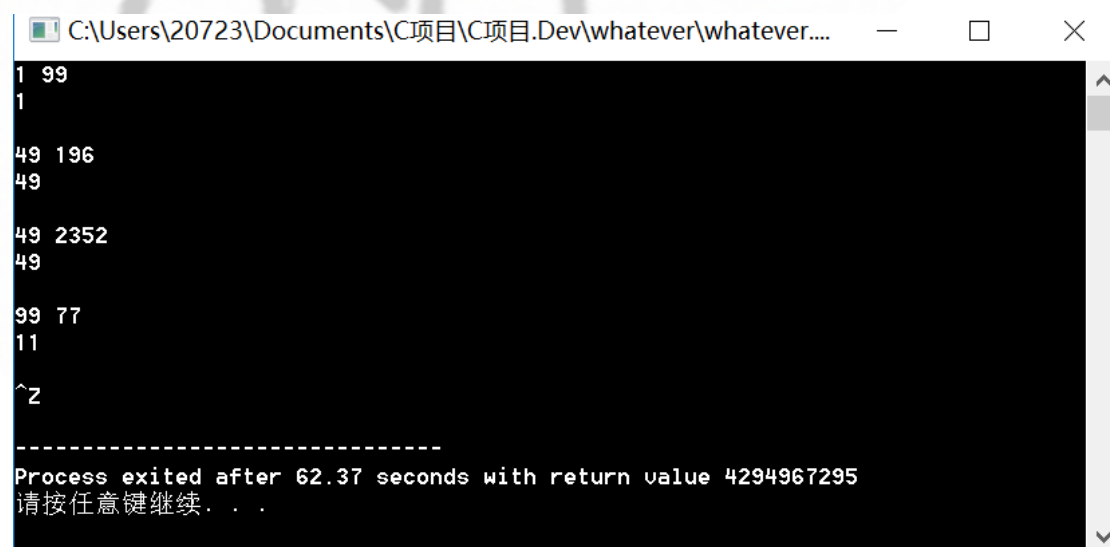
int gcd (int a,int b){
    int ret;
    if(b==0)ret=a;
    else ret=gcd(b,a%b);
    return ret;
}
```

3) 测试

(a) 测试数据

1 99 (理论结果 1)  
 49 196 (理论结果 49)  
 49 2352 (理论结果 49)  
 99 77 (理论结果 11)

(b) 运行结果如图 3-5 所示



```
C:\Users\20723\Documents\C项目\C项目.Dev\whatever\whatever....
1 99
1
49 196
49
49 2352
49
99 77
11
^Z
-----
Process exited after 62.37 seconds with return value 4294967295
请按任意键继续...
```



图 3-5 程序设计题（1）运行结果截图

（2）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。

**解答：**

1) 算法流程解释如下。

- ① 开始
- ② 利用筛法生成 10000 以内的质数表
- ③ 输入 x，a=3
- ④  $b=x-a$ ，利用质数表检测 a，b 是否为质数
- ⑤ 若是，输出
- ⑥ 否则， $a+=2$ ，回到④
- ⑦ 结束

2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int n,cnt;
    n=10000;//scanf("%d",&n);
    int a[n+1];
    for(cnt=0;cnt<=n;cnt++)a[cnt]=1;
    a[2]=1;
    for(cnt=2;cnt<=n;cnt++){
        if(a[cnt]==1){
            int i;
            for(i=2*cnt;i<=n;i+=cnt){
                a[i]=0;
            }
        }
    }
    int i,count;
    scanf("%d",&i);
    count=i;
    int x[i];
    while(count--){
        scanf("\n%d",&x[count]);
    }
    while(i--){
        int k,g;
```

```

    if(x[i]%2){
        printf("ERROR!\n");
        continue;
    }
    if(x[i]==4){
        printf("4=2+2\n");
        continue;
    }
    for(k=3;k<=x[i];k+=2){
        g=x[i]-k;
        if(a[k]==1&&a[g]==1){
            printf("%d=%d+%d\n",x[i],k,g);
            break;
        }
    }
}
return 0;
}

```

### 3) 测试

#### (a) 测试数据

78 (理论结果 78=5+73)

7878 (理论结果 7878=5+7873)

87 (理论结果 ERROR!)

8778 (理论结果 8778=17+8761)

8787 (理论结果 ERROR!)

#### (b) 运行结果如图 3-6 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\C语言作业\验证哥德...
5
78
7878
87
8778
8787
78=5+73
7878=5+7873
ERROR!
8778=17+8761
ERROR!

-----
Process exited after 6.131 seconds with return value 0
请按任意键继续. . .

```

图 3-6 程序设计题 (2) 运行结果截图

(3) 编写一个程序，证明对于在符号常量 BEGIN 和 END 之间的偶数这一猜测成立。例如，如果 BEGIN 为 10，END 为 20，程序的输出应为：

GOLDBACH'S CONJECTURE:

Every even number  $n \geq 4$  is the sum of two primes.

10=3+7

12=5+7

.....

20=3+17

解答：

1) 算法流程解释如下。

- ① 开始
- ② 输入 begin, end
- ③ 若 begin 为奇数,  $i = \text{begin} + 1$ ; 否则,  $i = \text{begin}$
- ④ 当  $i \leq \text{end}$ , 利用程序设计题 (2) 的算法进行质数和的分解
- ⑤  $i += 2$ , 回到④
- ⑥ 结束

2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int n,cnt;
    n=100000;//scanf("%d",&n);
    int a[n+1];
    for(cnt=0;cnt<=n;cnt++)a[cnt]=1;
    a[2]=1;
    for(cnt=2;cnt<=n;cnt++){
        if(a[cnt]==1){
            int i;
            for(i=2*cnt;i<=n;i+=cnt){
                a[i]=0;
            }
        }
    }
    int ax,b,count=1;
    while(scanf("%d%d",&ax,&b)!=EOF){
        if (ax%2){
            ax++;
        }
        int t;
```

```

for(t=ax;t<=b;t+=2){
    int k,g;
    if(t==4){
        printf("4=2+2\n");
    }
    for(k=3;k<=t;k+=2){
        g=t-k;
        if(a[k]==1&&a[g]==1){
            printf("%d=%d+%d\n",t,k,g);
            break;
        }
    }
    printf("\n");
}
return 0;
}

```

### 3) 测试

#### (a) 测试数据

9989 10000 (第一个数为奇数)

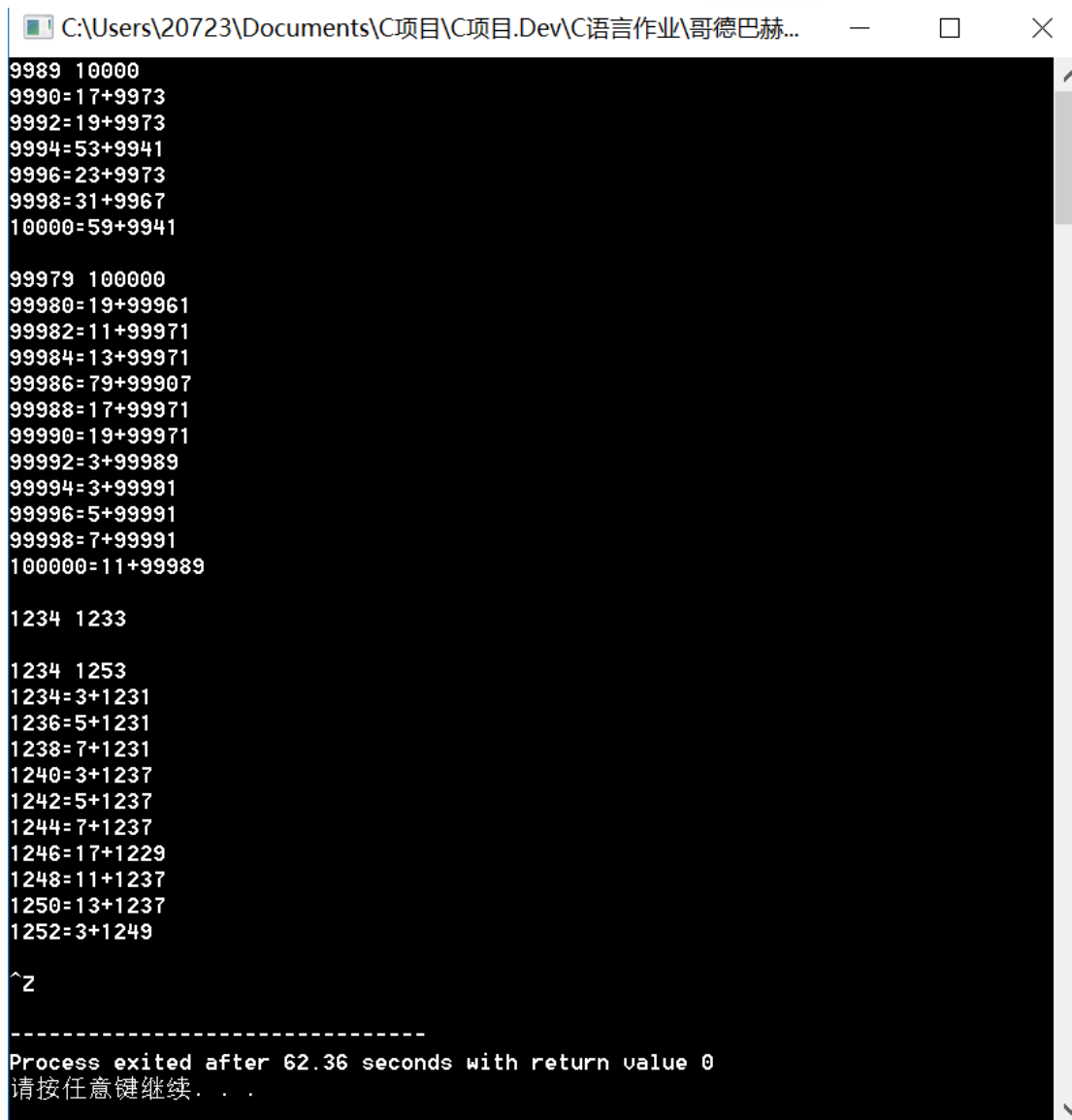
99979 100000

1234 1233 (第二个数比第一个数小, 不能输出)

1234 1253 (第二个数为奇数)

通过设置跨度较大的几组数据已经错误的输入来验证正确性

#### (b) 运行结果如图 3-7 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\C语言作业\哥德巴赫...
9989 10000
9990=17+9973
9992=19+9973
9994=53+9941
9996=23+9973
9998=31+9967
10000=59+9941

99979 100000
99980=19+99961
99982=11+99971
99984=13+99971
99986=79+99907
99988=17+99971
99990=19+99971
99992=3+99989
99994=3+99991
99996=5+99991
99998=7+99991
100000=11+99989

1234 1233
1234 1253
1234=3+1231
1236=5+1231
1238=7+1231
1240=3+1237
1242=5+1237
1244=7+1237
1246=17+1229
1248=11+1237
1250=13+1237
1252=3+1249
^Z
-----
Process exited after 62.36 seconds with return value 0
请按任意键继续. . .
    
```

图 3-7 程序设计题（3）运行结果截图

### 3.3.5. 选做题

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, 并编译和链接。然后运行生成的可执行文件。

源文件 file1.c 的内容为:

```

#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);
}

```

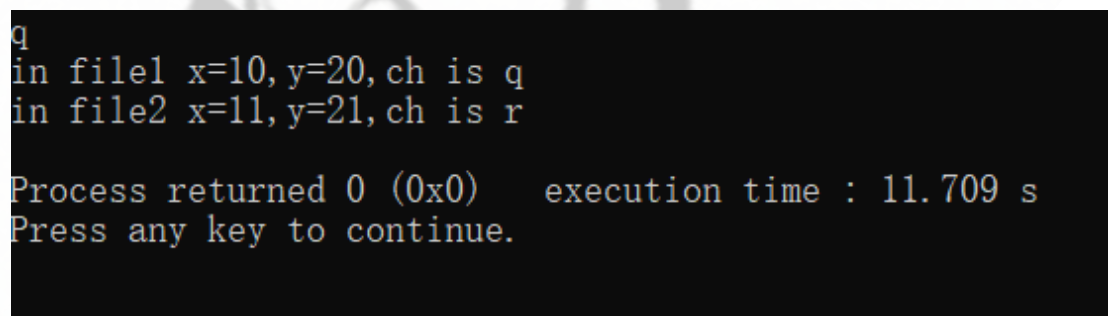
头文件 file.h 的内容为:

```

#include <stdio.h>
extern int x,y;    /* 外部变量的引用性说明 */
extern char ch;    /* 外部变量的引用性说明 */
void func1(void); /* func1函数原型 */

```

可执行程序的运行结果如图 3-8 所示



```

q
in file1 x=10,y=20,ch is q
in file2 x=11,y=21,ch is r

Process returned 0 (0x0)    execution time : 11.709 s
Press any key to continue.

```

图 3-8 选做题程序运行结果截图

### 3.3 实验小结

1. 程序调试题中刚开始不知道怎么进入函数，后发现单步进入即可实现该功能。
2. 助教检查程序运行时询问我刚进入函数时为什么 n 的值是个奇怪的数字，一开始没反应过来，由于设计函数参数传递，我随口回答那是 i 的地址，显然是

错误的回答，事后思考才明白那是  $n$  的存储单元原来的值，还未进行初始化，所以是个乱七八糟的值。



## 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、assert 宏的使用；
- (2) 练习带参数的宏定义、条件编译的使用；
- (3) 练习 assert 宏的使用；
- (4) 使用 Turbo C 2.0 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。

### 4.2 实验内容

#### 4.2.1. 源程序改错题

```

1  #include "stdio.h"
2  #define SUM a+b
3  #define DIF a-b
4  #define SWAP(a,b)  a=b,b=a
5  void main
6  {
7      int b, t;
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
11     bis:%d",SUM, SUM*DIF);
12     SWAP(a,b);
13     Printf("\nNow a=%d,b=%d\n",a,b);
14 }
```

解答：

- (1) 错误修改：

1) 第 2 行宏定义时缺少括号导致替换时语义错误，正确形式为：

```
#define SUM (a+b)
```

2) 第 3 行宏定义时缺少括号导致替换时语义错误，正确形式为：

```
#define DIF (a-b)
```

3) 第 4 行定义 SWAP 函数时不能达到预想目的，正确形式为：

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

4) 第 5 行 main 函数缺少参数说明，正确形式为：

```
void main(void)
```

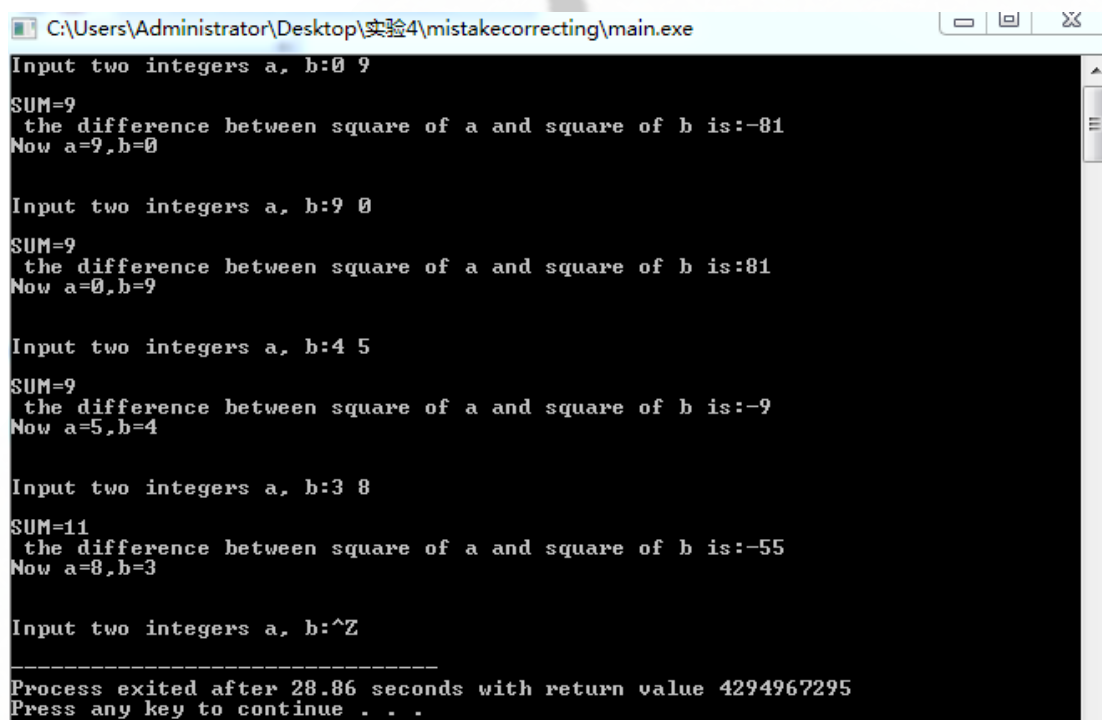
5) 第 7 行错误地将 a 声明为 t，正确形式为：

```
int a,b;
```

6) 第 9 行 scanf 函数中的逗号可能导致输入错误，正确形式为：

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

(2) 错误修改后运行结果如图 4-1 所示。



```

C:\Users\Administrator\Desktop\实验4\mistakecorrecting\main.exe
Input two integers a, b:0 9
SUM=9
the difference between square of a and square of b is:-81
Now a=9,b=0

Input two integers a, b:9 0
SUM=9
the difference between square of a and square of b is:81
Now a=0,b=9

Input two integers a, b:4 5
SUM=9
the difference between square of a and square of b is:-9
Now a=5,b=4

Input two integers a, b:3 8
SUM=11
the difference between square of a and square of b is:-55
Now a=8,b=3

Input two integers a, b:^Z

-----
Process exited after 28.86 seconds with return value 4294967295
Press any key to continue . . .
    
