
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: 网络空间安全学院

学 号: U202012043

姓 名: 范启航

指导教师: 张云鹤

报告日期: 2020 年 1 月 6 日

网络空间安全学院

目 录

1	表达式和标准输入与输出实验.....	2
1.1	实验目的.....	2
1.2	实验内容.....	2
1.3	实验小结.....	10
2	流程控制实验.....	11
2.1	实验目的.....	11
2.2	实验内容.....	11
2.3	实验小结.....	27
3	函数与程序结构实验	28
3.1	实验目的.....	28
3.2	实验内容.....	28
3.3	实验小结.....	40
4	编译预处理实验.....	41
4.1	实验目的.....	41
4.2	实验内容.....	41
4.3	实验小结.....	51
5	数组实验.....	52
5.1	实验目的.....	52
5.2	实验内容.....	52
5.3	实验小结.....	63
6	指针实验.....	64
6.1	实验目的.....	64
6.2	实验内容.....	64
6.3	实验小结.....	82
7	结构与联合实验.....	83
7.1	实验目的.....	83
7.2	实验内容.....	83
7.3	实验小结.....	95
8	文件操作实验.....	96
8.1	实验目的.....	96
8.2	实验内容.....	96
8.3	实验小结.....	102
	参考文献	103

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 3.14158
```

2) 第 10 行的 scanf 中写入数据应用&f, 正确形式为:

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

3) 第 11 行的除法应用浮点数进行, 正确形式为:

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

4) 第 15 行 scanf 输入浮点数是应用%lf, 正确形式为:

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

5) 第 17 行 printf 输出时不用取地址符&, 正确形式为:

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

6) 第 21 行 k 的高字节作为低字节应向右移, 正确形式为:

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

(2) 错误修改后运行结果:

```

C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验1\1.exe
Input Fahrenheit: 78
78 (F) = 25.56 (C)
input the radius r:1
The acreage is 3.14
newint = 84a1

-----
Process exited after 5.788 seconds with return value 0
请按任意键继续. . .

```

图 1-1 运行结果示意图

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：

- (1) 思路：用三个异或程序，将 `a`, `b` 值互换
- (2) 主要代码如下：

`a=a ^b ; b= a ^ b; a=a ^ b;`

运行结果：

```

C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验1\sketch.exe
Input two integers:3 5
a=5, b=3
-----
Process exited after 1.777 seconds with return value 8
请按任意键继续. . .
    
```

图 1-2 运行结果示意图

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。

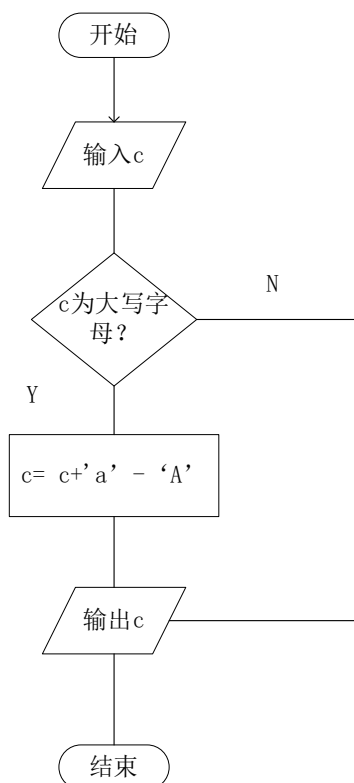


图 1-3 编程题 1 的程序流程图

2) 主要代码如下:

```
if(c >= 'A' && c <= 'Z')
    c = c + 'a' - 'A';
printf("%c", c);
```

(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:

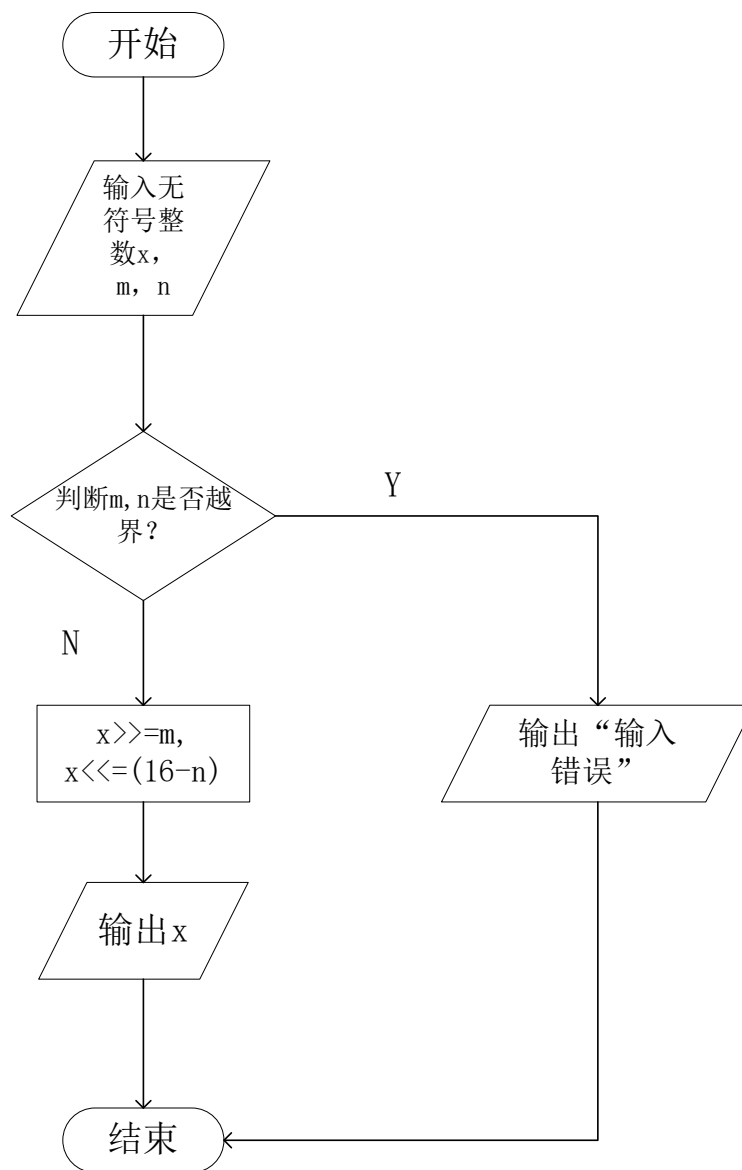


图 1-4 流程图程序流程设计 2 的流程图

2) 主要代码如下:

```
unsigned short x,m,n;
```

```
x >>= m;
```

```
x <<= (16-n);
```

```
printf("%X", x);
```

3) 测试

(a) 测试数据:

如表 1-1 所示。

表 1-1 编程题 3 的测试数据

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

```
C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验1\3(2).exe
4689 7 4
D000
-----
Process exited after 6.094 seconds with return value 0
请按任意键继续. . .
```

图 1-5 编程题 3 的测试用例一的运行结果

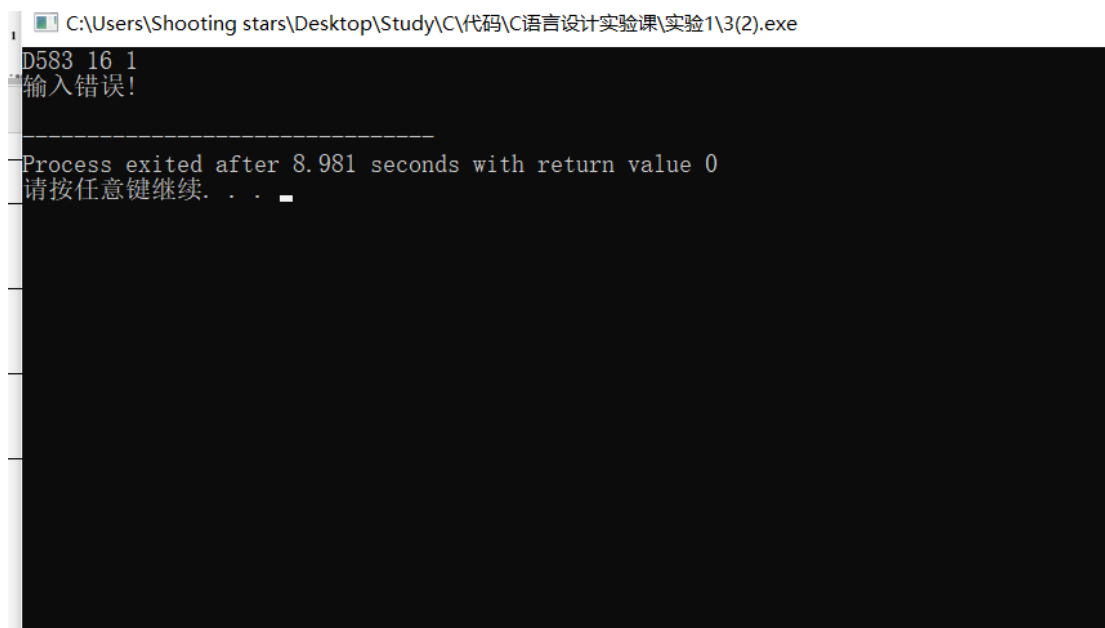


图 1-6 编程题 3 的测试用例二的运行结果

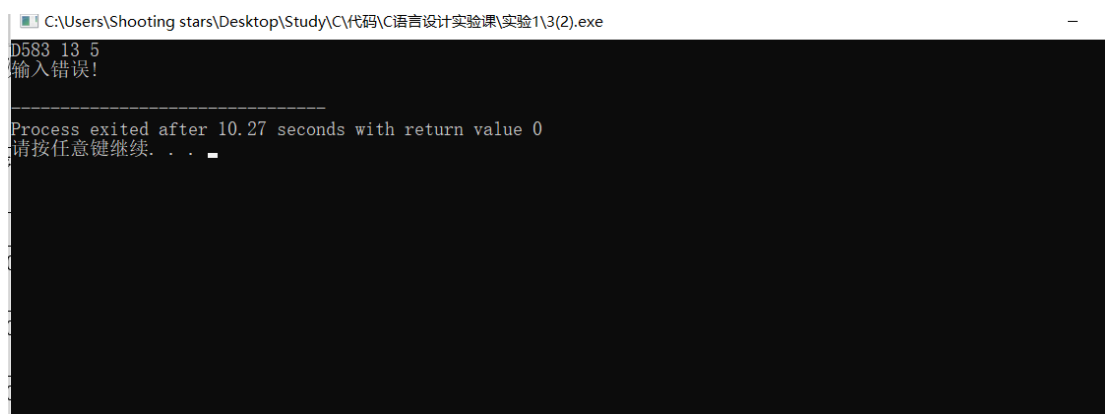


图 1-7 编程题 3 的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1.2.4 选做题

某加密算法对数据按字节进行加密，具体为：对字节的 8 个二进制位从右向左用 0~7 编号，先将 0、2、4 位分别与 1、3、5 位两两对应交换，接着对 0~5 位进行循环左移（左边移出的位接在右边），循环左移的位数有 6、7 两位的值决定。例如，若 6、7 位组成的二进制数为 01，则将 0~5 位左移 1 位，最后得到加密结果，如图 1-1 所示。

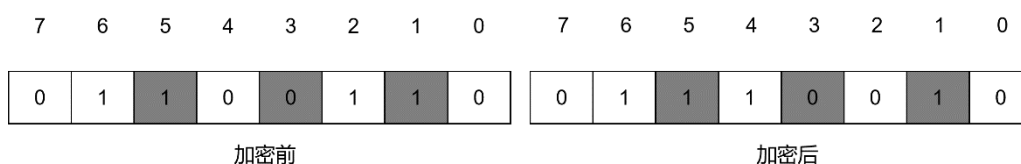


图 1-8 加密示意图

输入一行明文字符串，按该算法进行加密后输出密文。例如，输入“abcd”，则输出“dbfp”。

解答：

主要代码如下：

```
while(word[t] != '\0')
{
    int i;
    char a;
    a = word[t];
    int b[8],d[8];
    for(i = 0;i <8;i++){
        b[i] = a & 1;
        a >>=1;
    }
    for( i=0; i<6; i+=2){
        int z;
        z = b[i];
        b[i] = b[i+1];
        b[i+1] = z;
    }
    int c = 0;
    c = b[6]+b[7]*2;
    for(i = 0; i<8;i++)
        d[i] = b[i];
    for(i=5; i>c-1; i--)
        b[i]=d[i-c];
    for(i = 0; i<c; i++)
        b[i]=d[5-c+1+i];
    for(i = 0; i<8;i++)
        a+=b[i]<<i;
    printf("%c", a);
    t ++;
```

1.2.5 自设题

题目：

题目描述：有一只小鱼，它平日每天游泳 250 公里，周末休息（实行双休日），假设从周 $x(1 \leq x \leq 7)$ 开始算起，过了 $n(n \leq 10^6)$ 天以后，小鱼一共累计游泳了多少公里呢？

输入格式：输入两个整数 x, n (表示从周 x 算起，经过 n 天)

输出格式：输出一个整数，表示小鱼累计游泳了多少公里。

输入输出样例：

输入：3 10

输出：2000

解答：

主要代码：

```
for (i = x; i <= x + n; i++)
{
    if (!(i % 7 == 6 || i % 7 == 0))
        y += 250;
}
printf("%d", y);
return 0;
}
```

1.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

实验过程中遇到了一系列问题，比如变量的输入，长整型，短整型，无符号变量的输入与输出等，改错题中遇到了左移后高位不补 0 的现象需要主动补 0，代码优化实验中遇到了使用位运算进行两个变量交换的问题，在网上查找资料后学习到了通过三个异或运算进行变量交换方法，省去了定义新变量的过程，通过移位来达到处以 2 或乘以 2 的目的，通过（）中赋值来实现一步进行多个赋值操作，通过试验得到 scanf 也能进行 EOF 判断，通过草稿中手写可以轻松得出移位项数的判断，通过反复移位来实现二进制数的分段提取。

2 流程控制实验

2.1 实验目的

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

(2) 练习循环结构 for、while、do-while 语句的使用。

(3) 练习转移语句和标号语句的使用。

(4) 使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

2.2 实验内容

2.2.1 程序改错

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

/* 实验 2-1 改错题程序：合数判断器*/