```

图 4-1 源程序改错题修改后程序运行截图

### 4.3.2. 源程序修改替换题

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。

```

void main(void)    //编译预处理（stdio.h/宏函数/sum函数的声明）
{
    int a, b, c;
    float d, e;
    printf("Enter three integers:");
    scanf("%d,%d,%d",&a,&b,&c);    //可能不需要逗号
    printf("\nthe maximum of them is %d\n",max(a,b,c));
    
```

```
printf("Enter two floating point numbers:");
scanf("%f,%f",&d,&e);    //可能不需要逗号
printf("\nthe sum of them is   %f\n",sum(d,e));
}
```

```
int max(int x, int y, int z)
{
    int t;
    if (x>y)
        t=x;
    else
        t=y;
    if (t<z)
        t=z;
    return t;
}
```

```
float sum(float x, float y)
{
    return x+y;
}
```

解答:

```
#include<stdio.h>
#define max(a,b,c) a>b?a>c?a:c:b>c?b:c
```

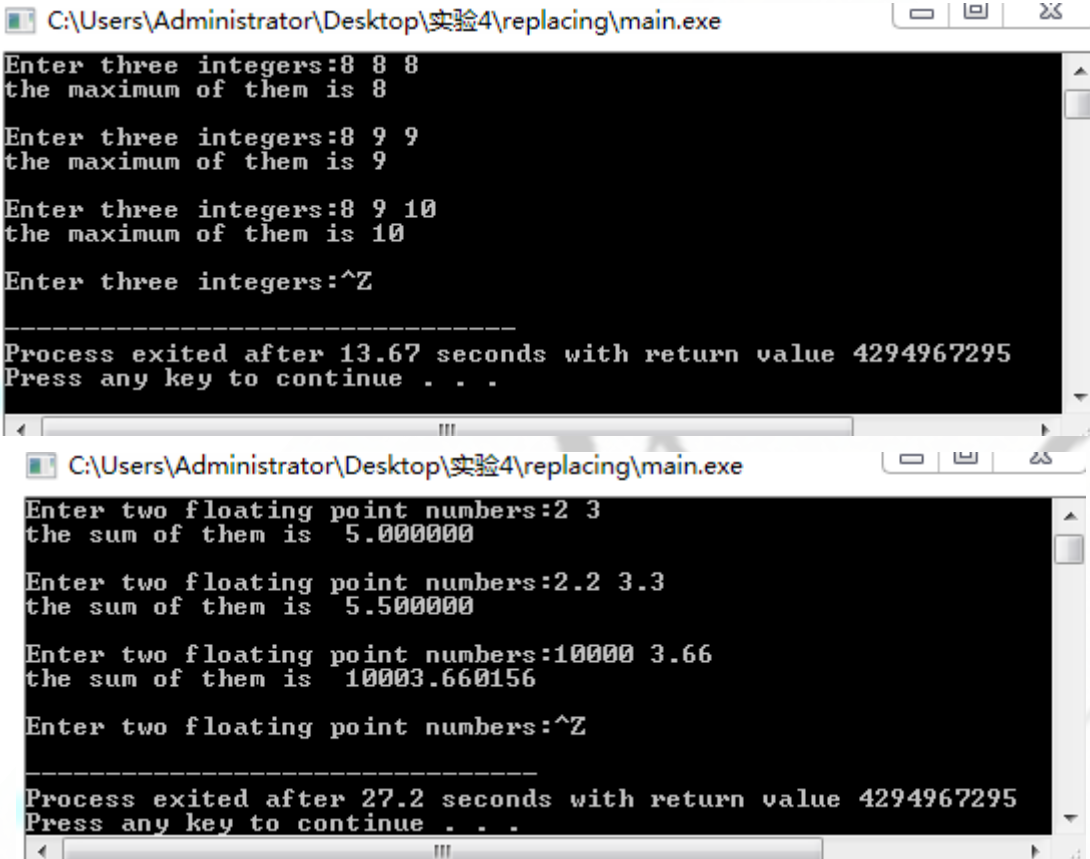
```
float sum (float,float);
void main(void)
{
    int a, b, c;
    float d, e;
    printf("Enter three integers:");
    scanf("%d%d%d",&a,&b,&c);
    printf("\nthe maximum of them is %d\n",max(a,b,c));

    printf("Enter two floating point numbers:");
    scanf("%f%f",&d,&e);
    printf("\nthe sum of them is   %f\n",sum(d,e));
}
```

```
float sum(float x, float y)
{
    return x+y;
}
```



修改后的程序运行结果如图 4-2 所示。



```

C:\Users\Administrator\Desktop\实验4\replacing\main.exe
Enter three integers:8 8 8
the maximum of them is 8

Enter three integers:8 9 9
the maximum of them is 9

Enter three integers:8 9 10
the maximum of them is 10

Enter three integers:^Z

-----
Process exited after 13.67 seconds with return value 4294967295
Press any key to continue . . .

C:\Users\Administrator\Desktop\实验4\replacing\main.exe
Enter two floating point numbers:2 3
the sum of them is 5.000000

Enter two floating point numbers:2.2 3.3
the sum of them is 5.500000

Enter two floating point numbers:10000 3.66
the sum of them is 10003.660156

Enter two floating point numbers:^Z

-----
Process exited after 27.2 seconds with return value 4294967295
Press any key to continue . . .
    
```

图4-2 源程序修改替换运行结果截图

#### 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 decimal\_fraction 时 watch 窗口中 x 为何值？

在返回 main 时，watch 窗口中 i 为何值？

- (3) 排除错误，使程序能正确输出面积 s 值的整数部分，不会输出错误信息 assertion failed。

```

#define R
void 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);
        printf("the integer fraction of area is %d\n", s_integer);
        assert((s-s_integer)<1.0);
    #endif
}

int integer_fraction(float x)
{
    int i=x;
    return i;
}

```

**解答：**

修改后的源程序如下。

```

#include <stdio.h>
#include <assert.h>
#define R
int integer_fraction(float);
void 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);
        printf("the integer fraction of area is %d\n", s_integer);
        assert((s-s_integer)<1.0);
    #endif
}

int integer_fraction(float x)
{
    int i=x;
    return i;
}

```

当测试数据为 5 时，进入函数 decimal\_fraction 时 watch 窗口中 x 为 78.5397491 在返回 main 时，watch 窗口中 i 为 78

watch 窗口如图 4-3 所示。

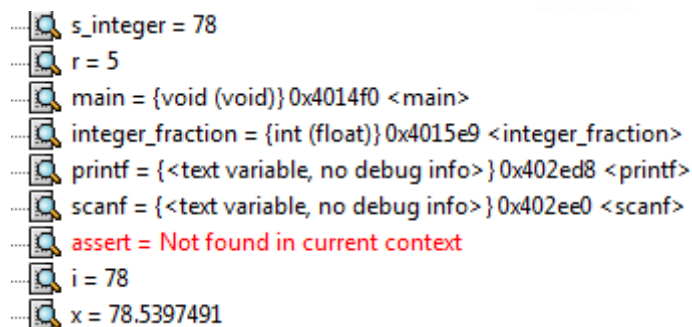


图 4-3 跟踪调试题 watch 窗口显示

#### 4.2.4. 编程设计题

(1) 三角形的面积是  $area = \sqrt{s(s-a)(s-b)(s-c)}$ ，其中  $s = (a+b+c)/2$ ， $a, b, c$  为三角形的三边，定义两个带参数的宏，一个用来求  $s$ ，另一个用来求  $area$ 。编写程序，用带参数的宏来计算三角形的面积。

**解答：**

1) 算法流程解释如下。

- (1) 利用宏定义定义  $s$  和  $area$  的表达式
- (2) 输入三角形三边长  $a, b, c$
- (3) 判断  $a, b, c$  能否构成三角形
- (4) 若能，利用宏  $s$  和  $area$  计算面积并输出
- (5) 结束

2) 源程序清单

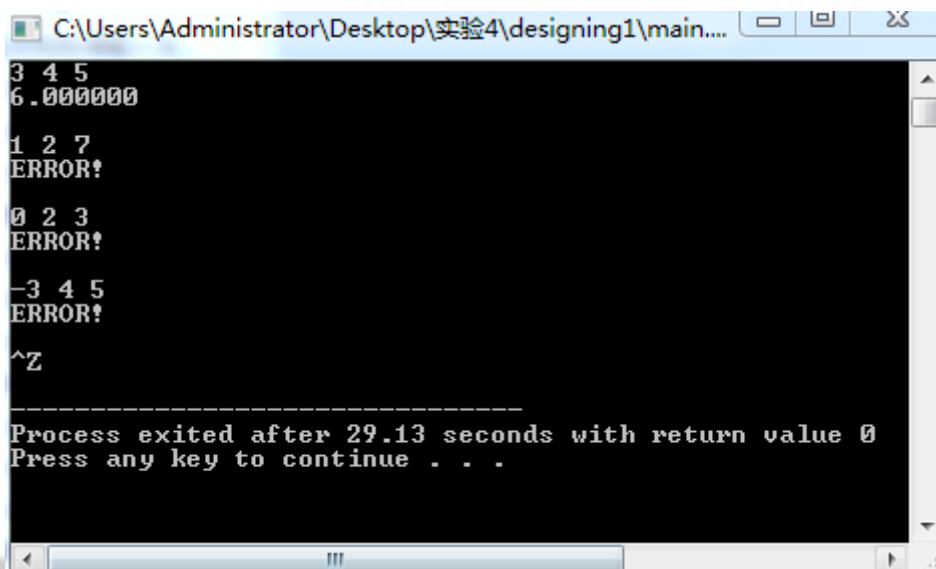
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define s(a,b,c) (a+b+c)/2
#define area(a,b,c) sqrt((a+b+c)*(b+c-a)*(a+c-b)*(a+b-c)/16.0)
int main()
{
    double a,b,c,s;
    scanf("%lf%lf%lf",&a,&b,&c);
    if(a>0&&b>0&&c>0&&a+b>c&&b+c>a&&c+a>b){
        printf("%lf\n",area(a,b,c));
    }
    else printf("ERROR!");
    return 0;
}
```

3) 测试

(a) 测试数据

- 3 4 5 (理论结果 6.000000)
- 1 2 7 (不满足两边之和大于第三边)
- 0 2 3 (三角形的边长不能为 0)
- 3 4 5 (三角形的边长不能为负数)

(b) 运行结果如图 4-4 所示。



```

C:\Users\Administrator\Desktop\实验4\designing1\main....
3 4 5
6.000000
1 2 7
ERROR!
0 2 3
ERROR!
-3 4 5
ERROR!
^Z
-----
Process exited after 29.13 seconds with return value 0
Press any key to continue . . .
  
```

图 4-4 程序设计题 1 运行结果截图

(2) 用条件编译方法来编写程序。输入一行电报文字，可以任选两种输出：一为原文输出；二为变换字母的大小写（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 s 文字，若#define CHANGE 0 则原文输出。

**解答：**

1) 算法流程解释如下。

(1) 开始

(2) 若 CHANGE 常量为 1，利用 ASCII 码将输入的每一个英文字母进行大小写转化后输出，其他字符直接输出

(3) 若 CHANGE 常量为 0，直接将每个字符依次输出

(4) 结束

2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>
#define CHANGE 1
int main()
{
    char c;
    #if CHANGE
        while(scanf("%c",&c)!='\n'){
            if(c>='a'&&c<='z')putchar(c-32);
            else if(c>='A'&&c<='Z')putchar(c+32);
            else if(c=='\n')break;
            else putchar(c);
        }
    #else
        while(scanf("%c",&c)!='\n'){
            if(c=='\n')break;
            putchar(c);
        }
    #endif
    return 0;
}
```

### 3) 测试

#### (a) 测试数据

abcDEFhijKLM1234567(CHANGE 1 )

abcDEFhijKLM1234567(CHANGE 0 )

#### (b) 测试结果如图 4-5 所示。

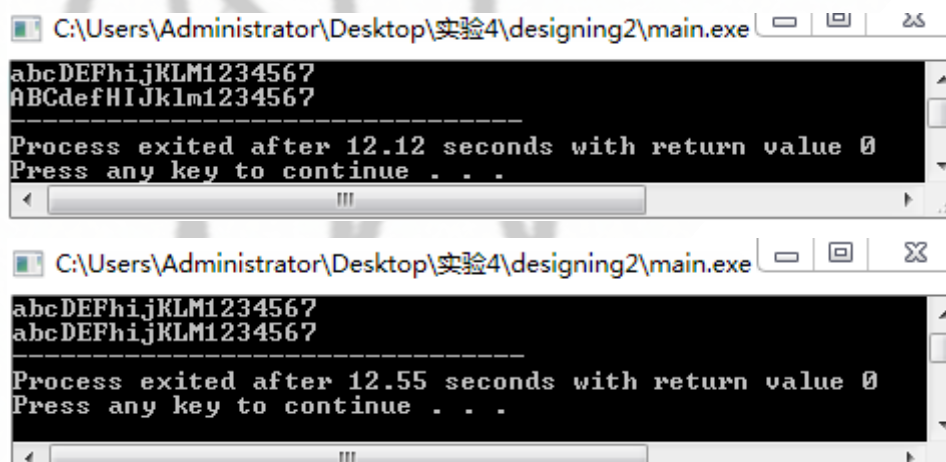


图4-5程序设计题2运行结果截图



### 4.3 实验小结

1. 在解答跟踪调试题时不太清楚 `assert` 断言需要将 `assert.h` 包含进来，导致编译不能通过，好在编译器会有提示，表示 `assert` 函数不明确，翻开书后知道了这一点，顺利解决了这个问题。
2. 程序设计题 1 刚开始没有判断输入数据的合理性就直接对数据进行处理，后意识到若数据不合理，可能导致 `sqrt` 函数不能正常运算，这才补上判断 `a, b, c` 的 `if` 语句。
3. 程序设计题 2 中进行大小写转换时误以为在 ASCII 码表中小写字母在前，大写字母在后，导致不能得到理想的输出结果，后查表后发现了这个问题，经过及时更正，立即解决了问题。

## 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

下面是用来将数组 a 中元素按升序排序后输出的源程序。分析源程序中存在的问题，并对源程序进行修改，使之能够正确完成任务。

源程序

```
1 #include<stdio.h>
2 int main(void)
3 {
4     int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};
5     void sort(int [],int);
6     int i;
7     sort(a[0],10);
8     for(i = 0; i < 10; i++)
9         printf("%6d",a[i]);
10    printf("\n");
11    return 0;
12 }
13 void sort(int b[], int n)
14 {
15     int i, j, t;
16     for (i = 0; i < n - 1; i++)
17         for (j = 0; j < n - i - 1; j++)
18             if(b[j] < b[j+1])
19                 t = b[j], b[j] = b[j+1], b[j+1] = t;
20 }
```

解答:

(1) 错误修改:

1) 第7行数组a作为函数参数传递时下标多余, 正确形式为:

sort(a, 10);

2) 第18行冒泡排序算法中不等号方向错误, 正确形式为:

if(b[j]>b[j+1]);

(2) 错误修改后运行结果如图 5-1 所示。

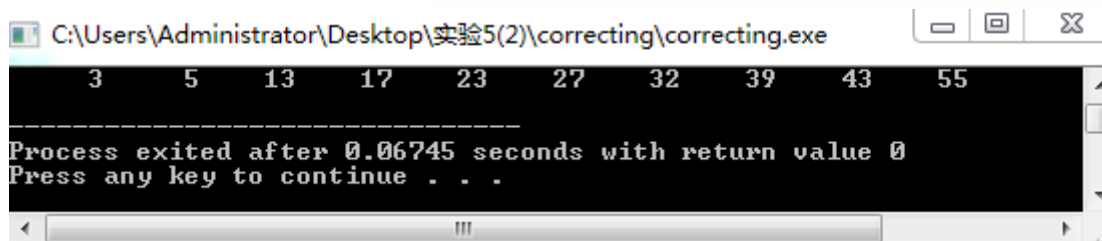


图 5-1 源程序改错题程序运行结果截图

### 5.2.2 源程序完善、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。请在源程序中的下划线处填写合适的代码来完善该程序。

源程序:

```
#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:

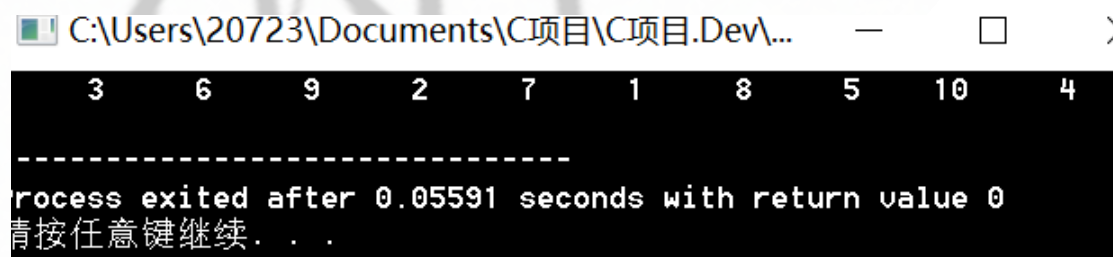
1) 完善后的源程序如下:

```

#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 */
        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) 完善后的程序运行结果如图 5-2 所示。



```

C:\Users\20723\Documents\C项目\C项目.Dev\...  —  □  >
3      6      9      2      7      1      8      5      10      4
-----
Process exited after 0.05591 seconds with return value 0
请按任意键继续. . .

```

图 5-2 源程序完善修改替换题 (1) 程序运行结果截图

(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。因此, 请采用做标记的办法修改 (1) 中的程序, 并使修改后的程序与 (1) 中的程序具有相同的功能。



解答:

1) 替换后的源程序如下:

```
#include <stdio.h>
#include <stdlib.h>
#define M 10
#define N 3

int main(void) {
    int a[M],i,n,cnt;
    for(i=0;i<M;i++){
        a[i]=1;           //数组初始化为1
    }
    i=-1;                 //确保第一次++i作为数组下标时值为1
    for(n=1;n<M;n++){     //n用来控制共报了几轮数，每轮淘汰一人
        for(cnt=0;cnt!=N;){ //cnt为报数变量
            if(a[++i]==1)cnt++; //不为1的话其对应的下标已被淘汰
            if(i>M-2)i=M;      //M号过后1号继续报数
        }
        if(i==M-1){
            a[M-1]=0;         //修改后i不再是应该被淘汰的下标
            printf("%6d",M);
        }
        else{
            a[i]=0;
            printf("%6d",i+1);
        }
    }
    for(i=0;i<M;i++)if(a[i])printf("%6d\n",++i);
    return 0;
}
```

2) 修改后的程序运行结果如图 5-3 所示。

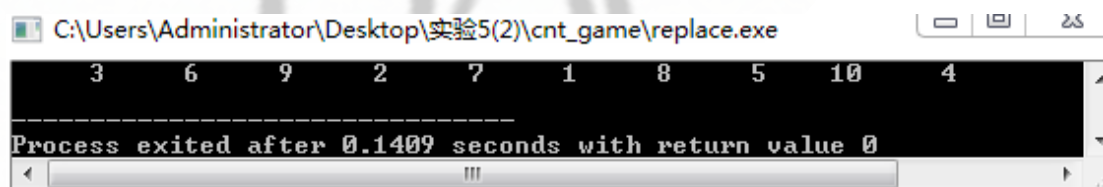


图 5-3 源程序完善修改替换题 (2) 程序运行结果截图

### 5.2.3 跟踪调试源程序

在下面所给的源程序中，函数 `strncat(s, t, n)` 本来应该将字符数组 `t` 的前 `n` 个字符连接到字符数组 `s` 中字符串的尾部。但函数 `strncat` 在定义时代码有误，



不能实现上述功能。请按下面的要求进行操作，并回答问题和排除错误。

(1) 单步执行源程序。进入函数 `strncat` 后观察表达式 `s`、`t` 和 `i`。当光条落在 `for` 语句所在行时，`i` 为何值？当光条落在 `strncat` 函数块结束标记（右花括号 `}`）所在行时，`s`、`t` 分别为何值？

(2) 分析函数出错的原因，排除错误，使函数正确实现功能，最后写出程序的输出结果。

源程序：

```
#include<stdio.h>
void strncat(char [],char [],int);
int main(void)
{
    char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";
        strncat(a, b, 4);
    printf("%s\n",a);
        return 0;
}
void strncat(char s[],char t[], int n)
```

```
{
    int i = 0, j;
    while(s[i++]) ;
    for(j = 0; j < n && t[j];)
        s[i++] = t[j++];
    s[i] = '\0';
}
```

修改后的源程序如下

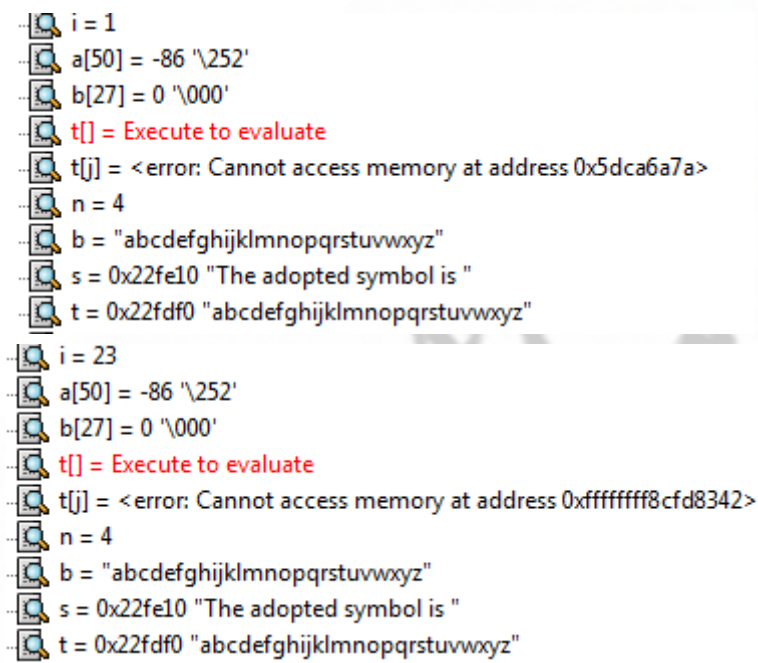
```
#include<stdio.h>
void strncat_(char [],char [],int);
int main(void)
{
    char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";
        strncat_(a, b, 4);
    printf("%s\n",a);
        return 0;
}
void strncat_(char *s,char *t, int n)
{
    int i = 0, j;
    while(s[i++]);
    i=1; //修改的地方
    for(j = 0; j < n && t[j];)
        s[i++] = t[j++];
}
```

```

        s[i] = '\0';
    }

```

单步执行源程序。进入函数 `strncat` 后表达式 `s=0x22fe10`、`t=0x22fdf0` 和 `i=1`。当光条落在 `for` 语句所在行时，`i=23`。当光条落在 `strncat` 函数块结束标记（右花括号 `}`）所在行时，`s=0x22fe10`、`t=0x22fdf0`。对应的 watch 窗口如图 5-4 所示。



5-4 跟踪调试题 watch 窗口截图

## 5.2.4 程序设计

编写并上机调试运行能实现以下功能的程序。

- (1) 编写一个程序,从键盘读取数据,对一个  $3 \times 4$  矩阵进行赋值,求其转置矩阵,然后输出原矩阵和转置矩阵。

**解答:**

- 1) 算法流程解释如下。

- ① 开始
- ② 定义两个二维数组,一个为  $3 \times 4$  (`a`),另一个为  $4 \times 3$  (`b`)
- ③ 输入 `a` 数组的每一个值
- ④ 将 `a` 数组的每一个值赋给 `b` 数组对应变换下标后的单元
- ⑤ 输出 `a`, `b` 数组的每一个值
- ⑥ 结束

- 2) 源程序清单

```

#include <stdio.h>
#include <stdlib.h>

```

```
int main()
{
    int a[3][4],b[4][3];
    int i,j;
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
            scanf("%d",&a[i][j]);
    for(i=0;i<3;i++){
        for(j=0;j<4;j++){
            b[j][i]=a[i][j];
            printf("%4d",a[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    for(j=0;j<4;j++){
        for(i=0;i<3;i++)
            printf("%4d",b[j][i]);
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

### 3) 测试

#### (a) 测试数据

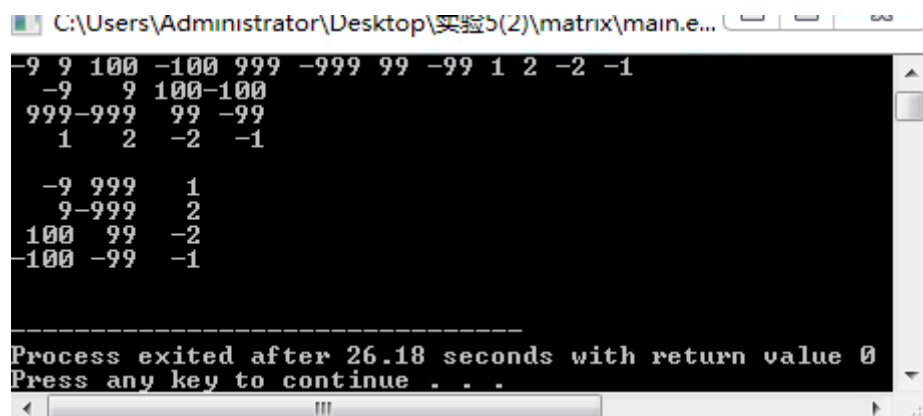
```
1 2 3 4 5 6 7 8 9 10 11 12
-9 9 100 -100 999 -999 99 -99 1 2 -2 -1
```

#### (b) 测试结果如图 5-5 所示

```
C:\Users\Administrator\Desktop\实验5(2)\matrix\main.exe
1 2 3 4 5 6 7 8 9 10 11 12
1 2 3 4
5 6 7 8
9 10 11 12

1 5 9
2 6 10
3 7 11
4 8 12

-----
Process exited after 15.55 seconds with return value 0
Press any key to continue . . .
```



```

C:\Users\Administrator\Desktop\实验5(2)\matrix\main.e...
-9 9 100 -100 999 -999 99 -99 1 2 -2 -1
-9 9 100 -100
999-999 99 -99
1 2 -2 -1

-9 999 1
9-999 2
100 99 -2
-100 -99 -1

-----
Process exited after 26.18 seconds with return value 0
Press any key to continue . . .
    
```

图 5-5 程序设计题（1）程序运行结果截图

- (2) 编写一个程序，其功能要求是：输入一个整数，将它在内存中二进制表示的每一位转换成为对应的数字字符，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) 算法流程解释如下。

- ① 开始
- ② 得到一个整数
- ③ 用逻辑尺取出其内存中的每一位，并转换为字符类型，倒序储存
- ④ 顺序输出字符数组中的每一位
- ⑤ 结束

- 2) 源程序清单

```

#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int n,i;
    scanf("%d",&n);
    int mask=1;
    char s[33];
    for(i=0;i<32;i++){
        if((n&mask)>>i)s[31-i]='1';
        else s[31-i]='0';
        mask<<=1;
    }
    s[32]='\0';
    puts(s);
    return 0;
}
    
```

}

### 3) 测试

#### (a) 测试数据

65535 2147483647 1 -1

(b) 测试结果如图 5-6 所示

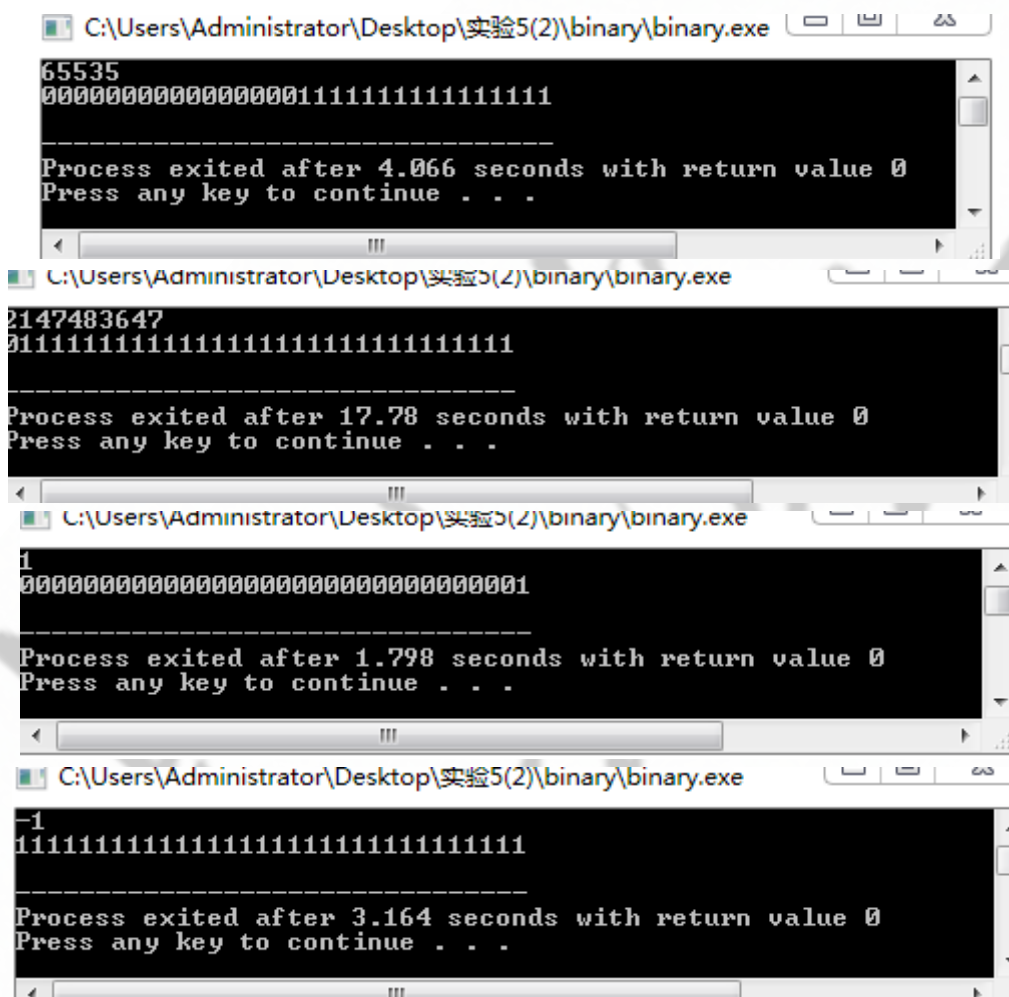


图 5-6 程序设计题（2）程序运行结果截图

- (3) 编写一个程序，其功能要求是：输入 n 个学生的姓名和 C 语言课程的成绩，将成绩按从高到低的次序排序，姓名同时作相应调整，输出排序后学生的姓名和 C 语言课程的成绩。然后，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，输出该成绩同学的姓名和 C 语言课程的成绩；否则输出提示 “not found!”。

解答：

- 1) 算法流程解释如下。

#### ① 开始



- ② 定义两个数组，一个用来保存成绩，一个用来保存姓名
- ③ 输入成绩和姓名
- ④ 冒泡排序
- ⑤ 输出排序结果
- ⑥ 输入要查询的数据，输出二分查找后的结果，与文件尾查询结束
- ⑦ 结束

## 2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void bubble_sort(int *gd,char (*s)[20],int n);
int mid_search(int g[],int n,int x);

int main()
{
    int n,i,x,pl;
    scanf("%d",&n);                                //输入学生数
    int gd[n];
    char nm[n][20];
    for(i=0;i<n;i++)
        scanf("%d %s",&gd[i],nm[i]);
    printf("\nAfter sorting: \n");
    bubble_sort(gd,nm,n);                            //对成绩进行排序
    for(i=0;i<n;i++)
        printf("%s  %4d\n",nm[n-i-1],gd[n-i-1]);
    while(scanf("%d",&x)!=EOF){
        pl=mid_search(gd,n,x);
        if(pl>=0)printf("%s\n\n",nm[pl]);            //pl若为-1，代表找不到
        else printf("NOT FOUND!\n\n");
    }
    return 0;
}

void strchg(char s[],char t[],int n){                //交换字符串函数
    char x[n];
    strcpy(x,s);
    strcpy(s,t);
    strcpy(t,x);
}
```

```

void bubble_sort(int *gd,char (*s)[20],int n){           //冒泡排序
    int i,j;
    for(i=0;i<n-1;i++){                                //共有n轮冒泡
        for(j=0;j<n-i;j++){
            if(gd[j]>gd[j+1]){
                gd[j]=gd[j]^gd[j+1],gd[j+1]=gd[j]^gd[j+1],gd[j]=gd[j]^gd[j+1];
                strchg(s[j],s[j+1],20);
            }
        }
    }
}

int mid_search(int g[],int n,int x){                    //二分查找函数
    int low=0,high=n-1,ret=-1,mid;                     //若找不到, -1提示错
    误
    do{
        mid=(low+high)/2;
        if(x>g[mid])low=mid+1;
        else if(x<g[mid])high=mid-1;
        else {
            ret=mid;                                    //若找到,ret为所查下标
            break;
        }
    }while(low<=high);
    return ret;                                         //返回查询结果
}

```

### 3) 测试

#### (a) 测试数据

```

10
98  bee
69  game
87  array
99  economy
75  fuel
92  kick
89  ill
72  utile
68  fail
100 exquisite

```

99  
77  
68  
69  
89  
101

(b) 测试结果如图 5-7 所示

```

C:\Users\Administrator\Desktop\实验5(2)\grade_sort\main.exe
10
98 bee
69 game
87 array
99 economy
75 fuel
92 kick
89 ill
72 utile
68 fail
100 exquisite

After sorting:
exquisite 100
economy 99
bee 98
kick 92
ill 89
array 87
fuel 75
utile 72
game 69
fail 68
99
economy
77
NOT FOUND!
68
fail
69
game
89
ill
101
NOT FOUND!
^Z

-----
Process exited after 201.1 seconds with return value 0
Press any key to continue . . .
    
```

图 5-7 程序设计题（3）程序运行结果截图

### 5.2.5 选做题程序设计

编写并上机调试运行能实现以下功能的函数和程序。

- (1) 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) 算法流程解释如下。

- ① 开始
- ② 输入字符串 `s`, `t`
- ③ 分别得到 `s`, `t` 的字符串长度
- ④ 将字符串 `s` 中第 `n` 个字符后的字符向后移 `x` 个单位 (`x` 为 `t` 的串长-1)
- ⑤ 将字符串 `t` 的每个字符一一插入
- ⑥ 结束

2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>

void strins_(char [],char [],int);

int main(void) {
    char s[1000],t[1000];
    int n;
    gets(s);
    gets(t);
    scanf("%d",&n);
    strins_(s,t,n);
    printf("%s",s);
    return 0;
}

void strins_(char s[],char t[],int n){
    int i=-1,num_t=-1;
    while(s[++i]);           //最终i得到字符串s的长度
    while(t[++num_t]);       //最终num_t得到字符串t的长度
    num_t--;                 //去掉'\0'的长度
    for(i>n-1;i--){
        s[i+num_t]=s[i];     //第n个字符后的字符向后移num_t个字节
    }
    for(i=0;i<num_t;i++)
```

```
s[n+i]=t[i];           //将字符串t插入
}
```

### 3) 测试

#### (a) 测试数据

abcdefghijklmnopqrstuvwxyz

12345678910

13

#### (b) 测试结果如图 5-8 所示

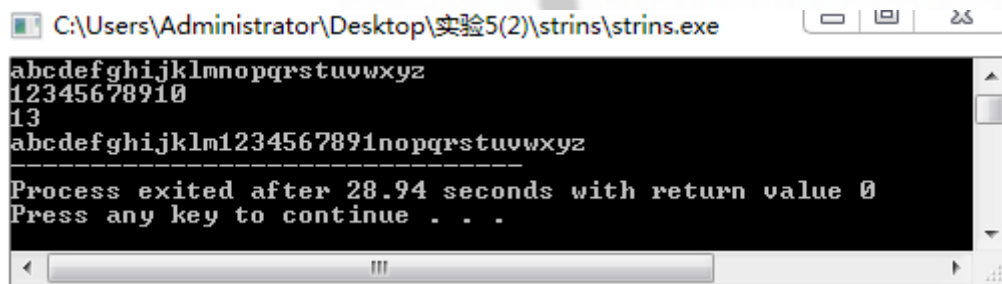
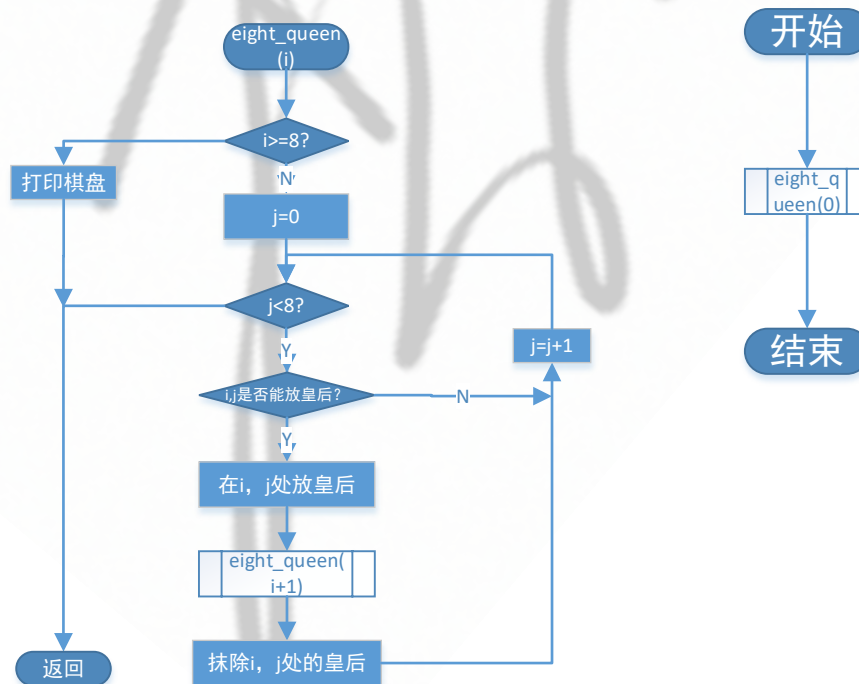


图 5-8 选做题（1）程序运行结果截图

- (2) 编写一个实现八皇后问题的程序，即：在  $8 \times 8$  方格国际象棋盘上放置 8 个皇后，任意两个皇后不能位于同一行、同一列或同一斜线（正斜线或反斜线）上，并输出所有可能的放法。

解答：

- 1) 算法流程解释如下。





## 2) 源程序清单

```
#include <stdio.h>
#include <stdlib.h>

void eight_queens(int (*a)[8],int i);
int check(int (*a)[8],int i,int j);
void printbase(int (*a)[8],int i,int j);

int count=0;           //count用来计数，共有多少种结果

int main(void) {
    int a[8][8]={0};      //a数组模拟8*8棋盘
    eight_queens(a,0);
    printf("in total: %d\n",count);
    return 0;
}

void eight_queens(int (*a)[8],int i){
    if(i>=8){
        printbase(a,8,8);      //如果i>=8,说明已有8个符合条件的皇后
        count++;
    }
    else{
        int j;
        for(j=0;j<8;j++){
            if(check(a,i,j)){
                a[i][j]=1;      //如果此处可以放皇后，标记为1
                eight_queens(a,i+1); //继续放下一个皇后
                a[i][j]=0;      //抹除皇后
            }
        }
    }
}

int check(int (*a)[8],int i,int j){      //检查某一位置是否可以放皇后的函数
    int ci,cj,ret=1;
    for(ci=0;ci<8;ci++){      //检查行
        if(ci==i)continue;
        if(a[ci][j]==1)return 0;
    }
    for(cj=0;cj<8;cj++){      //检查列
        if(cj==j)continue;
        if(a[i][cj]==1)return 0;
    }
}
```

```

    for(ci=i-1,cj=j-1;ci>=0&&cj>=0;ci--,cj--){//检查左上方的斜线
        if(a[ci][cj]==1)return 0;
    }
    for(ci=i+1,cj=j+1;ci<8&&cj<8;ci++,cj++){//检查右下方的斜线
        if(a[ci][cj]==1)return 0;
    }
    for(ci=i-1,cj=j+1;ci>=0&&cj<8;ci--,cj++){//检查左下方的斜线
        if(a[ci][cj]==1)return 0;
    }
    for(ci=i+1,cj=j-1;ci<8&&cj>=0;ci++,cj--){//检查右上方的斜线
        if(a[ci][cj]==1)return 0;
    }
    return ret;
}

void printbase(int (*a)[8],int i,int j){           //打印棋盘函数
    printf("*****\n");
    int ci,cj;
    for(ci=0;ci<i;ci++){
        printf("*");
        for(cj=0;cj<j;cj++){
            printf(" %d",a[ci][cj]);
        }
        printf("*\n");
    }
    printf("*****\n\n");
}

```

### 3) 测试

#### (a) 测试数据

无

#### (b) 测试结果如图 5-9 所示。

```

C:\Users\Administrator\Desktop\实验5(2)\eight_queens\eight_queens.exe

* 1 0 0 0 0 0 0 0*
* 0 0 1 0 0 0 0 0*
* 0 0 0 0 1 0 0 0*
*****
*****
* 0 0 0 0 0 1 0*
* 0 0 0 0 1 0 0*
* 0 0 1 0 0 0 0*
* 1 0 0 0 0 0 0*
* 0 0 0 0 0 1 0*
* 0 0 0 0 0 0 1*
* 0 1 0 0 0 0 0*
* 0 0 0 1 0 0 0*
*****
*****
* 0 0 0 0 0 0 1*
* 0 1 0 0 0 0 0*
* 0 0 0 1 0 0 0*
* 1 0 0 0 0 0 0*
* 0 0 0 0 0 1 0*
* 0 0 0 0 1 0 0*
* 0 0 1 0 0 0 0*
* 0 0 0 0 1 0 0*
*****
*****
* 0 0 0 0 0 0 1*
* 0 1 0 0 0 0 0*
* 0 0 0 0 1 0 0*
* 0 0 1 0 0 0 0*
* 1 0 0 0 0 0 0*
* 0 0 0 0 0 1 0*
* 0 0 0 1 0 0 0*
* 0 0 0 0 1 0 0*
*****
*****
* 0 0 0 0 0 0 1*
* 0 0 1 0 0 0 0*
* 1 0 0 0 0 0 0*
* 0 0 0 0 1 0 0*
* 0 1 0 0 0 0 0*
* 0 0 0 0 1 0 0*
* 0 0 0 0 0 1 0*
* 0 0 0 1 0 0 0*
*****
*****
* 0 0 0 0 0 0 1*
* 0 0 0 1 0 0 0*
* 1 0 0 0 0 0 0*
* 0 0 1 0 0 0 0*
* 0 0 0 0 1 0 0*
* 0 1 0 0 0 0 0*
* 0 0 0 0 0 1 0*
* 0 0 0 0 1 0 0*
*****
in total: 92

-----
Process exited after 0.4064 seconds with return value 0
Press any key to continue . . .
    
```

图 5-9 选做题（2）程序运行结果截图

### 5.3 实验小结

1. 第一次做程序填空题，确实非常不适应，第一遍看程序时完全不知道它的每一个变量的作用是什么，也不知道是想怎样完成这个任务，后来结合注释才慢慢明白它的思路是怎样，这才琢磨出那几个空填什么。所以做程序填空题首要任务是弄清楚这个程序的思路是什么，也就是它的算法是怎样的，这才是最关键的。

2. 成绩排序那个题，一开始不清楚二维数组作为函数参数该如何传递，经过上网搜索资料后得知是以指针数组的形式进行参数传递，弥补了我一个知识漏洞。

3. 最后一个选做题，乍一看能想到的只有穷举，然后多所有情况挨个进行判断，但这种算法的时间复杂度实在太高，属于不可解类型。后查找资料，了解到回溯算法，看了几个样例程序后，便开始自己动手写，最终成功。



## 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

下面程序是否存在错误？如果存在，原因是什么？如果存在错误，要求在计算机上对这个例子程序进行调试修改，使之能够正确执行。

```
1#include "stdio.h"
2void main(void)
3{
4    float *p;
5    scanf("%f",p);
6    printf("%f\n",*p);
7}
```

解答：

(1) 错误修改：

1) 第 3 行未对指针 p 进行初始化，正确形式为：

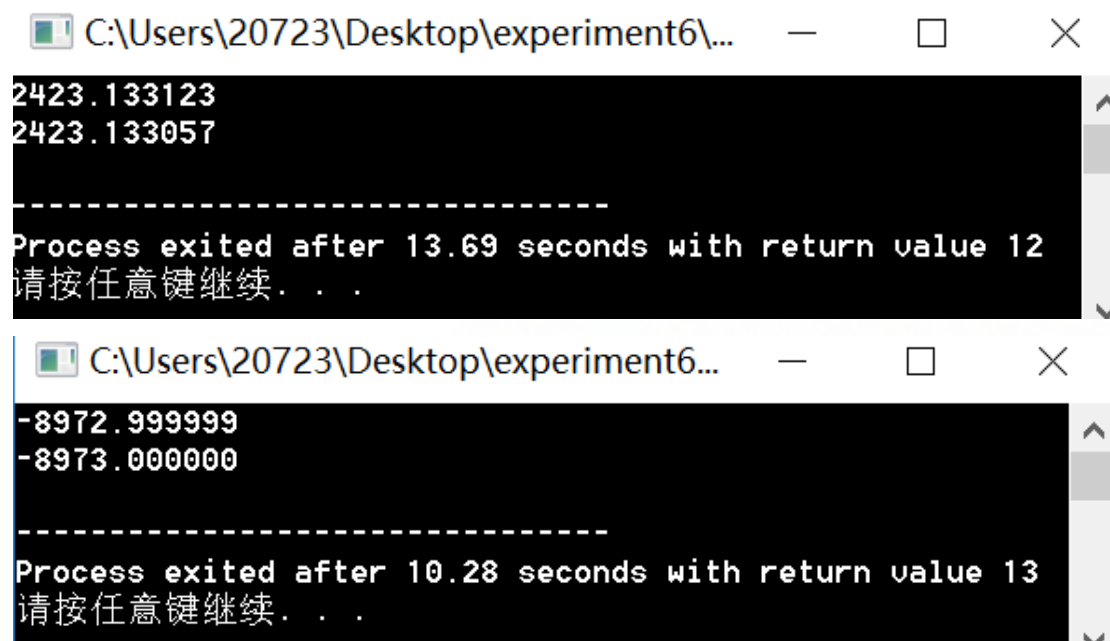
```
float f, *p=&f;
```

(2) 错误修改后运行结果如图 6-1 所示。



```
C:\Users\20723\Desktop\experiment6...
0.000000
0.000000
-----
Process exited after 6.684 seconds with return value 9
请按任意键继续...
```





```
C:\Users\20723\Desktop\experiment6\...  
2423.133123  
2423.133057  
-----  
Process exited after 13.69 seconds with return value 12  
请按任意键继续. . .  
  
C:\Users\20723\Desktop\experiment6...  
-8972.999999  
-8973.000000  
-----  
Process exited after 10.28 seconds with return value 13  
请按任意键继续. . .
```

图 6-1 源程序改错题程序运行结果截图

### 6.2.2 源程序完善、修改、替换

(1) 下面的程序通过函数指针和菜单选择来调用字符串拷贝函数或字符串连接函数，请在下划线处填写合适的表达式、语句、或代码片段来完善该程序。

```
#include "stdio.h"  
#include "string.h"  
void main(void)  
{  
  
    char a[80],b[80],c[160],*result=c;  
    int choice,i;  
    do{  
        printf("\t\t1 copy string.\n");  
        printf("\t\t2 connect string.\n");  
        printf("\t\t3 exit.\n");  
        printf("\t\tinput a number (1-3) please!\n");  
        scanf("%d",&choice);  
    }while(choice<1 || choice>3);  
    switch(choice){  
    case 1:  
        p=strcpy;  
        break;  
    case 2:  
        p=strcat;  
        break;  
    case 3:
```

```

        goto down;
    }
    getchar();
    printf("input the first string please!\n");
    i=0;

    printf("input the second string please!\n");
    i=0;

    result=      (a,b);
    printf("the result is %s\n",result);
down:
    ;
}

```

(2) 请上机运行第 (1) 题程序，使之能按下面要求输出结果：((输入) 表示该数据是键盘输入数据)

```

1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!

```

2 (输入)

```

input the first string please!
the more you learn, (输入)
input the second string please!
the more you get. (输入)
the result is the more you learn,the more you get.