```
1  #include <stdio.h>
2  int main( )
3  {
4  int i, x, k, flag = 0;
5  printf("本程序判断合数，请输入大于 1 的整数，以 Ctrl+Z 结束\n");
6  while (scanf("%d", &x) != EOF) {
7  for(i=2,k=x>>1;i<=k;i++)
8      if (!x%i) {
9          flag = 1;
10         break;
11     }
12  if(flag=1) printf("%d 是合数", x);
13  else printf("%d 不是合数", x);
14  }
15  return 0;
```

解答：

(1) 错误修改：

- 1) 第 8 行中 if 内少了括号，应为 if(!(x%i))
- 2) 第 12 行中 if 语句中判断是否相等应用==号，改为 if(flag == 1)
- 3) while 循环中 flag 应复位为 0；while 循环中插入 flag = 0；

(2) 错误修改后运行结果：



```

C:\Users\Shooting stars\Desktop\Study\C\代码\语言设计实验课\实验2\1.exe
本程序判断合数，请输入大于1的整数，以Ctrl+Z结束
12
12是合数
13
13不是合数
17
17不是合数
19
19不是合数
Z
-----
Process exited after 9.519 seconds with return value 0
请按任意键继续. . .
  
```

图 2-1 运行结果示意图

2.2.2 程序修改替换

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

(2) 修改实验 2-1 程序，将 for 循环改用 do-while 循环。

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

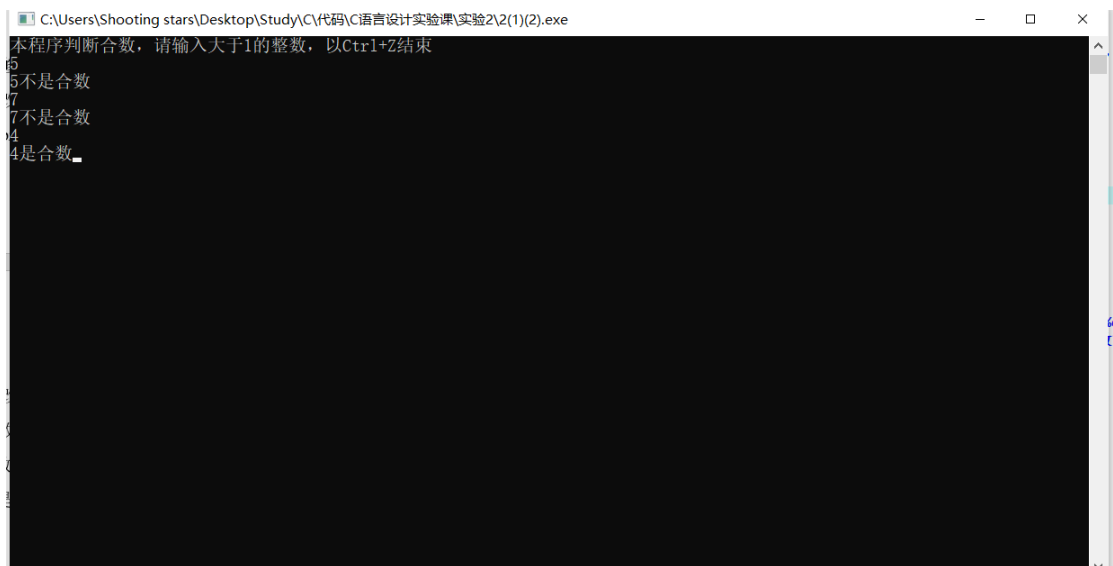
解答：

(1) (2) 替换后的如下：

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

```
2     if (!(x==2))
3     {
4         i = 2, k = x>>1;
5         do{
6             if (!(x%i))
7             {
8                 flag = 1;
9                 i = k;
10            }
11        i ++;
12    }while(i <= k);
13    }else
14    flag = 0;
15    if(flag == 1) printf("%d 是合数", x);
16    else printf("%d 不是合数", x);
17    flag = 0;
18    }
19
```

运行结果示意图：



```
C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验2\2(1)(2).exe
本程序判断合数，请输入大于1的整数，以Ctrl+Z结束
5不是合数
7不是合数
4是合数
```

图 2-2 运行结果示意图

(3) 代码如下：

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

```
int isheshu(int );

int main()
{
    int heshu3,flag = 0;
    int heshu;
    for(heshu3 = 100;heshu3<=999;heshu3++)
    {
        heshu = heshu3;
        if(isheshu(heshu) && isheshu(heshu/10) && isheshu(heshu/10))
        }
    }
    return 0;
}

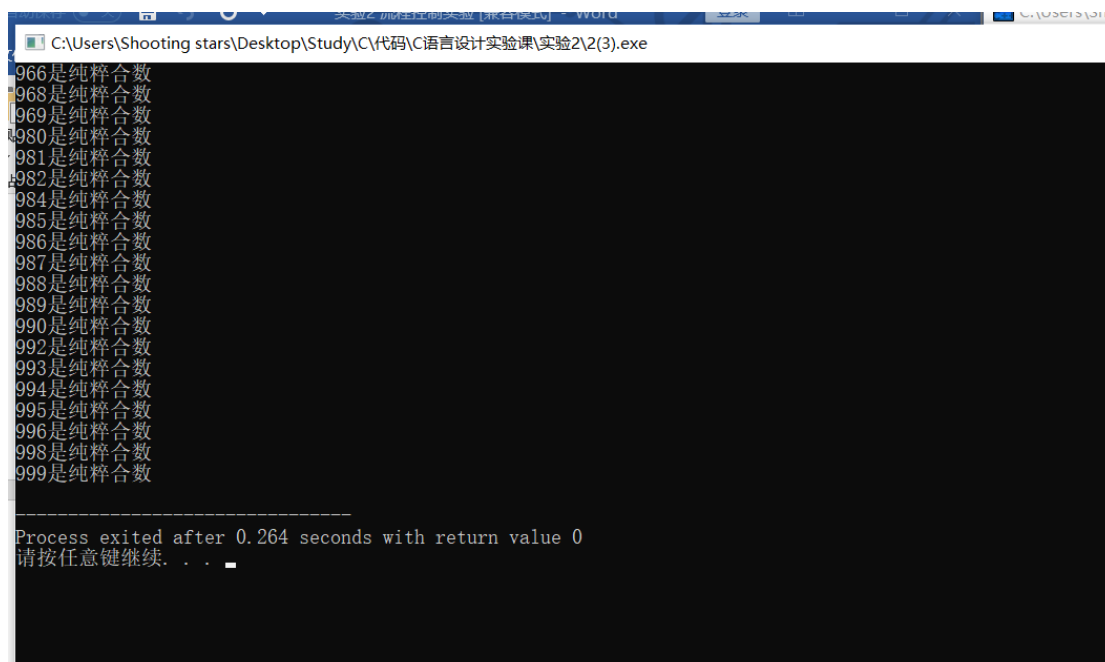
int isheshu(int x )
{
    int i,k,flag;
    i = 2, k = x>>1, flag = 0;
    if(!(x ==0 || x==1 || x==2))
    {
        do{
            if (!(x%i))
            {
                flag = 1;
                i = k;
            }
            i ++;
        }while(i <= k);
    }
}
```

```
else flag = 0;

return flag;

}
```

运行示意图：



```
C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验2\2(3).exe
966是纯粹合数
968是纯粹合数
969是纯粹合数
980是纯粹合数
981是纯粹合数
982是纯粹合数
984是纯粹合数
985是纯粹合数
986是纯粹合数
987是纯粹合数
988是纯粹合数
989是纯粹合数
990是纯粹合数
992是纯粹合数
993是纯粹合数
994是纯粹合数
995是纯粹合数
996是纯粹合数
998是纯粹合数
999是纯粹合数

Process exited after 0.264 seconds with return value 0
请按任意键继续...
```

图 2-3 运行结果示意图

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； $x > 5000$ ，收取 25% 的税金。（注意税金的计算按照阶梯计税法，比如，工资为 4500，那么税金= $1000 \times 5\% + 1000 \times 10\% + 1000 \times 15\% + 501 \times 20\%$ ）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用 if 语句和 switch 语句来实现。

解答：

流程图如下：

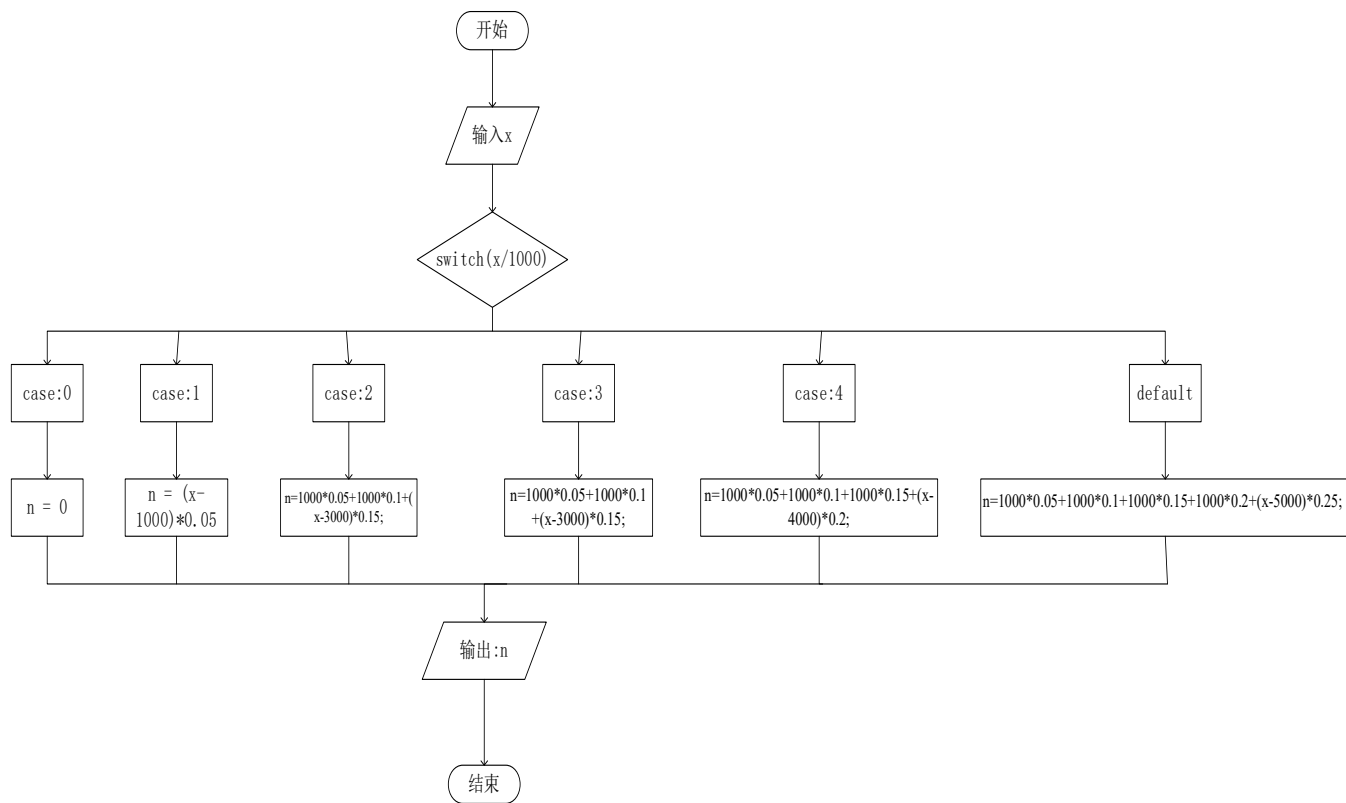


图 2-4 程序设计流程图

代码如下：

```

switch 语句：
switch(x/1000)
{
case 0: n=0;
break;
case 1: n=(x-1000)*0.05;
break;
case 2: n=1000*0.05+(x-2000)*0.1;
break;
case 3: n=1000*0.05+1000*0.1+(x-3000)*0.15;
break;
case 4: n=1000*0.05+1000*0.1+1000*0.15+(x-4000)*0.2;
break;
default : n=1000*0.05+1000*0.1+1000*0.15+1000*0.2+(x-5000)*0.25;
}
    
```

```
break;}
```

if 语句:

```
if(x>0 && x<1000) n=0;
else if(x>=1000 && x<2000) n=(x-1000)*0.05;
else if(x>=2000 && x<3000) n=1000*0.05+(x-2000)*0.1;
else if(x>=3000 && x<4000) n=1000*0.05+1000*0.1+(x-3000)*0.15;
else if(x>=4000 && x<5000) n=1000*0.05+1000*0.1+1000*0.15+(x-4000)*0.2;
else if(x>=5000) n=1000*0.05+1000*0.1+1000*0.15+1000*0.2+(x-5000)*0.25;
```

运行示例:

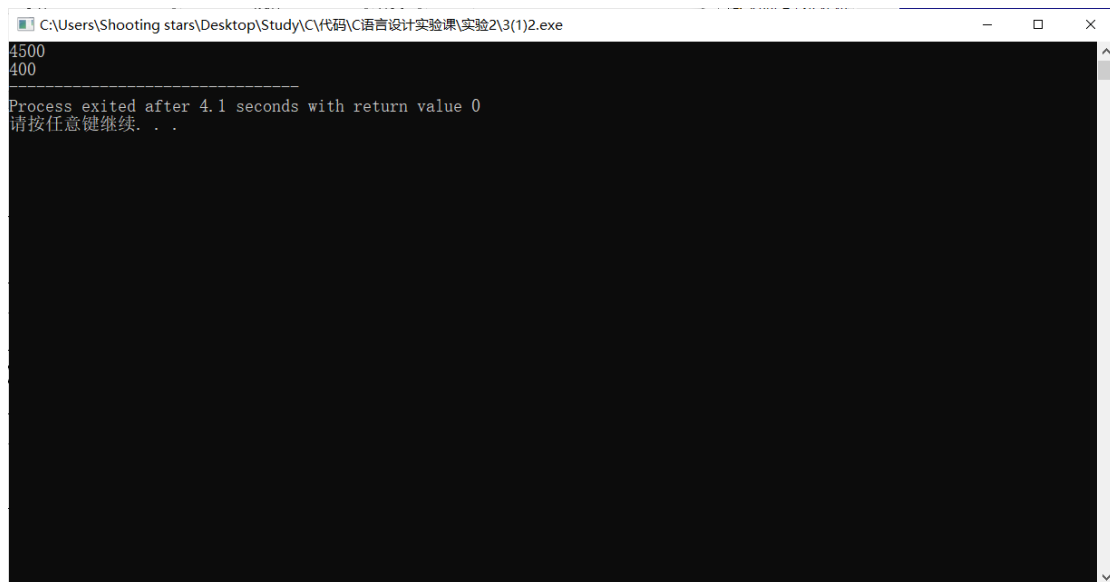


图 2-5 运行结果示意图

(2) 将输入的正文复制到输出, 复制过程中将每行一个以上的空格字符用一个空格代替。

解答:

流程图:

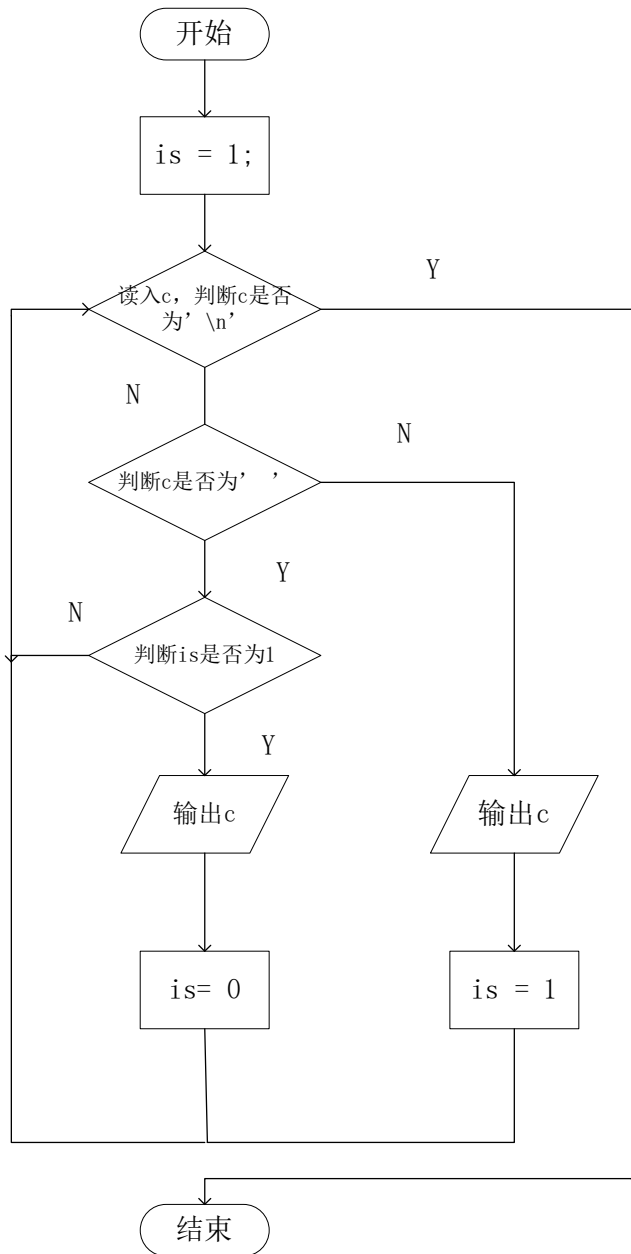


图 2-6 程序设计流程图

代码:

```

#include<stdio.h>
int main()
{
    char c,is;
    is = 1;
    while((c = getchar()) !='\n')
    {
        if(c == ' ')
    
```

```
{  
    if(is)  
    {    putchar(c);  
        is = 0;  
    }  
}  
else  
{  
    putchar(c);  
    is = 1;  
}  
}
```

运行示例：

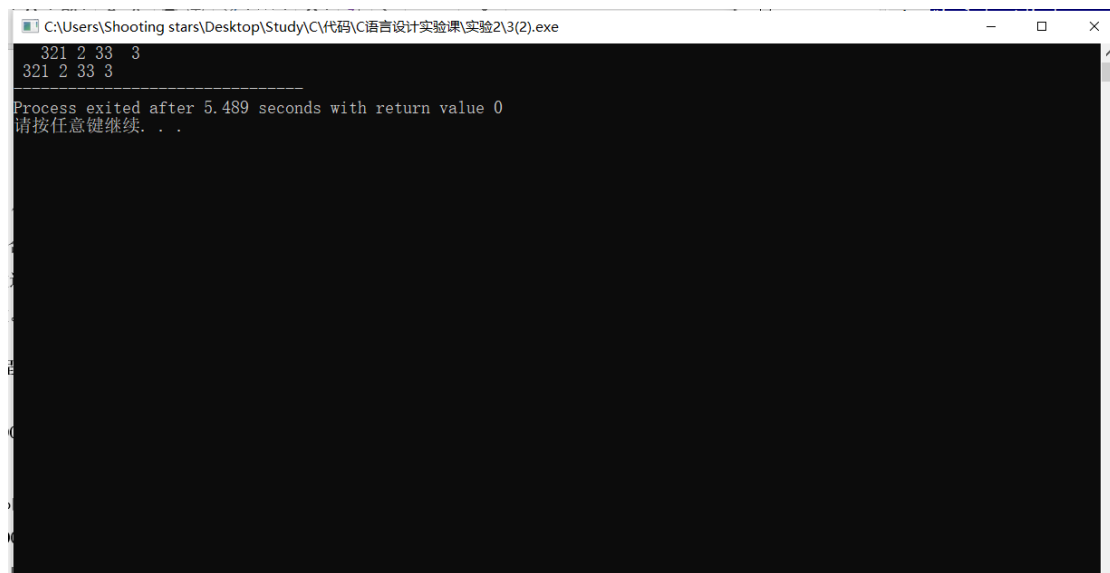


图 2-7 运行结果示意图

(3) 打印如下的杨辉三角形。

1					/*第 0 行 */
1	1				/*第 1 行 */
1	2	1			/*第 2 行 */
1	3	3	1		
1	4	6	4	1	

```

        1   5   10  10  5   1
      1   6   15  20  15  6   1
    1   7   21  35  35  21  7   1
  1   8   28  56  70  56  28  8   1
1   9   36  84 126 126 84  36  9   1

```

第 i 行第 j 列位置的数据值可以由组合 C_i^j 表示，而 C_i^j 的计算如下：

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

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

根据以上公式，采用顺推法编程，输出金字塔效果的杨辉三角形。特别要注意空格的数目，一位数之间是 3 个空格，两位数之间有 2 个空格，3 位数之间只有一个空格。

解答：

主要代码：

```

for(i=0; i<=number; i++)
{
    a[i][0] = 1;
}
for(i = 0; i<=number; i++)
{
    for(j = 1; j<=i; j++){
        a[i][j]= a[i][j-1]*(i-j+1)/j; }
}
for(i=0; i<=number; i++)
{
    while(k<2*number-2*i)
    {
        printf(" ");
        k++;
    }
    k = 0;
}

```

运行示例：

```

10
      1
     1 1
    1 2 1
   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
1 10 45 120 210 252 210 120 45 10 1
Process exited after 0.8454 seconds with return value 0
请按任意键继续. . .
    
```

图 2-8 运行结果示意图

(4) 625 这个数很特别，625 的平方等于 390625，其末 3 位也是 625。请编程输出所有这样的 3 位数：它的平方的末 3 位是这个数本身。、

解答：

流程图：

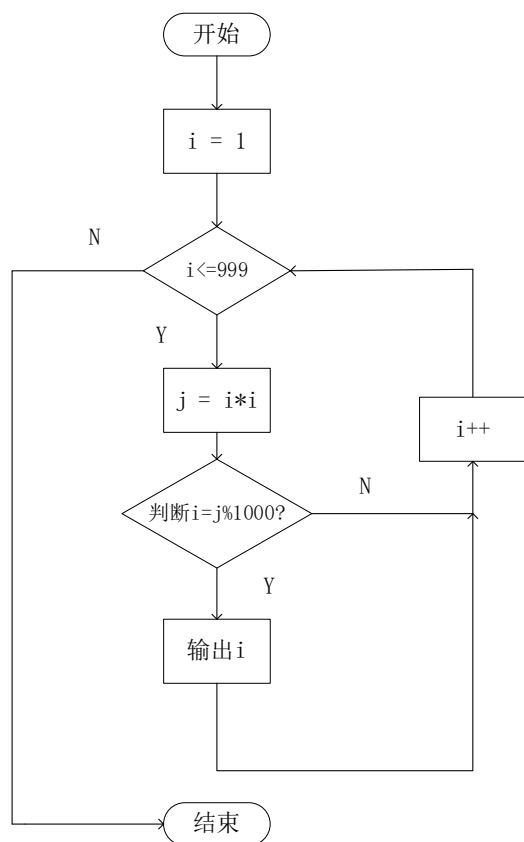


图 2-9 程序设计流程图

代码：

```
int i,j;
for(i = 100; i<=999; i++)
{
    j = i*i;
    if(i == j%1000)
        printf("%d\n",i);
}
```

运行示例：

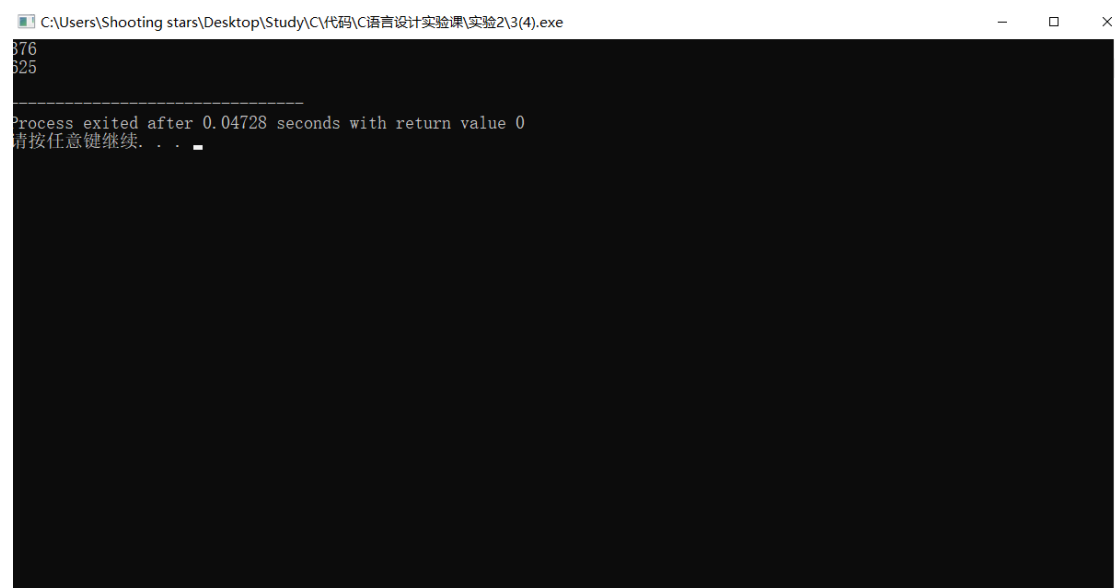


图 2-10 运行结果示意图

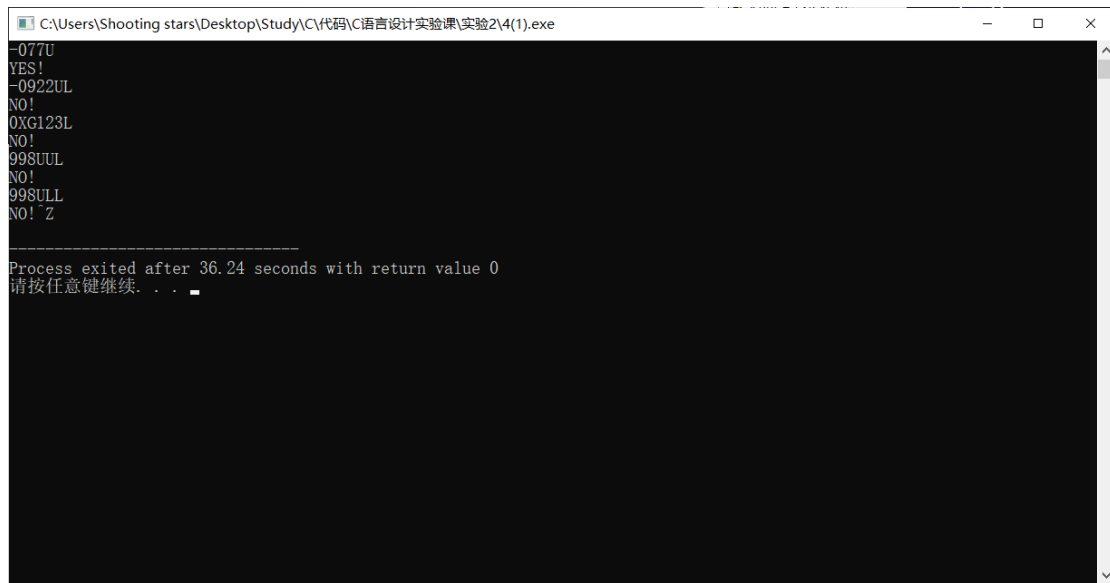
2.2.3 选做题

(1) 判断给定的字符串是否是合法的 C 整型常量，如果是，则输出 Yes；不是，则输出 No。例如，0xabL 是 C 整形常量，而 092 不是 C 整形常量。要求程序能够循环接受用户的输入，每行输入一个字符串，给出判定结果，直至输入 Ctrl+Z 结束。

解答：

主要代码：见电子版报告

运行示例：



```
C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验2\4(1).exe
-077U
YES!
-0922UL
NO!
0XG123L
NO!
998UUL
NO!
998ULL
NO! `Z

-----
Process exited after 36.24 seconds with return value 0
请按任意键继续. . .
```

图 2-11 运行结果示意图

(2) 输入正整数 x ($2 \leq x \leq 79$)，输出所有形如 $abcde/fghij=x$ 的表达式，其中 $a \sim j$ 由不同的数字 $0 \sim 9$ 组成。例如： $x=32$ 时，输出为：75168/02349=32。

解答：

代码：

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

```
int main()
```

```
{
```

```
    int n,x,y,flag,i,j,x1,y1;
```

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

```
    for(x = 12345; x <= 98765; x++)
```

```
    {
```

```
        int a[10] = {0};
```

```
        i=0;
```

```
        flag = 1;
```

```
        x1 = x;
```

```
        if( x % n == 0)
```

```
        {
```

```
            y = x/n;
```

```
            y1 = y;
```



```
        for(i = 0;i<=4;i++)
        {
            a[i] = x1%10;
            x1 /=10;
        }
        for(i=5;i<=9;i++)
        {
            a[i] = y1%10;
            y1 /=10;
        }
        for(i = 0;i<=9;i++)
        {
            for(j = 0;j <=9;j++)
            {
                if(i != j)
                {
                    if(a[i] == a[j])
                    {
                        flag = 0;
                        break;
                    }
                }
            }
            if(flag == 0)
                break;
        }
        if (flag == 1)
            printf("%05d/%05d=%d\n", x,y,n);
    }
}

return 0;
}
```

运行示例：



```

C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验2\4(2).exe
3
17469/05823=3
17496/05832=3
50382/16794=3
53082/17694=3
61749/20583=3
69174/23058=3
91746/30582=3
96174/32058=3
-----
Process exited after 1.237 seconds with return value 0
请按任意键继续. . .

```

图 2-12 运行结果示意图

2.2.4 自设题

题目描述：国王将金币作为工资，发放给忠诚的骑士。第一天，骑士收到一枚金币；之后两天（第二天和第三天），每天收到两枚金币；之后三天（第四、五、六天），每天收到三枚金币；之后四天（第七、八、九、十天），每天收到四枚金币.....；这种工资发放模式会一直这样延续下去：当连续 N 天每天收到 N 枚金币后，骑士会在之后的连续 $N+1$ 天里，每天收到 $N+1$ 枚金币。

请计算在前 KK 天里，骑士一共获得了多少金币。

输入格式：一个正整数 KK ，表示发放金币的天数。

输出格式：一个正整数 KK ，表示发放金币的天数。

解答

流程图：

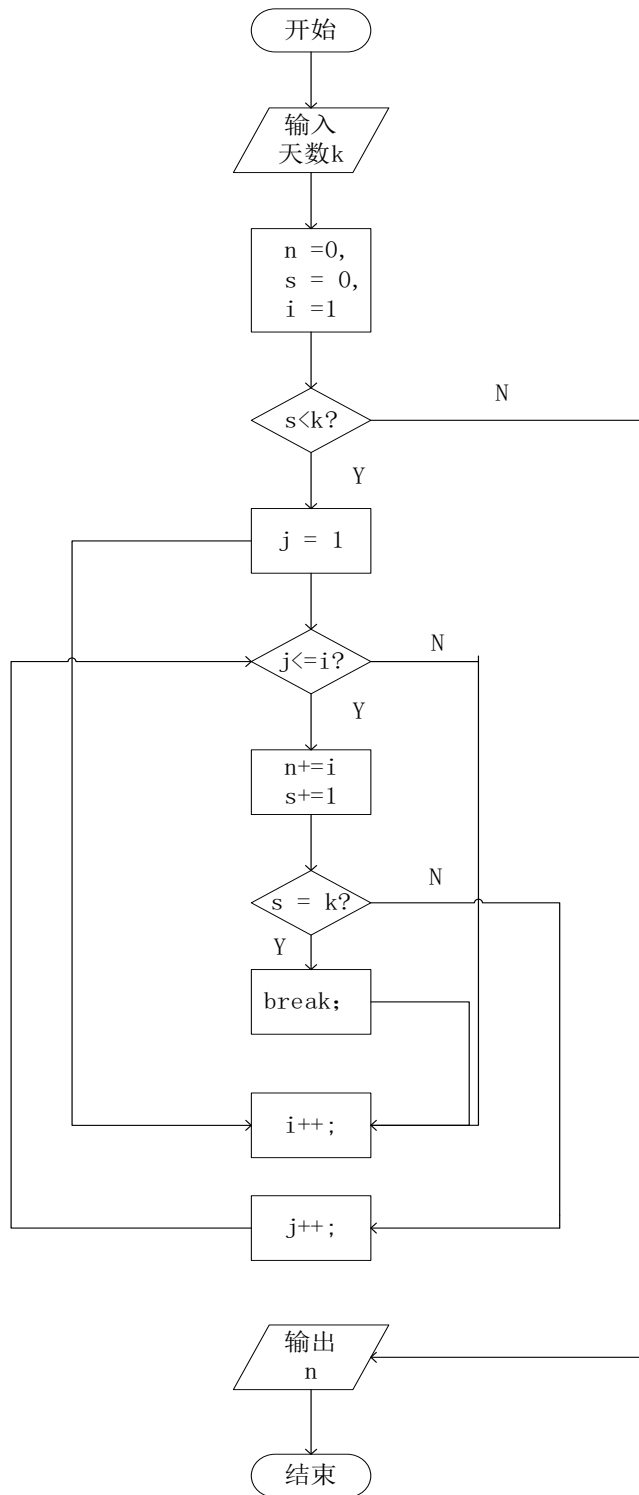


图 2-13 程序设计流程图

代码:

```

while( s<k)
{
    for(j = 1; j<=i; j++)

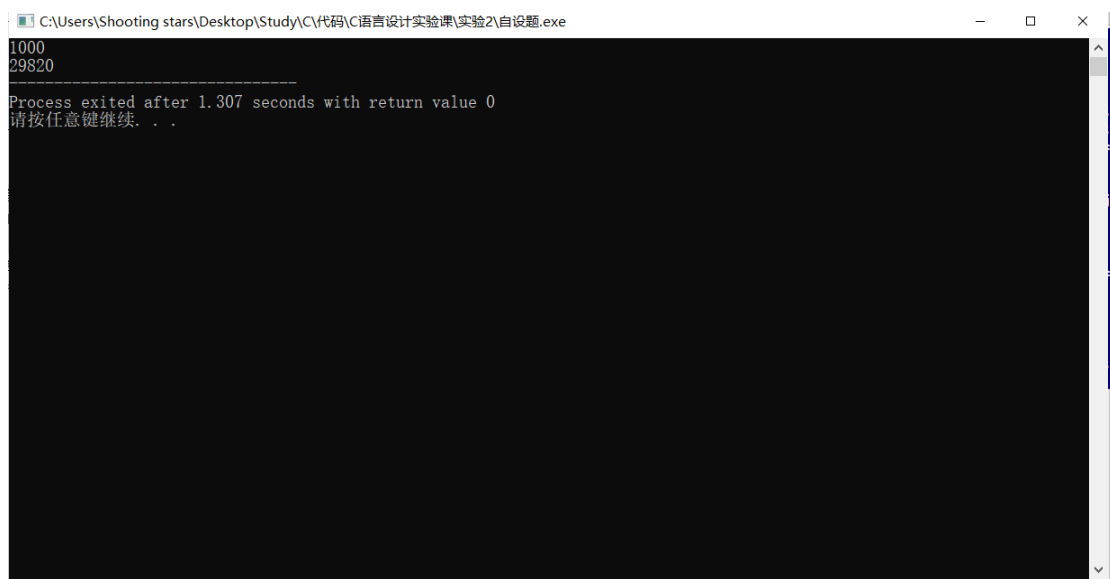
```

```

    {
        n += i;
        s +=1;
        if (s == k)
            break;
    }
    i++;
}

```

运行示例：



```

C:\Users\Shooting stars\Desktop\Study\C\代码\C语言设计实验课\实验2\自设题.exe
1000
29820
-----
Process exited after 1.307 seconds with return value 0
请按任意键继续. . .

```

图 2-14 运行结果示意图

2.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验过程中，有一些算法比较复杂，有许多重复的步骤，可以通过合并，`break` 等语句进行省略，极大地提高了运算效率，使代码的可读性更强，更有利于处理复杂的问题。

3 函数与程序结构实验

3.1 实验目的

- (1) 熟悉和掌握函数的定义、声明；函数调用与参数传递，函数返回值类型的定义和返回值使用。
- (2) 熟悉和掌握不同存储类型变量的使用。
- (3) 练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

3.2 实验内容

3.2.1 程序改错题

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

k=1 the sum is 1

k=2 the sum is 3

k=3 the sum is 9

.....

k=20 the sum is 2561327494111820313

/*实验 3-1 改错题程序：计算 $s=1!+2!+3!+\dots+n!$ */

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int k;
5      for(k=1;k<=20;k++)
6          printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
7      return 0;
8  }
9  long sum_fac(int n)
10 {
```

```

11     long s=0;
12     int i,fac;
13     for(i=1;i<=n;i++)
14         fac*=i;
15     s+=fac;
16     return

```

解答：

错误修改：

1) 第 2 行未申明函数，正确形式为：

```
long long sum_fac(int);
```

2) 第 6 行输出格式错误，正确形式为：

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

3) 函数返回类型，fac,s 应用 long long 型整数

4) fac 循环后未初始化；

5) 第 15 行累加操作应在循环内部进行，正确形式为：

```

{
    fac*=i;
    s+=fac;
}

```

错误修改后运行结果：



```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验3\1.exe
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
k=6 the sum is 873
k=7 the sum is 5913
k=8 the sum is 46233
k=9 the sum is 409113
k=10 the sum is 4037913
k=11 the sum is 43954713
k=12 the sum is 522956313
k=13 the sum is 6749977113
k=14 the sum is 93928268313
k=15 the sum is 1401602636313
k=16 the sum is 22324392524313
k=17 the sum is 378011820620313
k=18 the sum is 6780385526348313
k=19 the sum is 128425485935180313
k=20 the sum is 2561327494111820313

```

图 3-1 运行结果示意图

3.2.2 程序修改替换题

(1) 根据 $\sum_{i=1}^n i! = \sum_{i=1}^{n-1} i! + n!$ 将实验 3-1 改错题程序中 sum_fac 函数修改为一

个递归函数，用递归的方式计算 $\sum_{i=1}^n i!$ 。

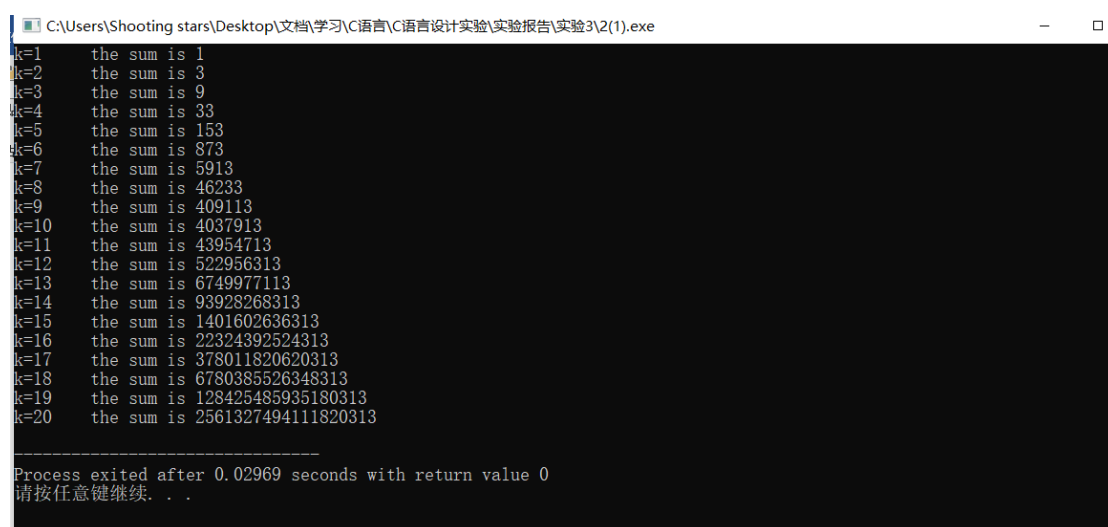
解答：

主要代码如下：

```

1 long long sum_fac(int n)
2 {
3     fac = 1;
4     for(i=1;i<=n;i++)
5     {
6         fac *=i;
7     }
8     if(n == 1)
9         s = 1;
10    else s =sum_fac(n-1) + fac;
11    return s;
12 }
```