```

解答：

(1) 完善后的源程序如下：

```

#include "stdio.h"
#include "string.h"
void main(void)
{
    char* (*p)(char *,char *);
    char a[80],b[80],c[160],*result=c;
    int choice,i;
    do{
        printf("\t\t1 copy string.\n");
        printf("\t\t2 connect string.\n");
        printf("\t\t3 exit.\n");
        printf("\t\tinput a number (1-3) please!\n");
        scanf("%d",&choice);
    }while(choice<1 || choice>3);
}

```

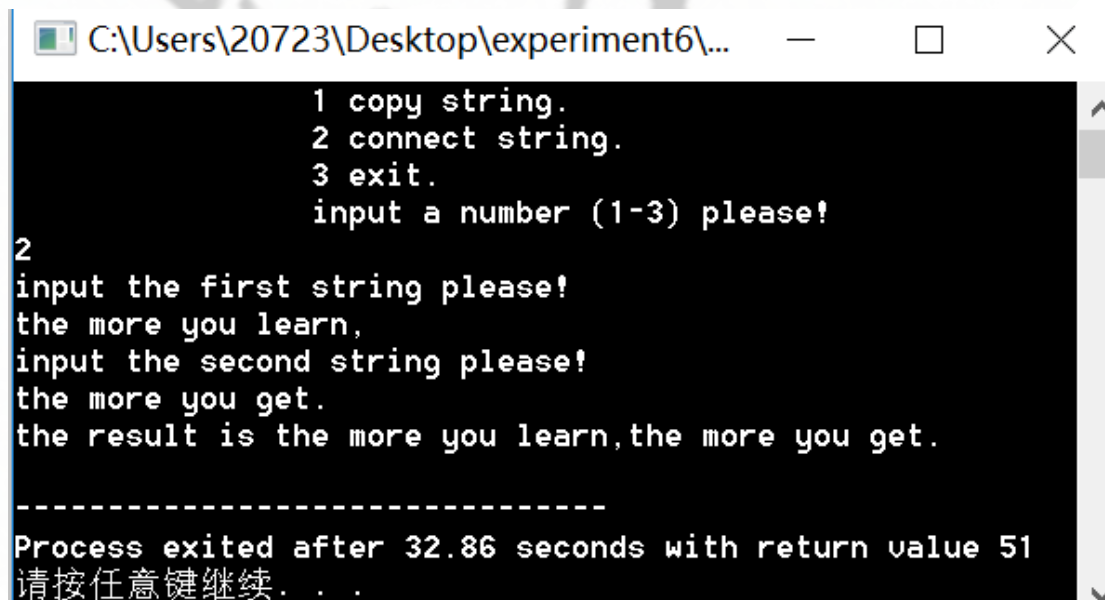
```

switch(choice){
case 1:
    p=strcpy;
    break;
case 2:
    p=strcat;
    break;
case 3:
    goto down;
}
getchar();
printf("input the first string please!\n");
i=0;
while((a[i++]=getchar())!='\n');a[--i]='\0';//相当于gets(a);输入字符串a

printf("input the second string please!\n");
i=0;
while((b[i++]=getchar())!='\n');b[--i]='\0';//相当于gets(b);输入字符串b

result=p(a,b);
printf("the result is %s\n",result);
down:
;
}
    
```

(2) 完善后的程序运行结果如图6-2所示。



```

C:\Users\20723\Desktop\experiment6\...
1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!
2
input the first string please!
the more you learn,
input the second string please!
the more you get.
the result is the more you learn,the more you get.
-----
Process exited after 32.86 seconds with return value 51
请按任意键继续. . .
    
```

图 6-2 源程序完善修改替换题程序运行结果截图

### 6.2.3 跟踪调试源程序

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：

源程序清单：

```
#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)
{
    char *p=s;
    while(*s++=*t++);
    return (p);
}
```

//修改的地方

(1)进入 strcpy 时 watch 窗口中 s 为 000000000062FE30，返回 main 时，watch 窗口中 s 为 000000000062FE4D。

(2) 排除错误后程序运行结果如图 6-3 所示。



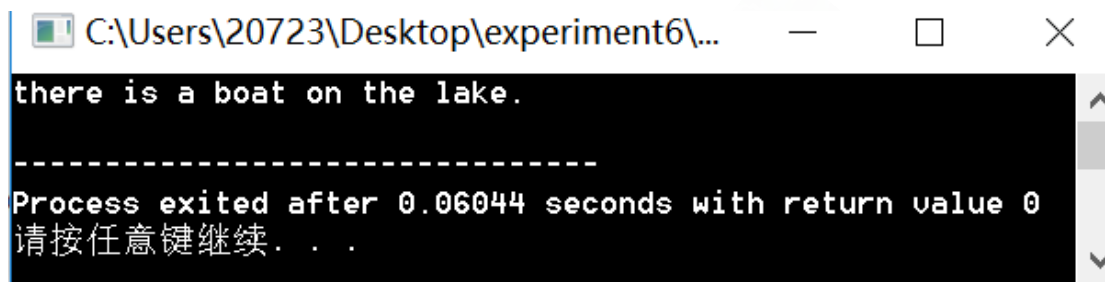


图 6-3 跟踪调试题程序运行结果截图

## 6.2.4 程序设计

(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 位并以数字字符的形式进行显示。

**解答：**

(1) 算法流程解释如下。

- ① 得到整数  $x$ ，并用字符指针指向内存模型中的最后一个字节(小段模式)
- ② 读出  $t$  指针所指单元，并取出高低 4 位，转换为 16 进制字符后输出， $t$  指针依次向前移动，并重复上述操作，直至指向内存中的第一个字节
- ③ 结束

(2) 源程序清单。

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    long x;
    char *t,b,c;
    unsigned int _b,_c;
    while(scanf("%ld",&x)!=EOF){
        int i;
        t=(char *)&x;
        t=t+3; //本机器使用小端模式
        for(i=0;i<4;i++){
            _b=*t>>4&0xf;
            _c=*t&0xf;
            if(_b<10)b=(char)(_b+'0');//转换成字符进行输出
            else b=(char)(_b-10+'A');
            if(_c<10)c=(char)(_c+'0');
            else c=(char)(_c-10+'A');
            printf("%c %c ",b,c);
        }
    }
}
```



```

        t--;
    }
    printf("\n");
}
return 0;
}

```

(3) 测试。

(a) 测试数据。

```

-1
-16
4294967295
262143
9876543210
-19283746

```

(b) 测试结果如图 6-4 所示。

```

-1
F F F F F F F F
-16
F F F F F F F 0
4294967295
F F F F F F F F
262143
0 0 0 3 F F F F
9876543210
4 C B 0 1 6 E A
-19283746
F E D 9 C 0 D E
^Z

-----
Process exited after 84.36 seconds with return value 0
请按任意键继续. . .

```

图 6-4 程序设计题（1）程序运行结果截图

(2) 利用大小为  $n$  的指针数组指向用 `gets` 函数输入的  $n$  行，每行不超过 80 个字符。编写一个函数，它将每一行中连续的多个空格字符压缩为一个空格字符。在调用函数中输出压缩空格后的各行，空行不予输出。

解答:

(1) 算法流程解释如下。

- ① 得到一个字符数组，用一个字符指针指向这个数组
- ② 定义 i（读下标），j（写下标）
- ③ 借指针用 i 遍历整个字符数组，读到非空格时写，第一个空格也写，  
然后跳过后面相邻的空格，读到第一个非空格字符再开始写
- ④ 重新输出这个数组
- ⑤ 将上述所有操作执行 n 次
- ⑥ 结束

(2) 源程序清单。

```
#include <stdio.h>
#include <stdlib.h>

void space_cut(char *a);

int main(int argc, char *argv[]) {
    int n,i;
    char *p[100],c[100][80];
    scanf("%d",&n);
    getchar();
    for(i=0;i<n;i++){
        gets(c[i]);           //得到n行字符串
        p[i]=c[i];
    }
    printf("\n");
    for(i=0;i<n;i++){
        space_cut(p[i]);
        if(p[i][0]){          //如果是空行，则不予输出
            puts(p[i]);
            printf("\n");
        }
    }
    return 0;
}

void space_cut(char a[]){
    int ci=0,j=0;              //ci是读下标，j是写下标
    while(a[ci]!='\0'){
        if(a[ci]!=' '){        //不是空格直接写
            a[j++]=a[ci++];
        }
        else{
            while(a[ci]==' '){
                ci++;
            }
            a[j]=a[ci];
            j++;
            ci++;
        }
    }
    a[j]='\0';
}
```

```

        a[j++]=' ';                //是空格则跳过后所有空格
        while(a[++ci]==' ');
        a[j++]=a[ci++];
    }
}
a[j]='\0';
}

```

(3) 测试。

(a) 测试数据。

8

skfdljfslkd (\*&879743#\$^%(\*#\$@#":>?<?> sdfjj

(空行)

(空行)

fjdkfksdfkjd kfdiurewioi nmmmvncxv./?.?>>?

"::"

(空行)

fdskfdkf jfkds kfds43298743284

98432748 (接上一行)

987432439 984328dfkjfdsk10-lew';.ds lkfds (b)

测试结果如图 6-5 所示。

```

8
skfdljfsl dk      (*&879743#$$^%(*#$$@#":>?<?> sdfjj

fjdkfk sdfkj d      kfdiurewioi      nmmuncxu./.?>>?
"::"

fdskfdkf      jfk dsk      kfds43298743284      98432748
987432439      984328dfkjfdskl0-lew';.ds      lkfds

skfdljfsl dk (*&879743#$$^%(*#$$@#":>?<?> sdfjj
fjdkfk sdfkj d kfdiurewioi nmmuncxu./.?>>?
"::"

fdskfdkf jfk dsk kfds43298743284 98432748
987432439 984328dfkjfdskl0-lew';.ds lkfds

-----
Process exited after 62.46 seconds with return value 0
请按任意键继续. . .
    
```

图 6-5 程序设计题（2）程序运行结果截图

(3) 编写一个程序，输入  $n$  个整数，排序后输出。排序的原则由命令行可选参数  $-d$  决定，并且有参数  $-d$  时按递减顺序排列，否则按递增数列排列。要求将排序算法定义为函数，利用指向函数的指针使该函数实现递增或递减排列。

解答：

(1) 算法流程解释如下。

- ① 对比 main 函数第二个参数是否与 “ $-d\backslash 0$ ” 完全一致
- ② 若是，从第三个参数开始依次将字符串转换为整型，并进行冒泡降序排列，并跳转④，否则跳转③
- ③ 从第二个参数开始依次转换为整型，并进行冒泡升序排列
- ④ 将排序后的数组依次输出
- ⑤ 结束

(2) 源程序清单。

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int compare(int ,int);
void bubble_sort(int *,int ,int );
    
```

```

int main(int argc, char *argv[]) {
    int a[argc],i;
    void (*p)(int *,int ,int );
    p=bubble_sort;
    if(argv[1][0]=='-'&&argv[1][1]=='d'&&argv[1][2]=='\0'){
        for(i=0;i<argc-2;i++)a[i]=str_to_int(argv[i+2]);
        p(a,argc-2,0);
        for(i=0;i<argc-2;i++)printf("%d ",a[i]);
    }
    else {
        for(i=0;i<argc-1;i++)a[i]=str_to_int(argv[i+1]);
        p(a,argc-1,1);
        for(i=0;i<argc-1;i++)printf("%d ",a[i]);
    }
    return 0;
}

int compare(int a,int b){
    return a>b;
}

void bubble_sort(int *a,int n,int key){    //冒泡排序，key用来指定升序还是降序
    int i,j;
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++)
            if(compare(a[j],a[j+1])==key)
                a[j]=a[j]^a[j+1],a[j+1]=a[j]^a[j+1],a[j]=a[j]^a[j+1];//交换两元素
    }
}

int str_to_int(char *s){                //将字符串转换为整型数据
    int l=strlen(s),ret=0,i;
    if(*s=='-'){                        //负数的情况
        for(i=1;i<l;i++)ret=10*ret+(s[i]-'0');
        ret*=-1;
    }
    else for(i=0;i<l;i++)ret=10*ret+(s[i]-'0');//正数的情况
    return ret;
}

```

(3) 测试。

(a) 测试数据。

98 89 91 90 97 96 99 85 93



```
-d 19 49 27 67 38 18 27
19 49 27 67 38 18 27
-d -76 -28 48 28 -13 12 -32 43
```

(b) 测试结果如图 6-6 所示。

```
C:\Users\20723>C:\Users\20723\Desktop\experiment6\design3\design3 98 89 91 90 97 96 99 85 93
85 89 90 91 93 96 97 98 99
C:\Users\20723>C:\Users\20723\Desktop\experiment6\design3\design3 -d 19 49 27 67 38 18 27
67 49 38 27 27 19 18
C:\Users\20723>C:\Users\20723\Desktop\experiment6\design3\design3 19 49 27 67 38 18 27
18 19 27 27 38 49 67
C:\Users\20723>C:\Users\20723\Desktop\experiment6\design3\design3 -d -76 -28 48 28 -13 12 -32 43
48 43 28 12 -13 -28 -32 -76
```

图 6-6 程序设计题（3）程序运行结果截图

(4) 设某个班有  $N$  个学生，每个学生修了  $M$  门课程（用 `#define` 定义  $N$ 、 $M$ ）。输入  $M$  门课程的名称，然后依次输入  $N$  个学生中每个学生所修的  $M$  门课程的成绩并且都存放到相应的数组中。编写下列函数：

- 计算每个学生各门课程平均成绩；
- 计算全班每门课程的平均成绩；
- 分别统计低于全班各门课程平均成绩的人数；
- 分别统计全班各门课程不及格的人数和 90 分以上（含 90 分）的人数。

在调用函数中输出上面各函数的计算结果。（要求都用指针操作，不得使用下标操作。）

**解答：**

(1) 算法流程解释如下。

- ① 用整型二维数组  $[N][M]$  保存  $N$  个学生的  $M$  门课成绩
- ② 对于每个  $i$ （行）遍历所有  $j$ （列）计算平均值得到每个学生的各门课程平均成绩
- ③ 对于每个  $j$ （列）遍历所有  $i$ （行）计算平均值得到全班每门课的平均成绩
- ④ 对于每个  $j$ （列）遍历所有  $i$ （行）依次与②计算得到的平均值进行比较，得到低于全班各门课程平均成绩的人数
- ⑤ 对于每个  $j$ （列）遍历所有  $i$ （行）依次与 60 进行比较，得到全班各门课程不及格人数
- ⑥ 对于每个  $j$ （列）遍历所有  $i$ （行）依次与 90 进行比较，得到全班各

门课程 90 分以上（含 90 分）的人数

⑦ 将上述计算结果依次输出

⑧ 结束

(2)源程序清单。

```
#include <stdio.h>
#include <stdlib.h>
#define N 7
#define M 6

void _stu_ave(int (*a)[M],int n,double *p);
void _sub_ave(int (*a)[M],int n,double *p);
void _lower_ave(int (*a)[M],int n,int *p,double *q);
void _failure(int (*a)[M],int n,int *p);
void _good(int (*a)[M],int n,int *p);

int main(int argc, char *argv[]) {
    int a[N][M],i,j,sum_stu=0,sum_sub=0,failure[M],sub_avr[M],good[M];
    char subname[M][10];
    for(i=0;i<M;i++)scanf("%s",subname+i);
    double stu[N],sub[M];
    for(i=0;i<N;i++)
        for(j=0;j<M;j++)
            scanf("%d",&a[i][j]);
    _stu_ave(a,N,stu);           //计算每个学生的平均值
    _sub_ave(a,N,sub);           //计算每个学科的平均值
    _lower_ave(a,N,sub_avr,sub); //计算每个学科低于平均值的人数
    _failure(a,N,failure);       //计算每个学科不及格的人数
    _good(a,N,good);            //计算每个学科优秀的人数
    printf("the GPA of every student:\n");
    for(i=0;i<N;i++)
        printf("%d  %lf\n",i+1,*(stu+i));           //输出每个学生的平均值
    printf("\n\nthe average grade of each subject:\n");
    for(j=0;j<M;j++)
        printf("%s  %lf\n",subname+j,*(sub+j));       //输出每个学科的平均值
    printf("\n\nthe number of the students under average grade of each subject:\n");
    for(j=0;j<M;j++)
        printf("%s  %d\n",subname+j,*(sub_avr+j)); //输出每个学科低于平均值
    的人数
    printf("\n\nthe number of the failure of each subject:\n");
    for(j=0;j<M;j++)
        printf("%s  %d\n",subname+j,*(failure+j)); //输出每个学科不及格的人数
    printf("\n\nthe number of excelent students of each subject:\n");
```

```

    for(j=0;j<M;j++)
        printf("%s  %d\n",subname+j,*(good+j));    //输出每个学科优秀的人数
    return 0;
}

void _stu_ave(int (*a)[M],int n,double *p){
//计算每个学生的平均值的函数 (p指向保存学生平均值的数组)
    int i,j,sum_stu=0;
    for(i=0;i<n;i++){
        sum_stu=0;
        for(j=0;j<M;j++){
            sum_stu+=*(a+i+j);
        }
        *(p+i)=sum_stu/(float)M;
    }
}

void _sub_ave(int (*a)[M],int n,double *p){
//计算每个学科的平均值的函数 (p指向保存学科平均值的数组)
    int i,j,sum_sub=0;
    for(j=0;j<M;j++){
        sum_sub=0;
        for(i=0;i<n;i++){
            sum_sub+=*(a+i+j);
        }
        *(p+j)=sum_sub/(float)n;
    }
}

void _lower_ave(int (*a)[M],int n,int *p,double *q){
//计算每个学科低于平均值的人数的函数 (p指向保存每个学科低于平均分人数
//的数组, q指向学生平均分的数组)
    int j,i;
    for(j=0;j<M;j++){
        *(p+j)=0;
        for(i=0;i<n;i++){
            if(*(a+i+j)<*(q+j))(*(p+j))++;
        }
    }
}

void _failure(int (*a)[M],int n,int *p){
//计算每个学科不及格的人数的函数 (p指向保存学科不及格人数的数组)
    int j,i;
    for(j=0;j<n;j++){
        *(p+j)=0;
        for(i=0;i<n;i++){
            if(*(a+i+j)<60)*(p+j)++;
        }
    }
}

```

```

    }
}
}
void _good(int (*a)[M],int n,int *p){
//计算每个学科优秀的人数的函数（p指向保存学科优秀人数的数组）
    int j,i;
    for(j=0;j<n;j++){
        *(p+j)=0;
        for(i=0;i<n;i++){
            if(*(a+i)+j>89)*(p+j)++;
        }
    }
}
}

```

(3) 测试。

(a) 测试数据。

math

chinese

Cprogram

English

computing

physics

98 96 87 99 87 100

100 98 70 75 80 88

67 68 58 48 29 60

82 87 84 80 70 59

78 76 75 80 54 40

79 87 37 78 50 80

89 86 87 60 58 70

(c) 测试结果如图 6-7 所示。

```

the GPA of every student:
1  94.500000
2  85.166664
3  55.000000
4  77.000000
5  67.166664
6  68.500000
7  75.000000

the average grade of each subject:
math  84.714287
chinese  85.428574
Cprogram  71.142860
English  74.285713
computing  61.142857
physics  71.000000

the number of the students under average grade of each subject:
math  4
chinese  2
Cprogram  3
English  2
computing  4
physics  4

the number of the failure of each subject:
math  0
chinese  0
Cprogram  2
English  1
computing  4
physics  2

the number of excelent students of each subject:
math  2
chinese  2
Cprogram  0
English  1
computing  0
physics  1

```

图 6-7 程序设计题（4）程序运行结果截图



### 6.2.5 选做题程序设计

(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) 算法流程解释如下。

- ① 将两个有  $N$  位整数和  $M$  为小数的数据用字符串形式保存
- ② 从最低位开始, 依次将两个字符串中对应位置的字符取出, 转化为整型, 相加, 取各位, 若大于十则向前进一位, 依次从低位向高位操作, 每次操作结果存放在一个字符数组中的对应位置
- ③ 若最高位需要进位则输出一个 1
- ④ 输出得到的新的字符串
- ⑤ 结束

(2) 源程序清单。

```
#include <stdio.h>
#include <stdlib.h>
#define N 20
#define M 10

int main(int argc, char *argv[]) {
    char x[N+M+2], y[N+M+2], z[N+M+2];
    z[N+M+1] = '\0';
    z[N] = '.';
    gets(x);
    gets(y);
    int cnt = N+M, key = 0;
    while(cnt >= 0) {
        if(cnt == N) {
            cnt--;
            continue;
        }
        int a = (int)(x[cnt] - '0'), b = (int)(y[cnt] - '0'); //将字符转换为整型数据
        z[cnt] = (char)((a+b+key) % 10 + '0'); //将相加结果转化为字符保存
        if(a+b+key >= 10) key = 1; //有进位的情况
        else key = 0;
        cnt--;
    }
}
```

```

if ((int)(x[0]-'0')+(int)(y[0]-'0')+key>=10)putchar('1');//最高位如果有进位补1
puts(z);
return 0;
}

```

(3) 测试。

(a) 测试数据。

```

12345678912345678912.1234567891
98765432109876543210.0123456789
999999999999999999.9999999999
999999999999999999.9999999999
89243423742909327874.2843626743
92832909096432789873.4387974369

```

(b) 测试结果如图 6-8 所示。

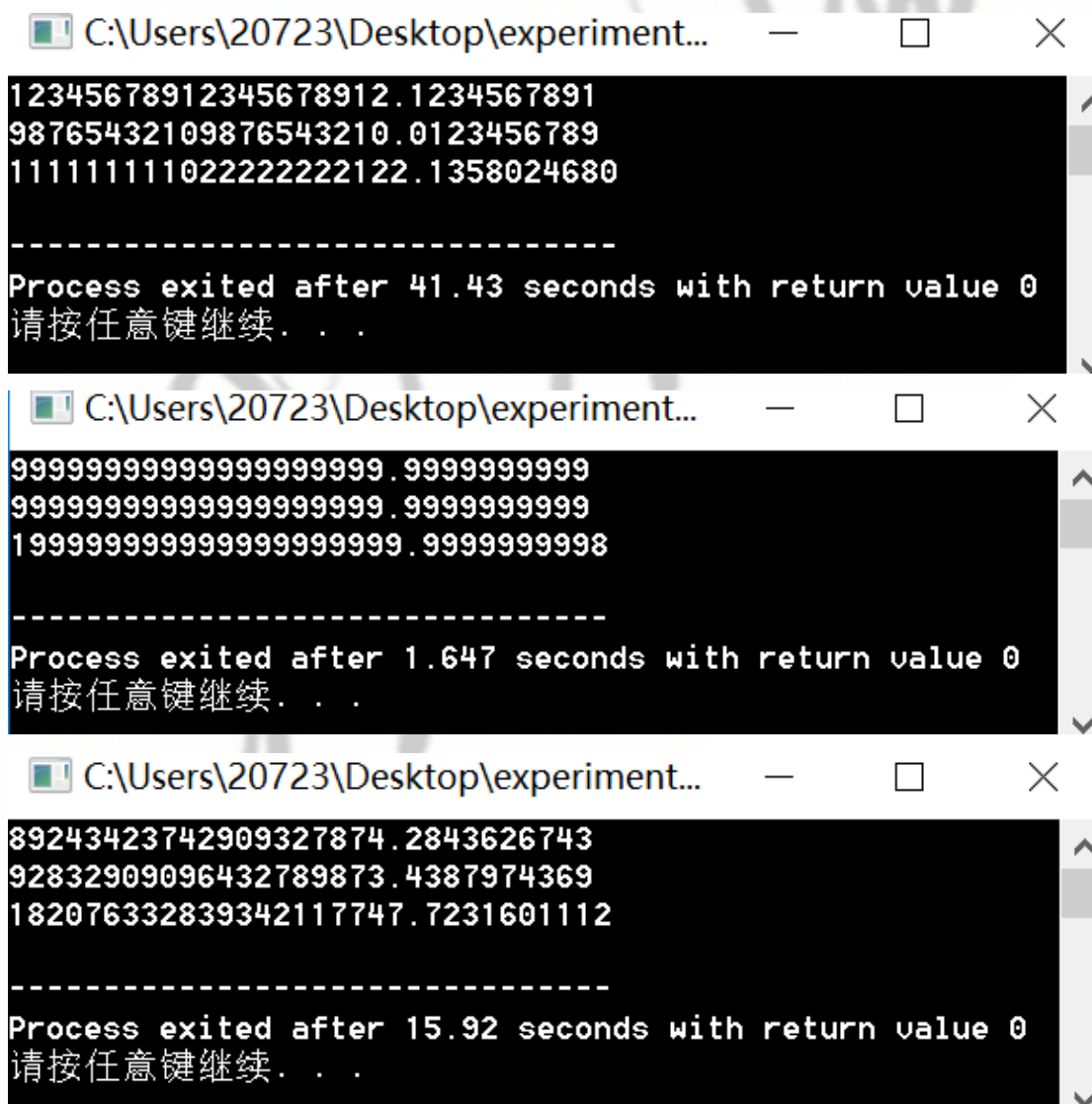


图 6-8 选做程序设计题（1）程序运行结果截图

（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）算法流程解释如下。

- ①将上述复杂声明（指针函数指针数组）分别指向两个返回值和接受参数类型的函数
- ②输入两个字符串
- ③用上述指针函数指针数组中的两个指针分别调用所指向的函数
- ④输出两个函数的返回结果
- ⑤结束

（2）源程序清单。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
    char *(*p[2])(const char *, const char *);
    p[0]=strstr;
    p[1]=strcat;
    char s[50],t[50];
    gets(s);
    gets(t);
    printf("%s\n",p[0](s,t));
    printf("%s",p[1](s,t));
    return 0;
}
```

（3）测试。

（a）测试数据。

```
fdsj' ;...';.;.;'*(^%^dsfjklfskklbfd
dsf
```

（b）测试结果如图 6-9 所示。

```
fdsj'';..'';;;;*(&%^dsfjklfskklfsd
dsf
dsfjklfskklfsd
fdsj'';..'';;;;*(&%^dsfjklfskklfsd
-----
Process exited after 19.91 seconds with return value 0
请按任意键继续. . .
```

图 6-9 选做程序设计题 (2) 程序运行结果截图

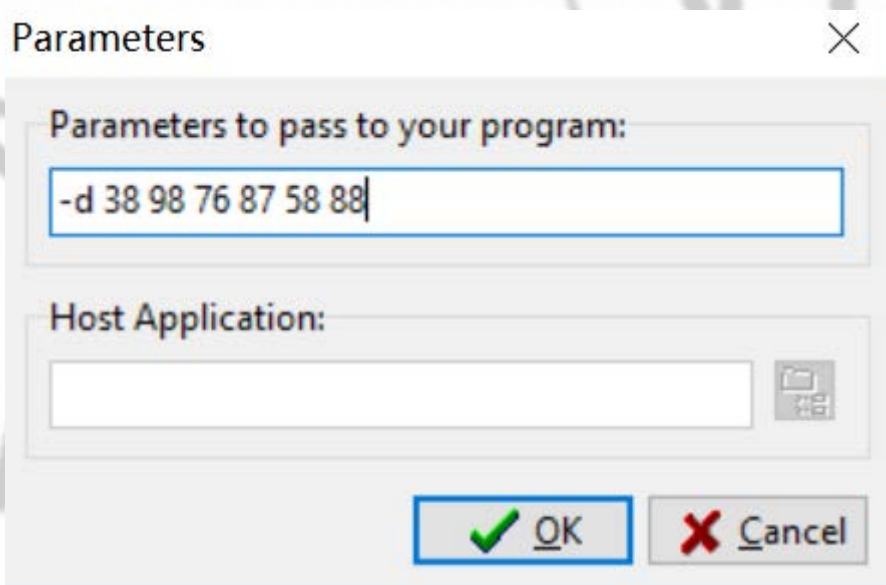
### (3) 指定 main 函数的参数

选择“project/ set programs’ arguments…”菜单命令，即可打开图 2.12 所示的对话框，在“Program arguments”文本框中输入 main 函数的参数。注意只输入命令行中文件名后的参数，文件名不输入。

解答:

### 测试（以程序设计题 3 为例进行演示）

(a) 测试所指定的参数如图 6-10 所示。



(b) 测试结果如图 6-11 所示。

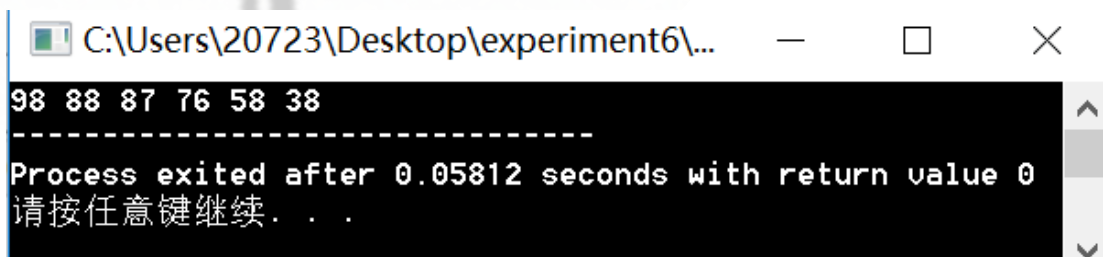


图 6-11 选做程序设计题 (3) 程序运行结果截图



### 6.3 实验小结

1. 压缩空格字符那一题按照要求，应用指针操作，我简单的声明了字符指针数组，但没有声明字符的二维数组，因此在调用 `gets` 函数时程序会崩溃，后才想起未对指针进行初始化，还必须声明一个字符二维数组才能使用。这让我明白，数组名本质上是一个指针，但指针不能直接当数组名用，还需要通过初始化使其指向一个数组。
2. 压缩空格字符时，刚开始遗忘了老师上课所讲的通过两个变量，一个读一个写的方法，用了一种效率比较低效的方法，即每次读完第一个空格后每有  $x$  个空格便向前移  $n$  位。后在老师的提醒下想到这个方法，并顺利的用 C 语言进行了实现。
3. 利用 `main` 函数参数进行排序的过程中，不知道 `main` 函数的第一个参数永远是文件名，导致不能完全正常运行，后翻书后发现这一问题，并顺利解决。
4. 将 `main` 函数的第二个参数与 `-d` 进行比较时，想当然的直接拿字符数组名与字符串进行比较，后意识到字符数组名只是个指针，需要将字符数组中的每一元素一一进行比较。



## 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 实验内容

#### 7.2.1 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, 'A', u, {100, 'B', 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	<code>(++p)-&gt;x</code>	100	100
2	<code>p++, p-&gt;c</code>	'B'	'B'
3	<code>*p++-&gt;t, *p-&gt;t</code>	'x'	'x'
4	<code>* (++p)-&gt;t</code>	'x'	'x'
5	<code>*++p-&gt;t</code>	'v'	'v'
6	<code>++*p-&gt;t</code>	'v'	'v'

编程验算结果如图7-1所示。

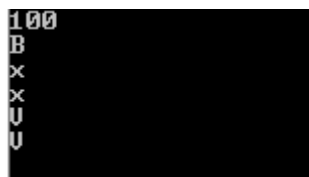


图7-1 程序验证题运行结果截图

### 7.2.2 源程序完善、修改、替换

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，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) 完善后的源程序如下：

```

#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 free_list(struct s_list *head);

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");
    free(head);
}

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 */
    }
}

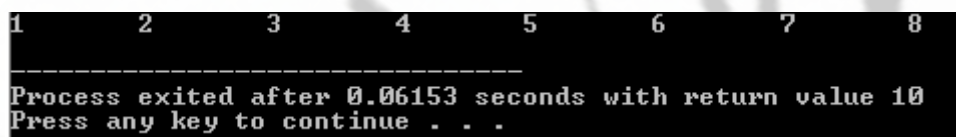
```

```

        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 */
}
void free_list(struct s_list *head){
    struct s_list *p;
    do{
        p=head->next;
        free(head);
        head=p;
    }while(p);
}

```

2) 完善修改后的程序运行结果如图 7-2 所示。



```

1      2      3      4      5      6      7      8
-----
Process exited after 0.06153 seconds with return value 10
Press any key to continue . . .

```

图7-2 程序完善修改调试题（1）程序运行结果

(2) 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。

解答：

1) 完善后的源程序如下：

```

#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 free_list(struct s_list *head);

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");
free_list(head);
}

void create_list(struct s_list **headp,int *p)
{
    struct s_list *tail=NULL,*loc_head;
    if(p[0]==0) /* 相当于*p==0 */
        ;
    else { /* loc_head指向动态分配的第一个结点 */
        tail=(struct s_list *)malloc(sizeof(struct s_list));
        tail->next=NULL;
        tail->data=*p++; /* 对数据域赋值 */
        loc_head=tail; /* 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 */
            loc_head=(struct s_list *)malloc(sizeof(struct s_list));
            loc_head->next=tail; /* tail指向新创建的结点 */
            loc_head->data=*p++; /* 向新创建的结点的数据域赋值 */
            tail=loc_head;
        }
    }
    *headp=loc_head; /* 使头指针headp指向新创建的链表 */
}

void free_list(struct s_list *head){
    struct s_list *p;
    do{
        p=head->next;
        free(head);
        head=p;
    }while(p);
}

```

2) 完善后的程序运行结果如图 7-3 所示。



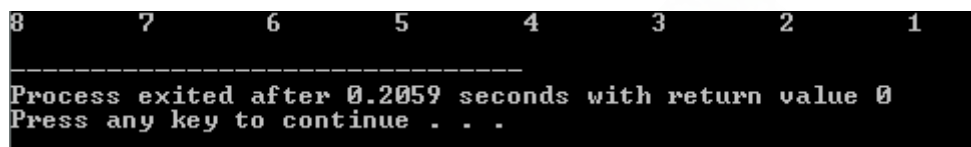


图7-3 程序完善修改调试题（2）程序运行结果

### 7.2.3 程序设计题

1. 设计一个字段结构 struct bits，它将一个 8 位无符号字节从最低位向最高位声明为 8 个字段，各字段依次为 bit0, bit1, ..., bit7, 且 bit0 的优先级最高。同时设计 8 个函数，第 i 个函数以 biti (i=0, 1, 2, ..., 7) 为参数，并且在函数体内输出 biti 的值。将 8 个函数的名字存入一个函数指针数组 p\_fun。如果 bit0 为 1，调用 p\_fun[0] 指向的函数。如果 struct bits 中有多位为 1，则根据优先级从高到低依次调用函数指针数组 p\_fun 中相应元素指向的函数。8 个函数中的第 0 个函数可以设计为：

```
void f0(unsigned short b)
{
    Printf("the function %d is called!\n", b);
}
```

**解答：**

(1) 算法流程解释如下。

- ① 开始
- ② 定义结构 bit，包含 8 个一个二进制位的的字段
- ③ 定义联合结构，一个字符和一个 bit 结构
- ④ 输入一个字符，保存到联合结构中的字符
- ⑤ 利用联合结构依次检查每个字符的每个二进制位是否为 1，若是，则调用相应函数
- ⑥ 结束

(2) 源程序清单。

```
#include <stdio.h>
#include <stdlib.h>

struct bits{                //字段结构
    unsigned short bit0:1;
    unsigned short bit1:1;
```