运行结果图：



```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验3\2(1).exe
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
k=6 the sum is 873
k=7 the sum is 5913
k=8 the sum is 46233
k=9 the sum is 409113
k=10 the sum is 4037913
k=11 the sum is 43954713
k=12 the sum is 522956313
k=13 the sum is 6749977113
k=14 the sum is 93928268313
k=15 the sum is 1401602636313
k=16 the sum is 22324392524313
k=17 the sum is 378011820620313
k=18 the sum is 6780385526348313
k=19 the sum is 128425485935180313
k=20 the sum is 2561327494111820313

Process exited after 0.02969 seconds with return value 0
请按任意键继续. . .
```

图 3-2 运行结果示意图

(2) 下面是计算 $s = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$ 的源程序，其中 x 是浮点

数， n 是整数。从键盘输入 x 和 n ，然后计算 s 的值。修改该程序中的 `sum` 和 `fac` 函数，使之计算量最小。

/*实验 3-2 程序修改替换第(2)题程序：根据公式计算 s */

```
1  #include<stdio.h>
2  double mulx(double x,int n);
3  long fac(int n);
4  double sum(double x,int n)
5  {
6      int i;
7      double z=1.0;
8      for(i=1;i<=n;i++)
9      {
10         z=z+mulx(x,i)/fac(i);
11     }
12     return z;
13 }
14 double mulx(double x,int n)
15 {
16     int i;
17     double z=1.0;
18     for(i=0;i<n;i++)
19     {
20         z=z*x;
21     }
22     return z;
23 }
24 long fac(int n)
25 {
26     int i;
27     long h=1;
28     for(i=2;i<=n;i++)
```



```

29     {
30         h=h*i;
31     }
32     return h;
33 }
34 int main()
35 {
36     double x;
37     int n;
38     printf("Input x and n:");
39     scanf("%lf%d",&x,&n);
40     printf("The result is %lf:",sum(x,n));
41     return 0;
42 }

```

解答：

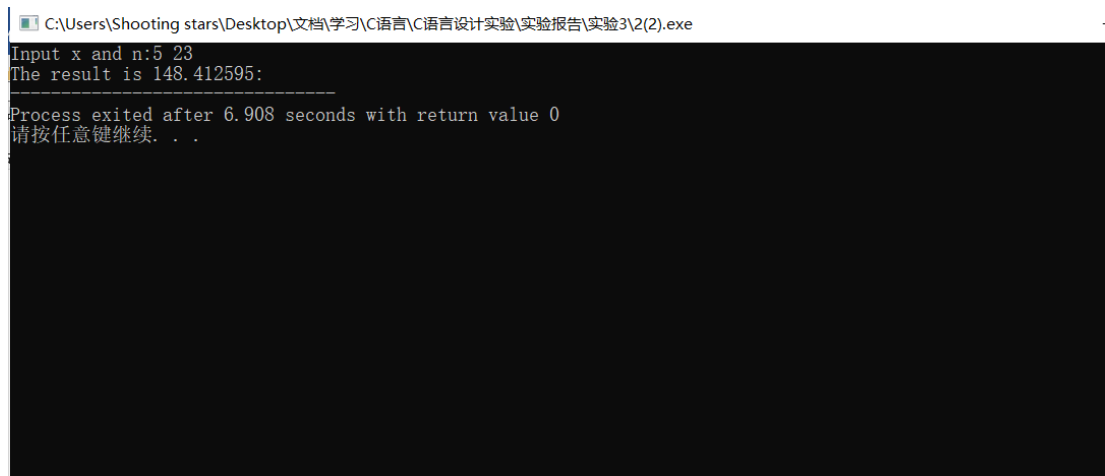
主要代码如下：

```

1 double sum(double x,int n )
2 {
3     double z = 1.0;
4     double last_x = 1.0;
5     long long last_i= 1;
6     int i;
7     for(i = 1; i<=n; i++)
8     {
9         z = z+last_x*x/(last_i*i);
10        last_x *=x;
11        last_i *=i;
12    }
13    return z;
14 }

```

运行结果图：



```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验3\2(2).exe
Input x and n:5 23
The result is 148.412595:
-----
Process exited after 6.908 seconds with return value 0
请按任意键继续. . .
    
```

图 3-3 运行结果示意图

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，在 watch 窗口中观察 lk, sum, n 值。具体操作如下：

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

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

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

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

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

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

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

```
int main(void)
```

```
{
```

```
    int i,k;
```

```

    long sum=0,fabonacci(int n);
    printf("Input n:");
    scanf("%d",&k);
    for(i=1;i<=k;i++){
        sum+=fabonacci(i);
        printf("i=%d\tthe sum is %ld\n",i,sum);
    }
    return 0;
}
long fabonacci(int n)
{
    if(n==1 || n==2)
        return 1;
    else
        return fabonacci(n-1)+fabonacci(n-2);
}

```

解答：

(1)sum = 0; k= 5;

(2)i=1;sum = 0; k =5

(3)调到了 printf 语句

(4)当 n = 3 时先进行 if 判断，后进入如 fabonacci(1)返回 1，在进入 fabonacci(2)返回 1，最后回到 fabonacci(3)返回 1+1=2;

(5)n 先由大变小后由小变大，k,sum 为 main 函数中的变量，在函数 fabonacci 中无法访问

3.2.4 程序设计

(1) 编程验证歌德巴赫猜想：一个大于等于 4 的偶数都是两个素数之和。要求设计一个函数对其形参 n 验证哥德巴赫猜想，并以“n=n1+n2”的形式输出结果。例如：n=6，输出“6=3+3”。main 函数循环接收从键盘输入的整数 n，如果 n 是大于或等于 4 的偶数，调用上述函数进行验证。

(2) 完全数 (Perfect number)，又称完美数或完备数，特点是它的所有真因子 (即除了自身以外的约数，包括 1) 之和恰好等于它本身。例如 $6=1+2+3$ ， $28=1+2+4+7+14$ 等。编程寻找 10^8 以内的所有完全数。要求设计一个函数，判定形参 n 是否为完全数，如果是，则以 n 的真因子之和的形式输出结果，例如 “ $6=1+2+3$ ”；否则，输出 “not a perfect number”，例如 “5 is not a perfect number”。

在 main 函数中调用该函数求 10^8 以内的所有完全数。

(3) 自幂数是指一个 n 位数，它的每个位上的数字的 n 次幂之和等于它本身。水仙花数是 3 位的自幂数，除此之外，还有 4 位的四叶玫瑰数、5 位的五角星数、6 位的六合数、7 位的北斗星数、8 位的八仙数等。编写一个函数，判断其参数 n 是否为自幂数，如果是，则返回 1；否则，返回 0。main 函数能反复接收从键盘输入的整数 k ， k 代表位数，然后调用上述函数求 k 位的自幂数，输出所有 k 位自幂数，并输出相应的信息，例如 “3 位的水仙花数共有 4 个 153，370，371，407”。当 $k=0$ 时程序结束执行。

解答：

(1)

主要代码如下：

```

1  int gd(int n)
2  {
3      int j,i;
4      for(i = 0; i<=k;i++)
5      {
6          for(j = 0;j <= k;j++)
7          {
8              if(a[i]+a[j]== n)
9                  printf("%d=%d+%d\n", n, a[i], a[j]);
10         }
11     }
12 }
13 void isprime(int n)
14 {

```

```
15     int i,j,flag;
16     for(i = 2;i<=n;i ++ )
17     {
18         flag  = 1;
19         for(j = 2 ; j<=i/2;j ++ )
20         {
21             if( i%j == 0){
22                 flag = 0;
23                 break;
24             }
25         }
26         if(flag)
27         {
28             a[k] = i;
29             k++;
30         }
31     }
32 }
```

运行结果图：

```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验3\4(1).exe
16
16=3+13
16=5+11
16=11+5
16=13+3

-----
Process exited after 7.441 seconds with return value 0
请按任意键继续. . .
    
```

图 3-4 程序设计运行结果图 1

```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验3\4(1).exe
34
34=3+31
34=5+29
34=11+23
34=17+17
34=23+11
34=29+5
34=31+3

-----
Process exited after 4.006 seconds with return value 0
请按任意键继续. . .
    
```

图 3-5 程序设计运行结果图 2

(2)

主要代码如下：

```

1 void pfn(int n)
2 {
3     int i,j,s = 0;
4     j = 0;
5     int a[100000] = {0};
6     for(i = 1; i <= (n > 1); i++)
7     {
8         if( n%i == 0)
9         {
10             a[j] = i;
11             j++;
12         }
13     }
    
```

```
14     j = 0;
15     while(a[j] != 0)
16     {
17         s += a[j];
18         j++;
19     }
20     j = 0;
21     if(s == n)
22     {
23         printf("%d=", n);
24         printf("%d", a[j]);
25         j++;
26         while(a[j] != 0)
27         {
28             printf("+%d", a[j]);
29             j++;
30         }
31     }
32     else printf("%d is not a perfect number", n);
33 }
```

运行结果图：

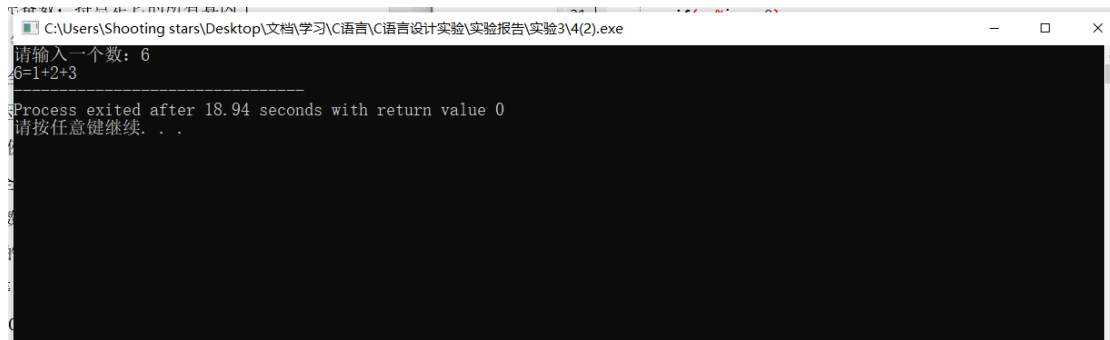


图 3-6 运行结果示意图

(3)

代码如下：

```

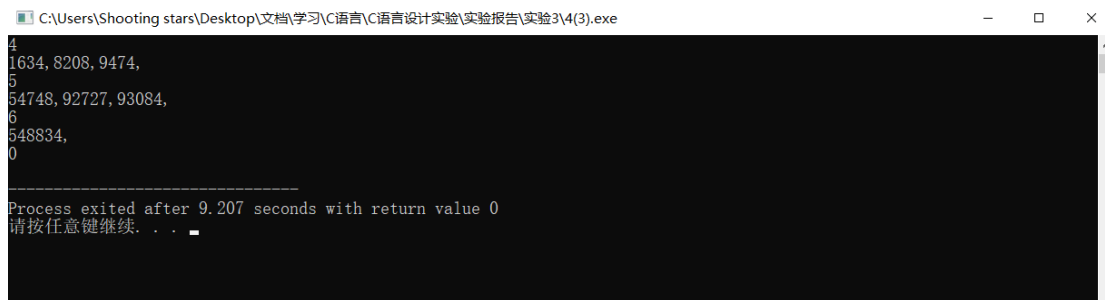
1  #include <stdio.h>
2  #include <math.h>
3  void zimi(int );
4
5  int main()
6  {
7      int k;
8      do
9      {
10         scanf("%d", &k);
11         if( k== 0)
12             break;
13         else zimi(k);
14     }while(1);
15     return 0;
16 }
17
18 void zimi (int n)
19 {
20     int i,j,k,s,x,temp;
21     for(i=pow(10,n-1);i<pow(10, n);i++)
22     {

```



```
23     s = 0;
24     temp = i;
25     for(j = 0;j<n;j++)
26     {
27         x = temp%10;
28         s += pow( x, n);
29         temp /=10;
30     }
31     if( s == i)
32         printf("%d,", s);
33 }
```

运行结果图：



```
C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验3\4(3).exe
4
1634, 8208, 9474,
5
54748, 92727, 93084,
6
548834,
0
-----
Process exited after 9.207 seconds with return value 0
请按任意键继续. . .
```

图 3-7 运行结果示意图

3.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

实验过程中遇到了代码量过大，导致运行效率偏低的情况，通过简化算法，使用数论等方法对代码运行速率进行了提示，同时增加了代码的可读性。

4 编译预处理实验

4.1 实验目的

- (1) 掌握文件包含、宏定义、条件编译和 `assert` 宏的使用；
- (2) 练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。
- (3) 熟悉多文件编译技术。

4.2 实验内容

4.2.1 程序改错

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

/*实验 4-1 改错与跟踪调试题程序：计算平方差、将换两数*/

```
1  #include<stdio.h>
2  #define SUM a+b
3  #define DIF a-b
4  #define SWAP(a,b)  a=b,b=a
5  int main()
6  {
7      int a,b;
8      printf("Input two integers a, b:");
9      scanf("%d%d", &a,&b);
10     printf("\nSUM=%d\n the difference between square of a and square of b
11 is:%d",SUM, SUM*DIF);
12     SWAP(a,b);
13     printf("\nNow a=%d,b=%d\n",a,b);
14     return 0;
15 }
```

解答：

(1) 1、2、3 行宏定义最好加上括号，应改为#define SUM (a+b)

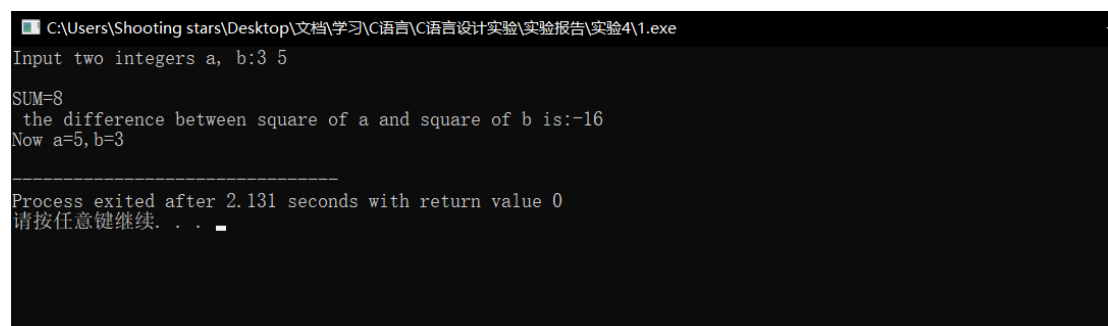
#define DIF (a-b)

(2) 第 4 行中交换变量的方法无法实现，将 b 赋给 a 时 b 的值已经不存在了，不能达到交换变量的效果，应改为：

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

(3) 定义一个变量 temp 用于暂时存储 a 的值

错误修改后运行结果：



```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验4\1.exe
Input two integers a, b: 3 5

SUM=8
the difference between square of a and square of b is:-16
Now a=5, b=3

-----
Process exited after 2.131 seconds with return value 0
请按任意键继续. . .
    
```

图 4-1 程序运行结果示意图

4.2.2 程序修改替换

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

要求：（1）对这个例子程序进行调试修改，使之能够正确完成指定任务；

（2）用带参数的宏替换函数 max，来实现求最大数的功能。

/*实验 4-2 程序修改替换题程序*/

```

1  #include<stdio.h>
2  int main(void)
3  {
4      int a, b, c;
5      float d, e;
6      printf("Input three integers:");
7      scanf("%d %d %d",&a,&b,&c);
    
```

```

8     printf("\nThe maximum of them is %d\n",max(a,b,c));
9
10    printf("Input two floating point numbers:");
11    scanf("%f %f",&d,&e);
12    printf("\nThe sum of them is  %f\n",sum(d,e));
13    return 0;
14 }
15
16 int max(int x, int y, int z)
17 {
18     int m=z;
19     if (x>y)
20         if(x>z) m=x;
21     else
22         if(y>z) m=y;
23     return m;
24 }
25
26 float sum(float x, float y)
27 {
28     return x+y;
29 }

```

解答：（1）未对函数进行申明，应在 main 函数前插入:float sum(float x, float y);

（2）第 20 行 if 语句存在混乱。

修改部分如下所示：

```

1 int max(int x, int y, int z);
2 float sum(float x, float y);          //未声明函数
3 int max(int x, int y, int z)

```

```

4  {
5      int m=z;
6      if (x>y)
7          {
8              if(x>z) m=x;          //if 语句存在混乱
9          }
10     else
11         if(y>z) m=y;
12     return m;
13 }

```

运行结果示意图：

```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验4\2(1).exe
Input three integers:3 4 5
The maximum of them is 5
Input two floating point numbers:3.5 2.7
The sum of them is 6.200000
Process exited after 12.43 seconds with return value 0
请按任意键继续. . .

```

图 4-2 程序结果运行图

4.2.3 跟踪调试

下面程序利用 R 计算圆的面积 s，以及面积 s 的整数部分。现要求：

- (1) 修改程序，使程序编译通过且能运行；
- (2) 单步执行。进入函数 `integerl_fraction` 时，watch 窗口中 x 为何值？在返回 main 时，watch 窗口中 i 为何值？
- (3) 修改程序，使程序能输出面积 s 值的整数部分（要求四舍五入），不会输出错误信息 `assertion failed`。

```

1  /*实验 4-3 跟踪调试题程序利用 R 计算圆的面积 s*/
2  #define  R
3  int main(void)

```

```

4  {
5      float  r, s;
6      int s_integer=0;
7      printf ("Input a number: ");
8      scanf("%f",&r);
9      #ifdef  R
10         s=3.14159*r*r;
11         printf("Area of round is: %f\n",s);
12         s_integer=integer_fraction(s);
13         assert((s-s_integer)<0.5);
14         printf("The integer fraction of area is %d\n", s_integer);
15     #endif
16     return 0;
17 }
18
19 int integer_fraction(float x)
20 {
21     int i=x;
22     return i;
23 }

```

解答：

修改后的主要程序代码：

```

1  #define  R
2  #include <stdio.h>           //未加入头文件；
3  #include<assert.h>
4      assert((s-s_integer)<0.5);
5

```

输入 r = 5;进入函数 integerl_fraction 函数时，watch 窗口中的 x 的值为 78.539 在返回 main 时，watch 窗口中 i=78

(3)修改后主要代码如下：

```

1  int integer_fraction(float x)
2  {
3      int i=x;

```

```

4    if(x - i > 0.5)
5        i++;
6    return i;
7 }

```

4.2.4 程序设计

(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，要求编写程序用带参数的宏来计算三角形的面积。定义两个带参数的宏，一个用来求 s ，另一个用来求 $area$ 。

解答：

流程图：

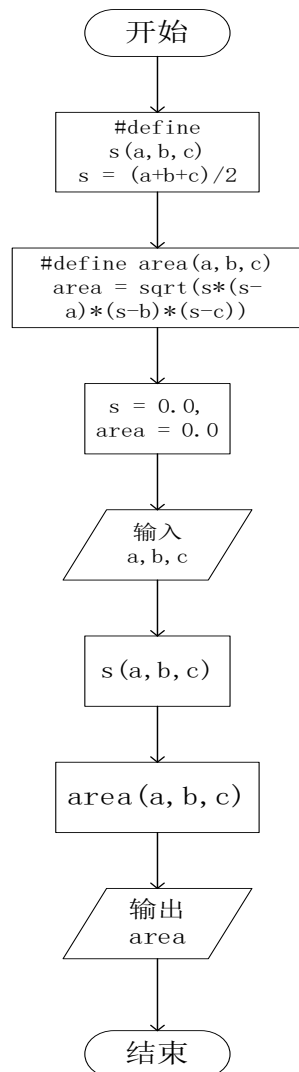
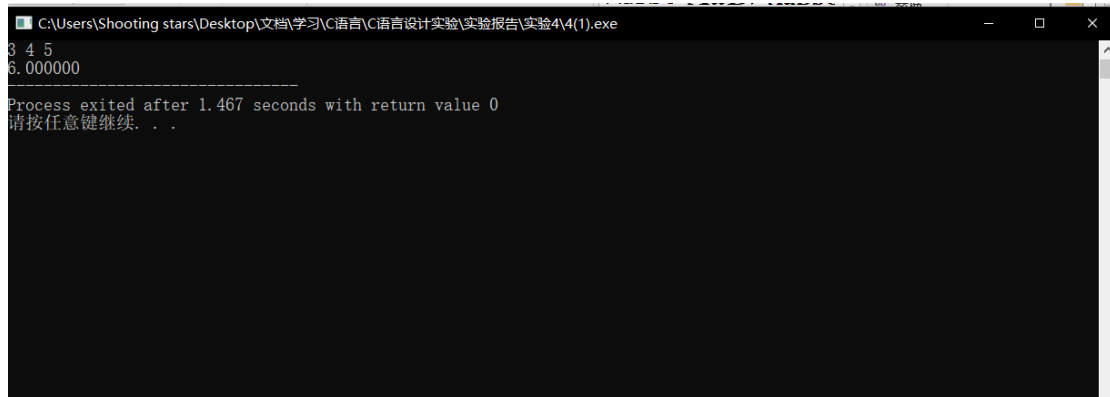


图 4-3 程序设计流程图

主要代码：

- 1 `#define s(a,b,c) s=(a+b+c)/2`
- 2 `#define area(a,b,c) area = sqrt(s*(s-a)*(s-b)*(s-c))`

运行结果：



```

C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验4\4(1).exe
3 4 5
6.000000
Process exited after 1.467 seconds with return value 0
请按任意键继续. . .

```

图 4-4 结果运行示意图

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

解答：

流程图：

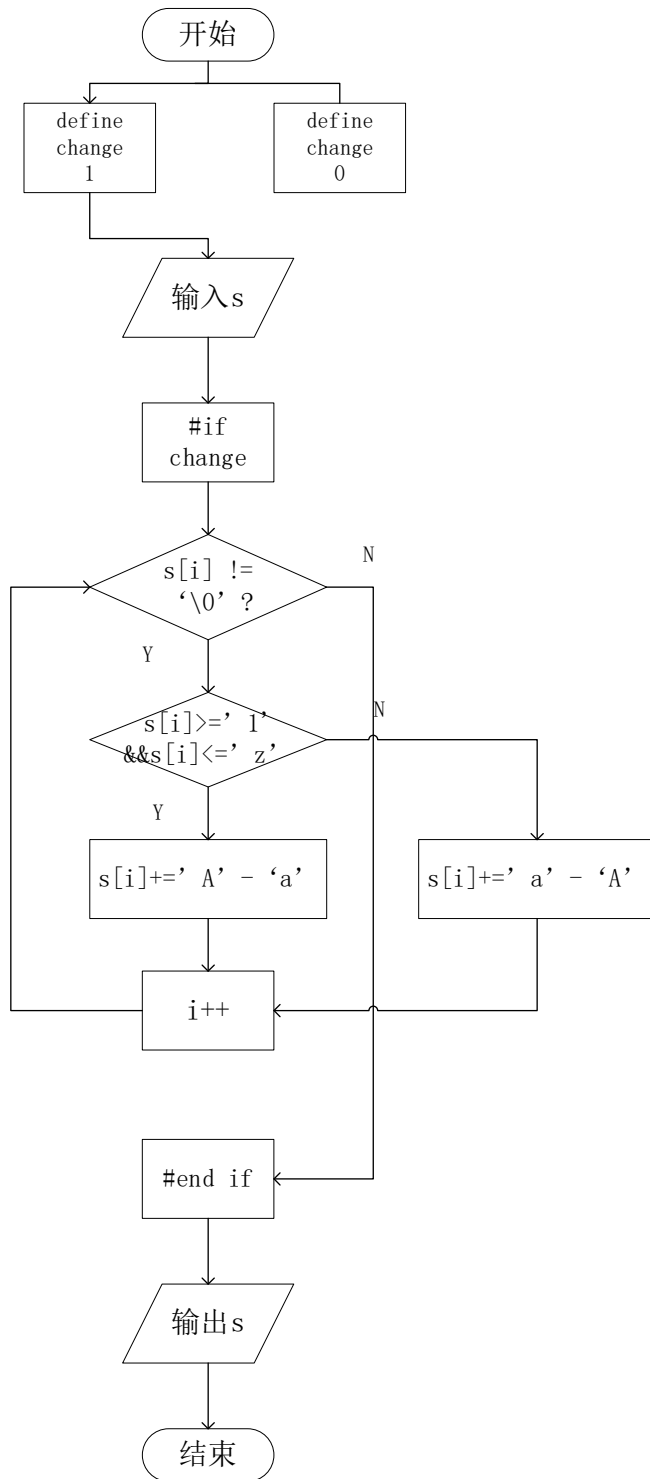


图 4-5 程序设计流程图

主要代码：

```

1  // #define CHANGE 1
2  #define CHANGE 0
3  #if CHANGE

```

```

4      while(s[i] != '\0')
5      {
6          if(s[i] >= 'a' && s[i] <= 'z')
7              s[i] += 'A' - 'a';
8          else s[i] += 'a' - 'A';
9          i++;
10     }
11     #endif

```

运行结果：

(1) #define CHANGE 1 时：

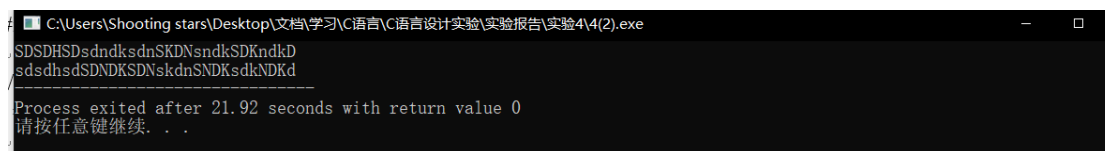


图 4-6 程序运行结果图

(2)#define CHANGE 0 时：

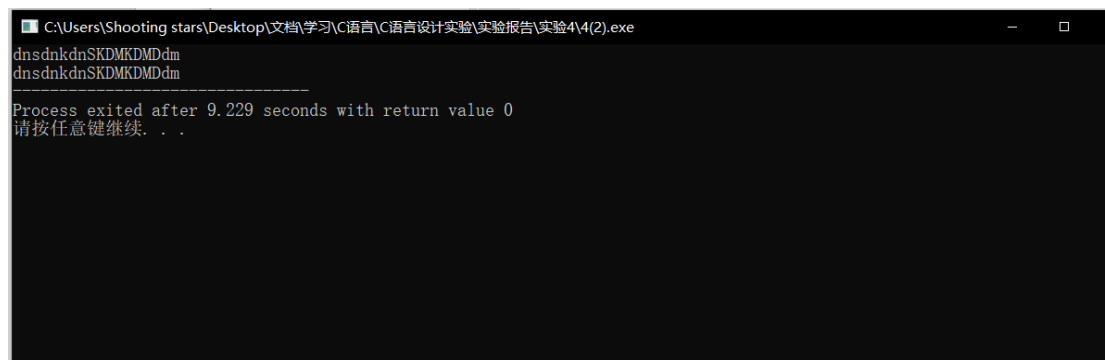


图 4-7 程序运行结果图

4.2.5 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，补充 file.h 头文件内容，然后编译和链接。然后运行最后生成的可执行文件。

```

1  /*源文件 file1.c 的内容*/
2      #include "file.h"
3  int x,y;          /* 外部变量的定义性说明 */
4  char ch;          /* 外部变量的定义性说明 */