```

    unsigned short bit2:1;
    unsigned short bit3:1;
    unsigned short bit4:1;
    unsigned short bit5:1;
    unsigned short bit6:1;
    unsigned short bit7:1;
};

union a{
    struct bits b;
    char c;
};

void f0(unsigned short b);
void f1(unsigned short b);
void f2(unsigned short b);
void f3(unsigned short b);
void f4(unsigned short b);
void f5(unsigned short b);
void f6(unsigned short b);
void f7(unsigned short b);

int main(int argc, char *argv[]) {
    union a ch;
    scanf("%c",&ch.c);
    if(ch.b.bit0==1)f0(ch.b.bit0);
    if(ch.b.bit1==1)f1(ch.b.bit1);
    if(ch.b.bit2==1)f2(ch.b.bit2);
    if(ch.b.bit3==1)f3(ch.b.bit3);
    if(ch.b.bit4==1)f4(ch.b.bit4);
    if(ch.b.bit5==1)f5(ch.b.bit5);
    if(ch.b.bit6==1)f6(ch.b.bit6);
    if(ch.b.bit7==1)f7(ch.b.bit7);
    return 0;
}

void f0(unsigned short b){
    printf("the function 0 is called:%d(bit0)\n",b);
}
void f1(unsigned short b){
    printf("the function 1 is called:%d(bit1)\n",b);
}
void f2(unsigned short b){
    printf("the function 2 is called:%d(bit2)\n",b);
}

```

```

}
void f3(unsigned short b){
    printf("the function 3 is called:%d(bit3)\n",b);
}
void f4(unsigned short b){
    printf("the function 4 is called:%d(bit4)\n",b);
}
void f5(unsigned short b){
    printf("the function 5 is called:%d(bit5)\n",b);
}
void f6(unsigned short b){
    printf("the function 6 is called:%d(bit6)\n",b);
}
void f7(unsigned short b){
    printf("the function 7 is called:%d(bit7)\n",b);
}

```

(3) 测试。

(a) 测试数据。

a

9

<

+

(回车)

(b) 测试结果如图 7-4 所示。

```

a
the function 0 is called:1(bit0)
the function 5 is called:1(bit5)
the function 6 is called:1(bit6)

```

```

9
the function 0 is called:1(bit0)
the function 3 is called:1(bit3)
the function 4 is called:1(bit4)
the function 5 is called:1(bit5)

```

```

<
the function 2 is called:1(bit2)
the function 3 is called:1(bit3)
the function 4 is called:1(bit4)
the function 5 is called:1(bit5)

```

```

+
the function 0 is called:1(bit0)
the function 1 is called:1(bit1)
the function 3 is called:1(bit3)
the function 5 is called:1(bit5)

```

```
the function 1 is called:1(bit1)
the function 3 is called:1(bit3)
```

图 7-4 程序设计题 1 运行结果截图

2. 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。用函数编程实现下列功能：

- (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。

**解答：**

(1) 算法流程解释如下。

- ① 开始
- ② 动态创建链表（成绩单），遇文件尾创建过程结束，给尾元素赋 NULL
- ③ 根据每个结构变量的指针域显示链表中所有学生的成绩信息
- ④ 利用字符串比较和菜单选择方法更改指定学生指定科目的成绩
- ⑤ 再次利用②中方法显示链表中所有数据
- ⑥ 结束

(2) 源程序清单。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct grade{
    char number[11];
    char name[30];
    int e;           //English
    int m;           //Math
    int p;           //physics
    int c;           //C programming
    int s;           //sum
    double a;        //average
    struct grade *next;
}grade;
```



```

void creat_list(grade **headp);           //创建链表
void list_change(char *s,grade *head,int option,int result);//修改连表中特定元素值
void display_a(grade *headp);           //显示链表中的所有数据
void free_list(grade *head);             //释放链表所占用的内存单元

int main(int argc, char *argv[]) {
    grade *head;
    creat_list(&head);
    printf("\n");
    display_a(head);
    char change[11];
    int opt,r;           //opt选择更改的科目，r为目标分数
    printf("\ninput the number of the student and the subject and object
grade\n(1.English 2.math 3.physics 4.Cprogramming)\n");
    gets(change);
    scanf("%d %d",&opt,&r);
    list_change(change,head,opt,r);
    printf("\n");
    display_a(head);
    free_list(head);
    return 0;
}

void creat_list(grade **headp){ //动态创建链表，遇文件尾过程结束
    grade *h,*t;
    t=h=(grade *)malloc(sizeof(grade));
    t->next=NULL;
    scanf("%s %s %d %d %d %d",h->number,h->name,&h->e,&h->m,&h->p,&h->c
);
    t->s=t->e+t->m+t->p+t->c;
    t->a=(t->e+t->m+t->p+t->c)/4.0;
    getchar();
    char c;
    while((c=getchar())!=EOF){
        t->next=(grade *)malloc(sizeof(grade));
        t=t->next;
        t->next=NULL;    //保证尾元素始终为空指针
        t->number[0]=c;    //用于判断是否为文件尾的字符c还给字符串首字符

        scanf("%s %s %d %d %d %d",&(t->number[1]),t->name,&t->e,&t->m,&t->p,&t
->c);
        t->s=t->e+t->m+t->p+t->c;    //计算该学生的总成绩
        t->a=(t->e+t->m+t->p+t->c)/4.0; //计算该学生的平均成绩
        getchar();    //吸收换行符
    }
}

```



```

    *headp=h;
}
void list_change(char *s,grade *head,int option,int result){ //更改链表特定元素
    int haschange=0;        //用于判断是否存在该学生
    grade *p;
    p=head;
    while(p){
        if(strcmp(s,p->number)==0){
            switch (option){ //科目选择菜单
                case 1:p->e=result;break;
                case 2:p->m=result;break;
                case 3:p->p=result;break;
                case 4:p->c=result;break;
                default :printf("ERROR!\n");//若不存在该学科，报错
            }
            haschange=1;
            p->s=p->e+p->m+p->p+p->c;
            p->a=(p->e+p->m+p->p+p->c)/4.0;
            break;
        }
        p=p->next;
    }
    if(!haschange)printf("Not found the student!\n");//若不存在该学生，报错
}
void display_a(grade *headp){
    grade *p;
    p=headp;
    while(p){

        printf("%s %s %d %d %d %d %.2lf\n",p->number,p->name,p->e,p->m,p->p,
p->c,p->s,p->a);
        p=p->next;
    }
}
void free_list(grade *head){ //释放内存单元
    grade *p;
    do{
        p=head->next;
        free(head);
        head=p;
    }while(p);
}

```

(3) 测试。

(a) 测试数据。

```

U201814001 Anna 90 87 85 89
U201814010 Tim 97 80 79 85
U201814100 Beta 87 68 76 72
U201814050 Jim 100 78 59 69
U201814999 Linda 88 76 65 34
U201814666 Bob 100 90 45 20 (Ctrl+Z 结束数据输入)
U201814666 (被修改的学生)
4 75 (修改的科目以及目标成绩)
//以下为第二组
U201814999 huazhongkeda 100 100 100 100 (Ctrl+Z 结束数据输入)
U201814777 (被修改的学生, 但数据中无此学生信息, 报错)
4 100
//以下为第三组
U201814999 huazhongkeda 100 100 100 100 (Ctrl+Z 结束数据输入)
U201814999 (被修改的学生)
5 100 (修改的科目以及目标成绩, 但不存在编号为 5 的学科, 报错)

```

(b) 测试结果如图 7-5 所示。

```

U201814001 Anna 90 87 85 89
U201814010 Tim 97 80 79 85
U201814100 Beta 87 68 76 72
U201814050 Jim 100 78 59 69
U201814999 Linda 88 76 65 34
U201814666 Bob 100 90 45 20
^Z

U201814001 Anna 90 87 85 89 351 87.75
U201814010 Tim 97 80 79 85 341 85.25
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814999 Linda 88 76 65 34 263 65.75
U201814666 Bob 100 90 45 20 255 63.75

input the number of the student and the subject and object grade
<1.English 2.math 3.physics 4.Cprogramming>
U201814666
4 75

U201814001 Anna 90 87 85 89 351 87.75
U201814010 Tim 97 80 79 85 341 85.25
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814999 Linda 88 76 65 34 263 65.75
U201814666 Bob 100 90 45 75 310 77.50

-----
Process exited after 202.1 seconds with return value 0
Press any key to continue . . .

```

```

U201814999 huazhongkeda 100 100 100 100
^Z
U201814999 huazhongkeda 100 100 100 100 400 100.00
input the number of the student and the subject and object grade
<1.English 2.math 3.physics 4.Cprogramming>
U201814777
4 100
Not found the student!
U201814999 huazhongkeda 100 100 100 100 400 100.00
-----
Process exited after 34.69 seconds with return value 0
Press any key to continue . . .

U201814999 huazhongkeda 100 100 100 100
^Z
U201814999 huazhongkeda 100 100 100 100 400 100.00
input the number of the student and the subject and object grade
<1.English 2.math 3.physics 4.Cprogramming>
U201814999
5 100
ERROR!
U201814999 huazhongkeda 100 100 100 100 400 100.00
-----
Process exited after 21.98 seconds with return value 0
Press any key to continue . . .

```

图 7-5 程序设计题 2 程序运行结果截图

## 7.2.4 选做题程序设计

1. 对编程设计题第（2）题的程序，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。

**解答：**

(1) 算法流程解释如下。

- ① 开始
- ② 动态创建链表（成绩单），遇文件尾创建过程结束，给尾元素赋 NULL
- ③ 利用自引用结构动态创建链表
- ④ 输出排序前链表中所有数据
- ⑤ 利用冒泡算法，每一轮中若前者比后者大，依次交换两个结构变量中除 next 指针外的所有成员
- ⑥ 输出排序后链表中所有数据
- ⑦ 结束

(2) 源程序清单。

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>

typedef struct grade{
    char number[11];
    char name[30];
    int e;           //English
    int m;           //Math
    int p;           //physics
    int c;           //C programming
    int s;           //sum
    double a;        //average
    struct grade *next;
}grade;

void creat_list(grade **);
void display(grade *);
void bubble_sort_list_data(grade *);
void swap(int *,int *);
void strchange(char *a,char *b);
void free_list(grade *head);

int main(int argc, char *argv[]) {
    grade *head;
    creat_list(&head);           //创建链表
    printf("\n");
    display(head);              //输出链表中所有数据
    bubble_sort_list_data(head); //冒泡排序
    printf("\n");
    display(head);              //输出链表中所有数据
    free_list(head);            //释放链表所占用的内存单元
    return 0;
}

void creat_list(grade **headp){
    grade *h,*t;
    t=h=(grade *)malloc(sizeof(grade));
    t->next=NULL;
    scanf("%s %s %d %d %d %d",h->number,h->name,&h->e,&h->m,&h->p,&h->c);
    h->s=h->e+h->m+h->p+h->c;      //计算总分
    h->a=(h->e+h->m+h->p+h->c)/4.0; //计算平均分
    getchar();
    char c;
    while((c=getchar())!=EOF){ //判断学生成绩信息输入是否结束
```



```

        t->next=(grade *)malloc(sizeof(grade));
        t=t->next;
        t->next=NULL;
        t->number[0]=c;    //将用于判断文件尾的字符归还给学号字符串
        scanf("%s %s %d %d %d %d",&(t->number[1]),t->name,&t->e,&t->m,&t->p,
&t->c);
        t->s=t->e+t->m+t->p+t->c;
        t->a=(t->e+t->m+t->p+t->c)/4.0;
        getchar();
    }
    *headp=h;
}
void display(grade *headp){
    grade *p;
    p=headp;
    while(p){

        printf("%s %s %d %d %d %d %.2lf\n",p->number,p->name,p->e,p->m,p->p,
p->c,p->s,p->a);
        p=p->next;
    }
}
void bubble_sort_list_data(grade *head){
    int length=0,i,j;
    grade *p=head;
    while(p){
        length++;           //计算链表中结构体个数，用于控制冒泡次数
        p=p->next;
    }
    for(i=0;i<length-1;i++){
        p=head;
        for(j=0;j<length-i-1;j++){
            if(p->s > p->next->s){
                strchange(p->number,p->next->number);//依次交换所有学生信息
                strchange(p->name,p->next->name);
                swap(&p->e,&p->next->e);
                swap(&p->m,&p->next->m);
                swap(&p->p,&p->next->p);
                swap(&p->c,&p->next->c);
                swap(&p->s,&p->next->s);
                double temp=p->a;p->a=p->next->a;p->next->a=temp;
            }
            p=p->next;
        }
    }
}

```



```

    }
}
void swap(int *a,int *b){
    int t=*a;
    *a=*b;
    *b=t;
}
void strchange(char *a,char *b){//字符串交换函数
    char c[30];
    strcpy(c,a);
    strcpy(a,b);
    strcpy(b,c);
}
void free_list(grade *head){
    grade *p;
    do{
        p=head->next;
        free(head);
        head=p;
    }while(p);
}

```

(3) 测试。

(a) 测试数据。

```

U201814001 Anna 90 87 85 89
U201814010 Tim 97 80 79 85
U201814100 Beta 87 68 76 72
U201814050 Jim 100 78 59 69
U201814999 Linda 88 76 65 34
U201814666 Bob 100 90 45 20 (Ctrl+Z 结束数据输入)

```

(b) 测试结果如图 7-6 所示。

```

U201814001 Anna 90 87 85 89
U201814010 TIm 97 80 79 85
U201814100 Beta 87 68 76 72
U201814050 Jim 100 78 59 69
U201814999 Linda 88 76 65 34
U201814666 Bob 100 90 45 20
^Z

U201814001 Anna 90 87 85 89 351 87.75
U201814010 TIm 97 80 79 85 341 85.25
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814999 Linda 88 76 65 34 263 65.75
U201814666 Bob 100 90 45 20 255 63.75

U201814666 Bob 100 90 45 20 255 63.75
U201814999 Linda 88 76 65 34 263 65.75
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814010 TIm 97 80 79 85 341 85.25
U201814001 Anna 90 87 85 89 351 87.75

-----
Process exited after 6.41 seconds with return value 0
请按任意键继续. . .

```

图 7-6 选做程序设计题 1 程序运行结果截图

2. 对选做题第（1）题，进一步写出用交换结点指针域的方法升序排序的函数。

解答：

(1) 算法流程解释如下。

- ① 开始
- ② 自引用结构动态创建链表，遇文件尾创建过程结束
- ③ 输出所有结点数据
- ④ 通过交换指针域冒泡排序（头两个结点特殊处理）
- ⑤ 输出排序后所有结点数据
- ⑥ 结束

(2) 源程序清单。

```

#include <stdio.h>
#include <stdlib.h>

```

```

typedef struct grade{
    char number[11];
    char name[30];
    int e;           //English
    int m;           //Math
    int p;           //physics
    int c;           //C programming
    int s;           //sum
    double a;        //average
    struct grade *next;
}grade;

void creat_list(grade **headp);
void display(grade *headp);
void bubble_sort_list_pointer(grade **head);
void free_list(grade *head);

int main(int argc, char *argv[]) {
    grade *head;
    creat_list(&head);           //创建链表
    printf("\n");
    display(head);
    bubble_sort_list_pointer(&head); //通过交换指针域冒泡排序
    printf("\n");
    display(head);
    free_list(head);
    return 0;
}

void creat_list(grade **headp){ //创建链表
    grade *h,*t;
    t=h=(grade *)malloc(sizeof(grade));
    t->next=NULL;
    scanf("%s %s %d %d %d %d",h->number,h->name,&h->e,&h->m,&h->p,&h->c
);
    h->s=h->e+h->m+h->p+h->c;
    h->a=(h->e+h->m+h->p+h->c)/4.0;
    getchar();
    char c;
    while((c=getchar())!=EOF){
        t->next=(grade *)malloc(sizeof(grade));
        t=t->next;
        t->next=NULL;
        t->number[0]=c;
    }
}

```

```

scanf("%s %s %d %d %d %d",&(t->number[1]),t->name,&t->e,&t->m,&t->p,&t
->c);
    t->s=t->e+t->m+t->p+t->c;
    t->a=(t->e+t->m+t->p+t->c)/4.0;
    getchar();
}
*headp=h;
}
void display(grade *headp){
    grade *p;
    p=headp;
    while(p){

        printf("%s %s %d %d %d %d %.2lf\n",p->number,p->name,p->e,p->m,p->p,
p->c,p->s,p->a);
        p=p->next;
    }
}
void bubble_sort_list_pointer(grade **head){
    int length=0,i,j;
    grade *p=*head,*temp,*t;
    while(p){
        length++;
        p=p->next;
    }
    p=*head;
    for(i=0;i<length-1;i++){
        p=*head;
        if(p->s > p->next->s){
            temp=p->next->next;
            p->next->next=*head;
            *head=p->next;
            p->next=temp;
        }
        p=*head;
        t=p->next;
        for(j=0;j<length-i-2;j++){
            if(t->s>t->next->s){
                temp=t->next->next;
                t->next->next=p->next;
                p->next=t->next;
                t->next=temp;
            }
        }
    }
}

```

//计算链表中结点个数，控制冒泡次数

//交换头两个结点的指针域需特殊对待

//交换结点指针域