```

```

5  int main(void)
6  {
7      x=10;
8      y=20;
9      ch=getchar();
10     printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);
11     func1();
12     return 0;
13 }
14
15 /*源文件 file2.c 的内容为: */
16 #include "file.h"
17 void func1(void)
18 {
19     x++;
20     y++;
21     ch++;
22     printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);
23 }

```

解答：

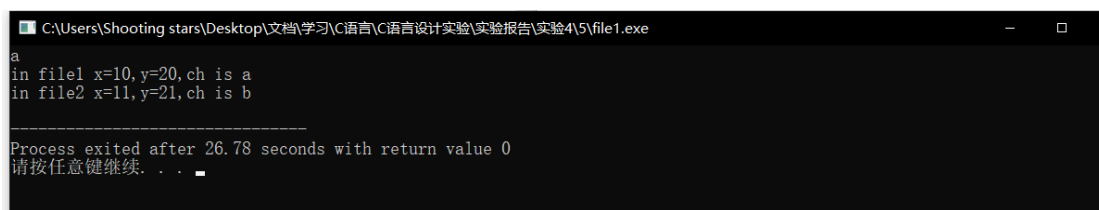
头文件：

```

1  //file.h
2  #ifndef _INCLUDE_FILE_2_
3  #define _INCLUDE_FILE_2_
4  #include "file2.c"
5  #endif
6  #include<stdio.h>
7  void func1(void);
8  extern int x,y;
9  extern char ch;

```

运行结果：



```
C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验4\5\file1.exe
a
in file1 x=10,y=20,ch is a
in file2 x=11,y=21,ch is b

-----
Process exited after 26.78 seconds with return value 0
请按任意键继续. . .
```

图 4-8 运行结果示意图

4.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在此次实验中，使用了宏定义，实验过程中遇到了宏定义展开后混乱的情况，通过添加括号的方式来避免此类问题，通过改变 `define` 的值来改变程序的运行在调试过程中十分重要且方便。头文件实验题操作过程中，遇到了知识点记忆不深刻的情况，通过复习课本查找资料最终解决了 `redefine` 的问题，通过自己动手查找资料，请教同学解决了一系列问题。

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 源程序改错与跟踪调试

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

Programming Language

ProgrammingLanguage Language

ProgramingLnuage

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

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

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

```
1  /*实验 5-1 程序改错与跟踪调试题程序*/
2  #include<stdio.h>
3  void strcat(char [],char []);
4  void strdelc(char [],char );
5  int main(void)
6  {
7  char a[]="Language", b[]="Programming";
8      printf("%s %s\n", b,a);
9  strcat(b,a);  printf("%s %s\n",b,a);
10     strdelc(b, 'a'); printf("%s\n",b);
11     return 0;
12 }
13 void strcat(char t[],char s[])
14 {
15     int i = 0,  j = 0;
16     while(t[i++]) ;
17     while((t[i++] = s[j++]) != '\0');
18 }
19 void strdelc(char s[], char c)
20 {
21     int j,k;
22     for(j=k=0; s[j] != '\0'; j++)
23         if(s[j] != c)    s[k++] = s[j];
24 }
```

解答：

- (1) 光标刚落在第二个 while 语句时，i 为 12， t[i] = '\0', i 应该减 1 回到第一个 '\0' 的位置，
- (2) 应该扩大两个字符串数组的长度，避免溢出
- (3) 应该在末尾加上 '\0'；

主要代码如下：

```
int main(void)
{
    char a[100]="Language", b[100]="Programming";           //增大字符串容
    量，防止溢出
    void strcat(char t[],char s[])
    {
        int i = 0, j = 0;
        while(t[i++] );
        i--;           //将 t[i]复位到'\0'的位置
        while((t[i++] = s[j++]) != '\0');
    }
    void strdelc(char s[], char c)
    {
        int j,k;
        for(j=k=0; s[j] != '\0'; j++)
            if(s[j] != c)
                s[k++] = s[j];
        s[k] = '\0'; //加入'\0'使字符串结束
    }
}
```

5.2.2 源程序完善和修改替换

(1) 下面的源程序用于求解瑟夫问题：M 个人围成一圈，从第一个人开始依次从 1 至 N 循环报数，每当报数为 N 时报数人出圈，直到圈中只剩一个人为止。①请在源程序中的下划线处填写合适的代码来完善该程序。

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

```
#define M 10
```

```
#define N 3
```

```

int main(void)
{
    int a[M], b[M];    /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int i, j, k;

    for(i = 0; i < M; i++)    /* 对圈中人按顺序编号 1—M */
        a[i] = i + 1;
    for(i = M, j = 0; i > 1; i--){
        /* i 表示圈中人个数，初始为 M 个，剩 1 个人时结束循环；j 表示当前报数人的位置 */
        for(k = 1; k <= N; k++)    /* 1 至 N 报数 */
            if(++j > i - 1) j = 0; /* 最后一个人报数后第一个人接着报，形成一个圈 */
            b[M-i] = j ? _____:_____; /* 将报数为 N 的人的编号存入数组 b */
        }
        if(j)
            for(k = --j; k < i; k++)    /* 压缩数组 a，使报数为 N 的人出圈 */
                _____;
    }
    for(i = 0; i < M-1; i++)    /* 按次序输出出圈人的编号 */
        printf(“%6d”, b[i]);
    printf(“%6d\n”, a[0]);    /* 输出圈中最后一个人的编号 */
    return 0;
}

```

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

解答：

① 第一处下划线 $a[j-1]$ ： $a[i-1]$;

第二处下划线 $a[k] = a[k+1]$;

② 更改后的主要代码如下：

```

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

```



```
    a[i] = i+1;
for(i = M, j = 1; i>1; i--){
    for(k = 1; k <= N;)
    {
        if(i == M)
            k++;
        j++;
        if(j >M)
            j = 1;
        if(a[j-1])
            k++;}
    b[M-i] = a[j-1];
    a[j - 1] = 0;
}
```

5.2.3 程序设计

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

解答：

流程图：

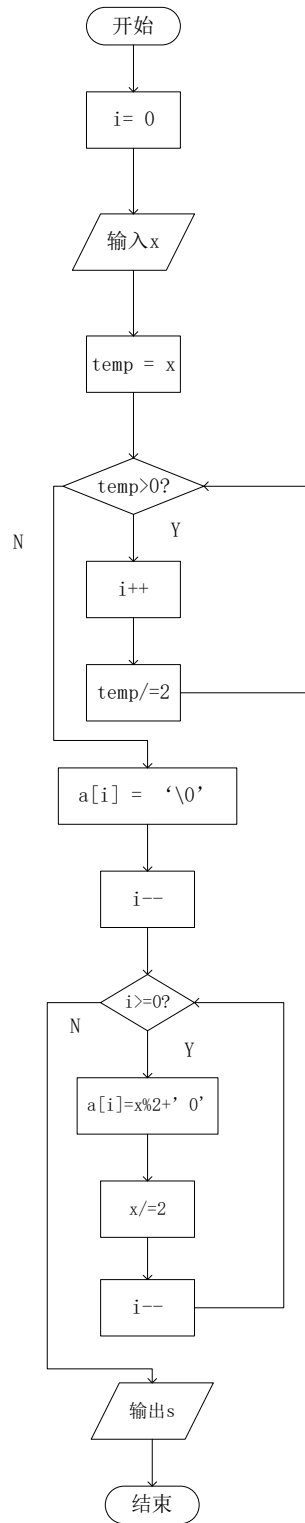


图 5-1 程序设计流程图

主要代码如下：

```

1      scanf("%d", &x);
2      temp = x;
3      while(temp>0)
    
```

```

4      {
5          i++;
6          temp /=2;
7      }
8      a[i] = '\0';
9      i--;
10     for(;i>=0; i--)
11     {
12         a[i] = x%2 + '0';
13         x /= 2;
14     }
15     printf("%s", a);
16     return 0;
17 }

```

运行结果：

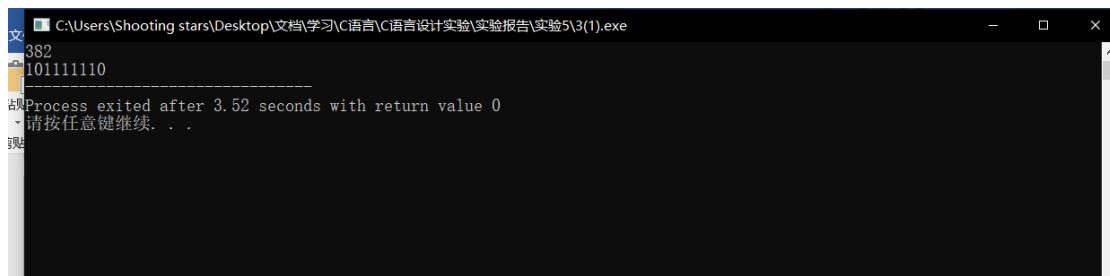


图 5-2 运行结果示意图

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

- ①输入 n 个学生的姓名和 C 语言课程的成绩。
- ②将成绩按从高到低的次序排序，姓名同时进行相应调整。
- ③输出所有学生的姓名和 C 语言课程的成绩。

解答：

流程图：

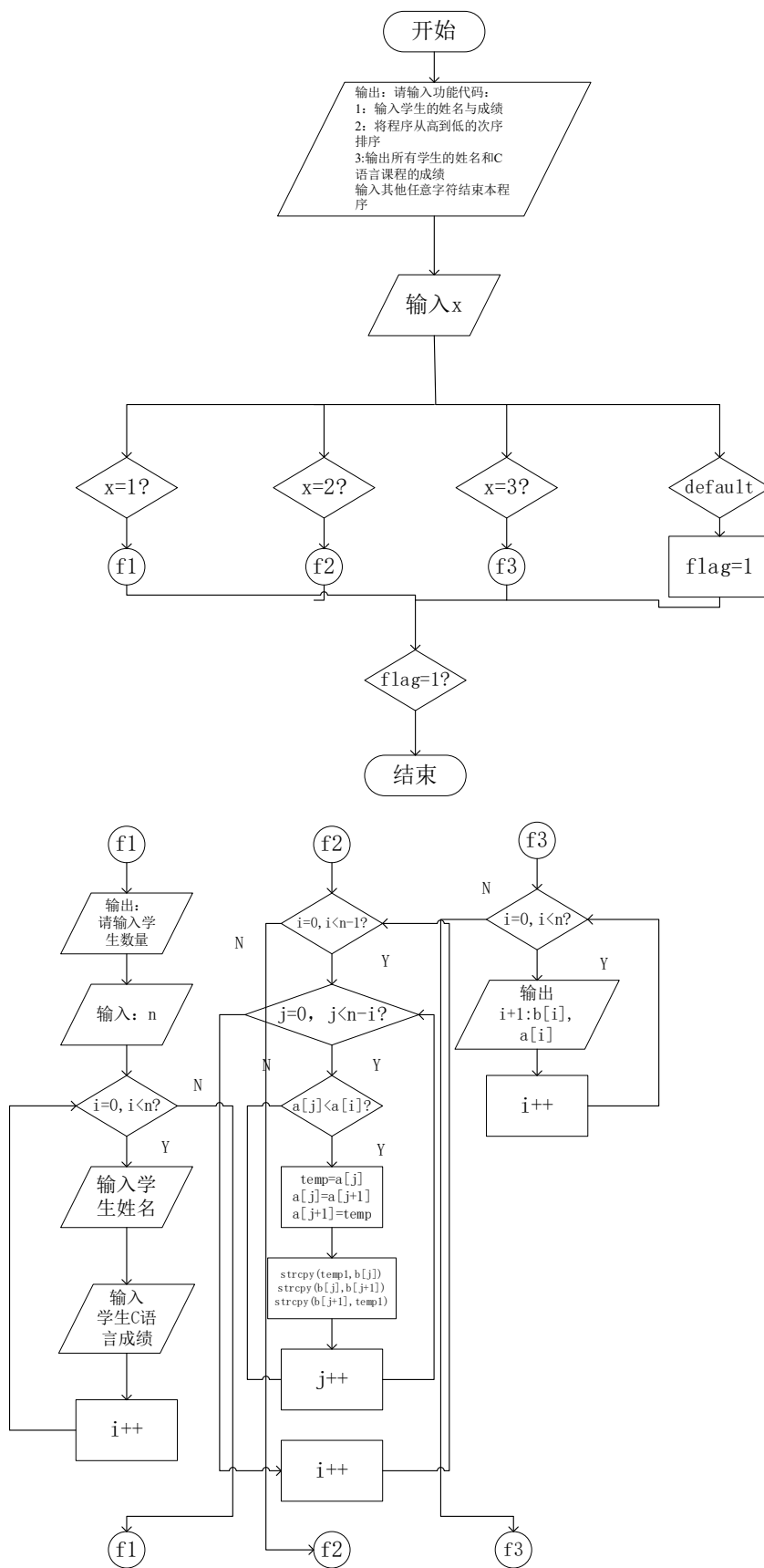
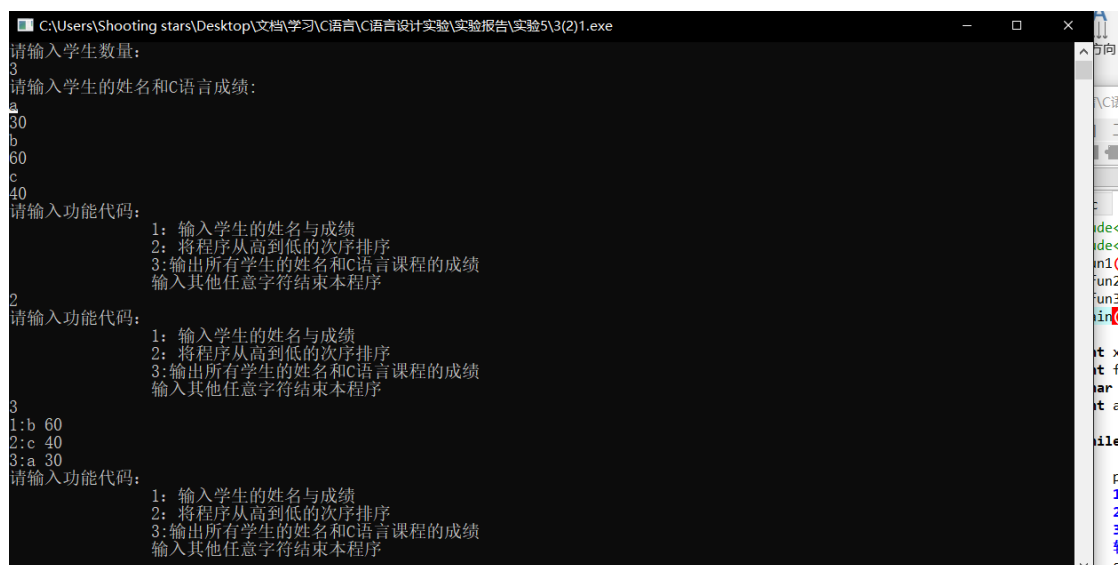