```

        p=p->next;
        t=p->next;
    }
}
}
void free_list(grade *head){
    grade *p;
    do{
        p=head->next;
        free(head);
        head=p;
    }while(p);
}

```

(3) 测试。

(a) 测试数据。

U201814001 Anna 90 87 85 89

U201814010 Tim 97 80 79 85

U201814100 Beta 87 68 76 72

U201814050 Jim 100 78 59 69

U201814999 Linda 88 76 65 34

U201814666 Bob 100 90 45 20 (Ctrl+Z 结束数据输入)

(b) 测试结果如图 7-7 所示。



```

U201814001 Anna 90 87 85 89
U201814010 Tim 97 80 79 85
U201814100 Beta 87 68 76 72
U201814050 Jim 100 78 59 69
U201814999 Linda 88 76 65 34
U201814666 Bob 100 90 45 20
^Z

U201814001 Anna 90 87 85 89 351 87.75
U201814010 Tim 97 80 79 85 341 85.25
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814999 Linda 88 76 65 34 263 65.75
U201814666 Bob 100 90 45 20 255 63.75

U201814666 Bob 100 90 45 20 255 63.75
U201814999 Linda 88 76 65 34 263 65.75
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814010 Tim 97 80 79 85 341 85.25
U201814001 Anna 90 87 85 89 351 87.75

-----
Process exited after 2.071 seconds with return value 0
请按任意键继续. . .

```

图 7-7 选做程序设计题 2 程序运行结果截图

3. 采用双向链表重做编程设计题第（2）题。

解答：

(1) 算法流程解释如下。

- ① 开始
- ② 动态创建双向链表（成绩单），遇文件尾创建过程结束，给尾元素赋 NULL
- ③ 根据每个结构变量的指针域顺序显示链表中所有学生的成绩信息
- ④ 利用字符串比较和菜单选择方法更改指定学生指定科目的成绩
- ⑤ 再次利用②中方法顺序显示链表中所有数据
- ⑥ 利用每个结点的 back 指针从尾结点逆序输出
- ⑦ 结束

(2) 源程序清单。

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>

typedef struct grade{
    char number[11];
    char name[30];
    int e;           //English
    int m;           //Math
    int p;           //physics
    int c;           //C programming
    int s;           //sum
    double a;        //average
    struct grade *next; //指向下一个结点
    struct grade *back; //指向上一个结点
}grade;

void creat_list(grade **headp);
void list_change(char *s,grade *head,int option,int result);
void display_a(grade *headp);
void display_reverse(grade *headp);
void free_list(grade *head);

int main(int argc, char *argv[]) {
    grade *head;
    creat_list(&head);           //创建双向链表
    printf("\n");
    display_a(head);             //顺序输出所有结点数据
    char change[11];
    int opt,r;                   //控制更改的科目和目标分数
    printf("\ninput the number of the student and the subject and object
grade\n(1.English 2.math 3.physics 4.Cprogramming)\n");
    gets(change);                //所更改的学生的学号
    scanf("%d %d",&opt,&r);
    list_change(change,head,opt,r);
    printf("\n");
    display_a(head);             //顺序输出所有结点数据
    printf("\n");
    display_reverse(head);       //通过双向链表的back指针逆序输出所有结点数据
    free_list(head);
    return 0;
}

void creat_list(grade **headp){
    grade *h,*t;
    t=h=(grade *)malloc(sizeof(grade));
```

```

t->back=NULL;           //第一个结点的back结点始终为NULL作为结束标记
t->next=NULL;
scanf("%s %s %d %d %d %d",h->number,h->name,&h->e,&h->m,&h->p,&h->c
);
t->s=t->e+t->m+t->p+t->c;
t->a=(t->e+t->m+t->p+t->c)/4.0;
getchar();
char c;
while((c=getchar())!=EOF){
    t->next=(grade *)malloc(sizeof(grade));
    t->next->back=t;      //每次创建的结点的back指针指向上一个结点
    t=t->next;
    t->next=NULL;       //最新创建的结点指向NULL
    t->number[0]=c;

    scanf("%s %s %d %d %d %d",&(t->number[1]),t->name,&t->e,&t->m,&t->p,&t
->c);
    t->s=t->e+t->m+t->p+t->c;
    t->a=(t->e+t->m+t->p+t->c)/4.0;
    getchar();
}
*headp=h;
}
void list_change(char *s,grade *head,int option,int result){
    int haschange=0;      //haschange检查是否找到了目标学生信息
    grade *p;
    p=head;
    while(p){
        if(strcmp(s,p->number)==0){
            switch (option){          //op控制更改的学科
                case 1:p->e=result;break;
                case 2:p->m=result;break;
                case 3:p->p=result;break;
                case 4:p->c=result;break;
                default :printf("ERROR!\n");
            }
            haschange=1;
            p->s=p->e+p->m+p->p+p->c;
            p->a=(p->e+p->m+p->p+p->c)/4.0;
            break;
        }
        p=p->next;
    }
    if(!haschange)printf("Not found the student!\n");
}

```

```

}
void display_a(grade *headp){
    grade *p;
    p=headp;
    while(p){

        printf("%s %s %d %d %d %d %d %.2lf\n",p->number,p->name,p->e,p->m,p->p,
p->c,p->s,p->a);
        p=p->next;
    }
}
void display_reverse(grade *headp){
    grade *p;
    p=headp;
    while(p->next)p=p->next;
    while(p){

        printf("%s %s %d %d %d %d %d %.2lf\n",p->number,p->name,p->e,p->m,p->p,
p->c,p->s,p->a);
        p=p->back;
    }
}
void free_list(grade *head){
    grade *p;
    do{
        p=head->next;
        free(head);
        head=p;
    }while(p);
}

```

(3) 测试。

(a) 测试数据。

U201814001 Anna 90 87 85 89

U201814010 Tim 97 80 79 85

U201814100 Beta 87 68 76 72

U201814050 Jim 100 78 59 69

U201814999 Linda 88 76 65 34

U201814666 Bob 100 90 45 20 (Ctrl+Z 结束数据输入)

U201814666 (被修改的学生)

4 75 (修改的科目以及目标成绩)



//以下为第二组

U201814999 huazhongkeda 100 100 100 100 (Ctrl+Z 结束数据输入)

U201814777 (被修改的学生, 但数据中无此学生信息, 报错)

4 100

//以下为第三组

U201814999 huazhongkeda 100 100 100 100 (Ctrl+Z 结束数据输入)

U201814999 (被修改的学生)

5 100 (修改的科目以及目标成绩, 但不存在编号为 5 的学科, 报错)

(b) 测试结果如图 7-8 所示。

```

U201814001 Anna 90 87 85 89
U201814010 Tim 97 80 79 85
U201814100 Beta 87 68 76 72
U201814050 Jim 100 78 59 69
U201814999 Linda 88 76 65 34
U201814666 Bob 100 90 45 20
^Z

U201814001 Anna 90 87 85 89 351 87.75
U201814010 Tim 97 80 79 85 341 85.25
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814999 Linda 88 76 65 34 263 65.75
U201814666 Bob 100 90 45 20 255 63.75

input the number of the student and the subject and object grade
<1.English 2.math 3.physics 4.Cprogramming>
U201814666
4 75

U201814001 Anna 90 87 85 89 351 87.75
U201814010 Tim 97 80 79 85 341 85.25
U201814100 Beta 87 68 76 72 303 75.75
U201814050 Jim 100 78 59 69 306 76.50
U201814999 Linda 88 76 65 34 263 65.75
U201814666 Bob 100 90 45 75 310 77.50

U201814666 Bob 100 90 45 75 310 77.50
U201814999 Linda 88 76 65 34 263 65.75
U201814050 Jim 100 78 59 69 306 76.50
U201814100 Beta 87 68 76 72 303 75.75
U201814010 Tim 97 80 79 85 341 85.25
U201814001 Anna 90 87 85 89 351 87.75

-----
Process exited after 121.3 seconds with return value 0
Press any key to continue . . .
    
```



```

U201814999 huazhongkeda 100 100 100 100
^Z
U201814999 huazhongkeda 100 100 100 100 400 100.00
input the number of the student and the subject and object grade
<1.English 2.math 3.physics 4.Cprogramming>
U201814777
4 100
Not found the student!

U201814999 huazhongkeda 100 100 100 100 400 100.00
U201814999 huazhongkeda 100 100 100 100 400 100.00

-----
Process exited after 23.59 seconds with return value 0
Press any key to continue . . .

U201814999 huazhongkeda 100 100 100 100
^Z
U201814999 huazhongkeda 100 100 100 100 400 100.00
input the number of the student and the subject and object grade
<1.English 2.math 3.physics 4.Cprogramming>
U201814999
5 100
ERROR!

U201814999 huazhongkeda 100 100 100 100 400 100.00
U201814999 huazhongkeda 100 100 100 100 400 100.00

-----
Process exited after 24.62 seconds with return value 0
Press any key to continue . . .

```

图 7-8 选做程序设计题 3 程序运行结果截图

### 7.3 实验小结

1. 交换结点数据域中的字符串，编写字符串交换函数时用了一个悬挂字符指针作为中间保存变量，导致程序直接崩溃，后发现问题后，创建数组解决了问题。
2. 交换结点指针域，在每次交换完后忘记使指针指向下一个结点，导致排序结构不正确，发现问题后，一开始使 p 和 t 两个指针分别指向自己指向的结点，但也不能正确运行，后发现在交换结点时，t 的值有发生变化，只能使它指向 p 更新后的下一个结点。
3. 刚开始写的所有链表程序都没有 free 所 malloc 来的空间，后写实验报告的时候突然想起这一点，这才为所有程序释放空间。

## 8 文件实验

### 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 实验内容

#### 8.2.1. 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。

**解答：**

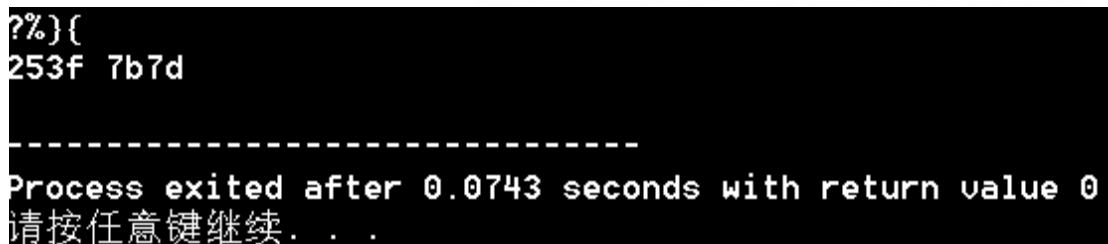
- ① 理论运行结果：?%} {

253f 7b7d

② 分析原因：第一行四个字符的 16 进制 ASCII 码分别为 3f 25 7d 7b

第二行分别输出 a, b 的值

③ 实际运行结果如图 8-1 所示。



```
?%}{
253f 7b7d
-----
Process exited after 0.0743 seconds with return value 0
请按任意键继续...
```

图 8-1 程序验证题 (1) 实际运行结果

(2) 将两处 sizeof(short) 均改为 sizeof(char) 结果有什么不同, 为什么?

**解答:**

1) 不同: 第一行只输出两个字符, 分别为? }

2) 原因: 进行写操作时分别只写了 a, b 的第一个字节 (sizeof(char)=1)

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

**解答:**

1) 不同: 第二行输出的数变为 9535 31613

2) 原因: 将 16 进制的 253f 和 7b7d 转换成十进制后分别为 9535 和 31613

### 8.2.2. 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) 逻辑错误: while(ch=fgetc(fp)!=EOF) 中忽视了运算符的优先级

2) 应该为: while((ch=fgetc(fp))!=EOF)

3) 修改调试后的源程序清单:

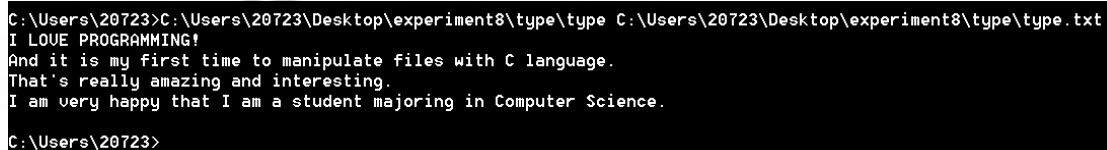
```

#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;
}

```

4) 运行结果如图8-2所示。



```

C:\Users\20723>C:\Users\20723\Desktop\experiment8\type\type C:\Users\20723\Desktop\experiment8\type\type.txt
I LOUE PROGRAMMING!
And it is my first time to manipulate files with C language.
That's really amazing and interesting.
I am very happy that I am a student majoring in Computer Science.
C:\Users\20723>

```

图 8-2 源程序修改替换题 (1) 运行结果截图

(2) 用输入输出重定向 freopen 改写 main 函数。

解答:



## 1) 源程序清单

```
#include<stdio.h>
#include<stdlib.h>

int main(int argc, char* argv[])
{
    char ch;
    if(argc!=2){
        printf("Arguments error!\n");
        exit(-1);
    }
    if((freopen(argv[1],"r",stdin))==NULL){          /* stdin 指向 filename */
        printf("Can't open %s file!\n",argv[1]);
        exit(-1);
    }
    while((ch=getchar())!=EOF)
        putchar(ch);
    fclose(stdin);                                  /* 关闭filename */
    return 0;
}
```

2) 程序运行结果截图如图8-3所示。

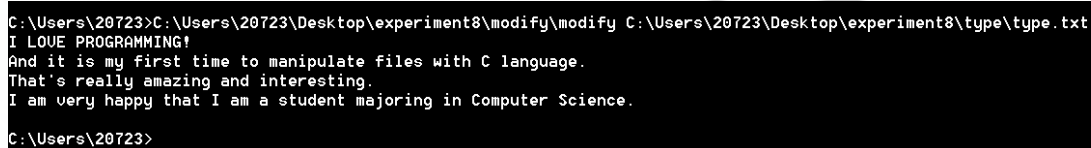


图8-3 源程序修改替换题（2）运行结果截图

### 3. 编程设计题

从键盘输入一行英文句子，将每个单词的首字母换成大写字母，然后输出到一个磁盘文件“test”中保存。

**解答：**

(1) 算法流程解释如下。

- ① 输入得到一个字符串
- ② 保证字符串不为空，否则报错并结束
- ③ 从第二个字符开始遍历字符串，若前一个字符不是字母，且这个字符为小写字母，则将此小写字母变为大写字母，此循环过程遇‘\0’结束
- ④ 使文件指针 fp 指向 D 盘下 test 文件
- ⑤ 对 fp 指向的文件执行 fprintf，输出修改后的字符串
- ⑥ 结束



(2) 源程序清单。

```
#include <stdio.h>
#include <stdlib.h>

int decide(char c);

int main(int argc, char *argv[]) {
    char str[200];
    FILE *fp;
    gets(str);
    int i=1;
    if(str[0]){//保证字符串不为空，否则报错
        while(str[i]){
            if(!decide(str[i-1]))&&(decide(str[i]))==1)str[i]=str[i]-'a'+'A';
            //若前一个字符不是字母，且这个字符为小写字母，则变为大写
            i++;
        }
        if(fp=fopen("D:\\test","w")){    //fp指向test
            fprintf(fp,"%s",str);
            fclose(fp);
        }
        else printf("Can't open the file!");
    }
    else printf("Empty string!");
    return 0;
}

int decide(char c){
    int ret=0;                //不是字母则返回0
    if(c>='a'&&c<='z')ret=1;
    else if(c>='A'&&c<='Z')ret=2;
    return ret;
}
```

(3) 测试。

(a) 测试数据。

Do you like PROgramMING???????un Hun.....i really like it!  
why? ??

(b) 测试结果如图 8-4 所示。

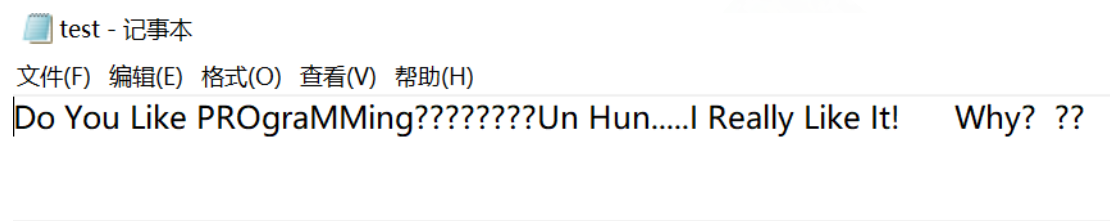


图 8-4 程序设计题程序运行结果截图

### 8.3 实验小结

1. 刚开始运行验证题时始终无输出，但是返回值正确，说明程序运行正常，但无结果，后发现所使用的机器无 D 盘，而打开的文件路径恰好在 D 盘，因而 fopen 函数返回了空指针，后发现问题后，插入 USB 作为 D 盘，后正常运行。这提示我以后要注意文件的路径。
2. 改错题中一开始没注意到判断文件尾的时候少了括号，因而程序始终无法正确运行，后在同学提醒下发现这一问题。以后编程过程中要细心谨慎，不得有半点马虎。
3. 完成程序设计题时思路还未理清便开始动手写程序，但始终没办法写出正确的程序，后冷静下来，仔细分析了问题的本质后问题便迎刃而解。所以在动手写程序之前一定要搞清楚用什么算。