图 5-3 程序设计流程图

主要代码如下:

```
1 void fun2(int a[],char b[][100],int n){
2     int i,j,k,temp;
3     char temp1[100];
4     for(i = 0; i<n-1;i ++){
5         {
6             for(j = 0; j<n-i; j++)
7                 {
8                     if(a[j] < a[j+1])
9                         {
10                            temp = a[j];
11                            a[j] = a[j+1];
12                            a[j+1] = temp;
13                            strcpy(temp1, b[j]);
14                            strcpy(b[j], b[j+1]);
15                            strcpy(b[j+1], temp1);
16                        }
17                    }
18                }
19 }
```

运行结果:



The screenshot shows a Windows command prompt window titled "C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验5\3(2)1.exe". The program prompts the user for the number of students, then for each student's name and C language score. It then prompts for a function code. The user enters '3', which corresponds to the option "3: 输出所有学生的姓名和C语言课程的成绩" (Output the names and C language scores of all students). The program then outputs the following data:

```
1: b 60
2: c 40
3: a 30
```

The program then prompts for a function code again, and the user enters '3' again, which corresponds to the same option. The program then outputs the following data:

```
1: b 60
2: c 40
3: a 30
```

图 5-4 运行结果示意图

(3) 对程序设计第 (2) 题的程序增加查找功能：输入一个 C 语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和 C 语言课程的成绩；否则，输出提示 “not found!”。

解答：

查找部分代码如下：

```

1 void fun5(int a[],char b[][100], int n)
2 {
3     int j,flag= 0,mid,low,high;
4     int down,up;
5     int c[100]= {0},i = 0, k;
6     printf("请输入某数值:\n");
7     scanf("%d", &j);
8     high = 0, low = n-1;
9     while(high <= low)
10    {
11        mid = (low+high)/2;
12        if(j>a[mid])
13            low = mid-1;
14        else if(j <a[mid])
15            high = mid+1;
16        else
17            {
18                c[i] = mid;
19                i ++;
20                down = mid+1;
21                up = mid-1;
22                while(a[down] == j)
23                {
24                    c[i] = down;
25                    down ++,i++;
26                }
27                while(a[up] == j)
28                {

```

```

29             c[i] = up;
30             up --,i++;
31         }
32         flag = 1;
33     }
34     if(flag)
35         break;
36 }
37 if(flag)
38     for(k = 0; k<i;k++)
39         printf("%s %d\n", b[c[k]], a[c[k]]);
40     else printf("not found!\n");
41 }

```

运行结果:

The screenshot shows a Windows command prompt window titled "C:\Users\Shooting stars\Desktop\文档\学习\C语言\C语言设计实验\实验报告\实验5\3(2).exe". The program displays a menu with five options: 1: 输入学生的姓名与成绩, 2: 将程序从高到低的次序排序, 3: 输出所有学生的姓名和C语言课程的成绩, 4: 清除所有数据, 5: 查找某成绩值的同学 (查找前请先进行2:排序). The user has entered '3' to select option 3. The program then prompts for '请输入某数值:' and the user enters '70'. The program then prompts for '请输入功能代码:' and the user enters '70'. The program then displays the output: '1: c 70', '2: e 70', '3: a 50', '4: b 30', '5: d 10'. The program then prompts for '请输入功能代码:' and the user enters '70'. The program then displays the output: '1: 输入学生的姓名与成绩', '2: 将程序从高到低的次序排序', '3: 输出所有学生的姓名和C语言课程的成绩', '4: 清除所有数据', '5: 查找某成绩值的同学 (查找前请先进行2:排序)'. The program then prompts for '请输入其他任意字符结束本程序'.

图 5-5 程序运行结果

5.2.4 程序设计选做题

编写并上机调试运行能实现以下功能的函数和程序。

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

解答:

主要代码如下：

```
1 void strnins(char s[], char t[], int n)
2 {
3     int i;
4     i = 0;
5     while(t[i])
6     {
7         s[n] = t[i];
8         n++, i++;
9     }
10    s[n] = '\0';
11 }
```

程序运行：

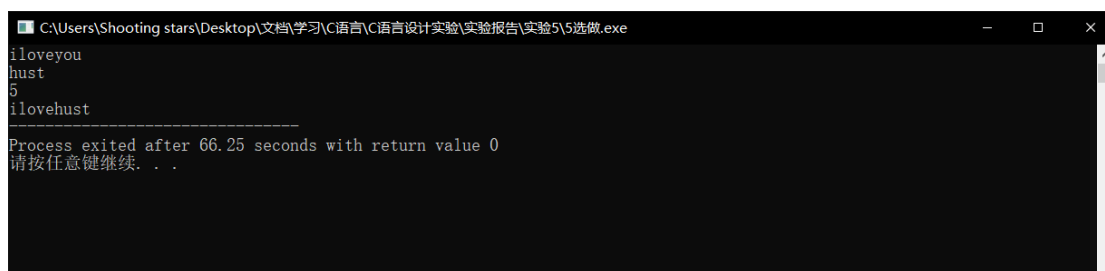


图 5-6 程序运行结果图

5.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

（1）在实验 1 中，通过多次 debug 与调试输出发现了程序中字符串溢出的情况，在编程过程中出现像数组溢出的情况，编译器并不会报错，但会在运行过程中出现许多问题，这就需要在日常编程过程中注意数组的大小。

（2）在实验 2 中，要学会读懂他人优秀的代码，理解代码实践过程中的意义，并能学习他人长处应用于自己的代码中。

6 指针实验

6.1 实验目的

- (1) 熟练掌握指针的说明、赋值、使用。
- (2) 掌握用指针引用数组的元素，熟悉指向数组的指针的使用。
- (3) 熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。
- (4) 掌握指针函数与函数指针的用法。
- (5) 掌握带有参数的 main 函数的用法。

6.2 实验内容

6.2.1 源程序改错题

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

Input a string:

programming ✓ （键盘输入）

programming

Input a string again:

language ✓ （键盘输入）

language

```
1  #include<stdio.h>
2  char *strcpy(char *, const char *);
3  int main(void)
4  {
5      char *s1, *s2, *s3;
```

```

6     printf("Input a string:\n", s2);
7     scanf("%s", s2);
8     strcpy(s1, s2);
9     printf("%s\n", s1);
10    printf("Input a string again:\n", s2);
11    scanf("%s", s2);
12    s3 = strcpy(s1, s2);
13    printf("%s\n", s3);
14    return 0;
15 }
16
17 /*将字符串 s 复制给字符串 t, 并且返回串 t 的首地址*/
18 char * strcpy(char *t, const char *s)
19 {
20     while(*t++ = *s++);
21     return (t);
22 }

```

解答:


(1) 错误修改:

①第五行的*s1,*s2 应改为 s1[100],s2[100]

②函数 strcpy 中应定义 char *t0 = t;

返回值应为 t0

(2) 错误修改后运行结果:



```

Input a string:
programming
programming
Input a string again:
langugae
langugae

```

图 6-1 运行结果示意图

6.2.2 源程序完善、修改替换题

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

①请在源程序中的下划线处填写合适的代码来完善该程序。

```
1  #include<stdio.h>
2  #include<_____>
3  #include<string.h>
4  #define N 4
5  /*对指针数组 s 指向的 size 个字符串进行升序排序*/
6  void strsort(char *s[], int size)
7  {
8      _____temp;
9      int i, j;
10     for(i=0; i<size-1; i++)
11         for (j=0; j<size-j-1; j++)
12             if (_____)
13                 {
14                     temp = s[j];
15                     _____;
16                     s[j+1] = temp;
17                 }
18 }
19 int main()
20 {
21     int i;
22     char *s[N], t[50];
23     for (i=0; i<N; i++)
24     {
25         gets(t);
26         s[i] = (char *)malloc(strlen(t)+1);
27         strcpy(_____);
```

```

28     }
29     strsort(______);
30     for (i=0; i<N; i++) puts(s[i]);
31     return 0;
32 }
    
```

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

解答：

①：

主要代码如下：

```

1 #include<stdlib.h>
2     char *temp;
3         if (strcmp(s[j], s[j+1]) > 0)
4             s[j] = s[j+1];
5             strcpy(s[i], t);
6     strsort(s,N);
    
```

运行结果图：

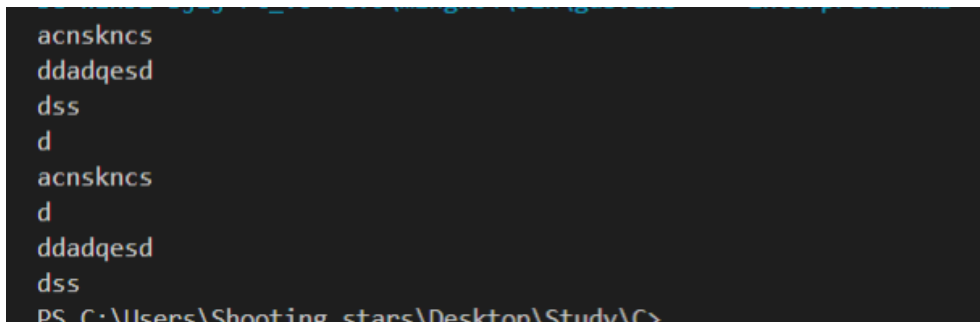


图 6-2 运行结果示意图

②：

代码如下：

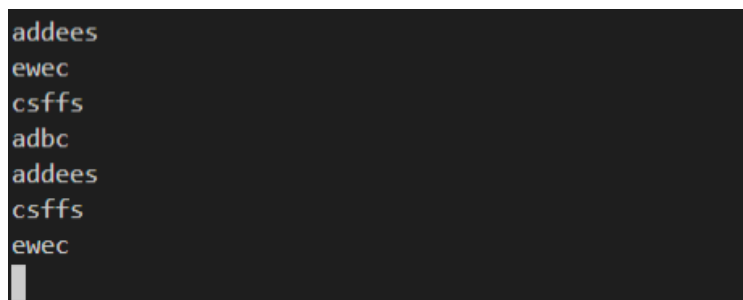
```

1 /*对指针数组 s 指向的 size 个字符串进行升序排序*/
2 void strsort(char **s, int size)
3 {
4     char **s0 = s;
    
```

```

5      char  *temp;
6      int  i, j;
7  for(i=0; i<size-1; i++)
8  {
9      s  =  s0;
10     for  (j=0; j<size-i-1; j++)
11     {
12         if  (strcmp(*s, *(s+1)) > 0)
13         {
14             temp  =  *s;
15             *s  =  *(s+1);
16             *(s+1)  =  temp;
17         }
18         s++;
19     }
20 }
21 }
    
```

运行结果：



```

addees
ewec
csffs
adbc
addees
csffs
ewec
    
```

图 6-3 运行结果示意图

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

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

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2✓ (键盘输入)

input the first string please!

the more you learn,✓ (键盘输入)

input the second string please!

the more you get. ✓ (键盘输入)

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

```

1
2 #include<stdio.h>
3 #include<string.h>
4 int main (void)
5 {
6     _____;
7     char a[80], b[80], *result;
8     int choice;
9     while(1)
10    {
11        do
12        {
13            printf("\t\t1 copy string.\n");
14            printf("\t\t2 connect string.\n");
15            printf("\t\t3 parse string.\n");
16            printf("\t\t4 exit.\n");
17            printf("\t\tinput a number (1-4) please.\n");
18            scanf("%d", &choice);
19        }while(choice<1 || choice>4);
20        switch(choice)
21        {
22            case 1:  p = strcpy;    break;
23            case 2:  p = strcat;    break;
24            case 3:  p = strtok; break;

```

```

25     case 4:  p = goto down;
26 }
27 getchar();
28 printf("input the first string please!\n");
29 _____;
30 printf("input the second string please!\n");
31 _____;
32 result = _____(a, b);
33 printf("the result is %s\n", result);
34 }
35     down:
36     return 0;
37 }

```

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

解答：①

主要代码如下：

```

1     char  *(*p)(char  *,const  char*);
2     gets(a);
3     gets(b);
4     result  =  p(a,  b);

```

运行结果如下：

```

1 copy string.
2 connect string.
3 parse string.
4 exit.
input a number (1-4) 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.
1 copy string.
2 connect string.
3 parse string.
4 exit.
input a number (1-4) please.

4
PS C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\experiment reports\experiment6>

```

图 6-4 运行结果示意图

②：主要代码如下：

```

1      char >(*p)(char *, const char*);
2      char >(*s[3])(char *, const char *);
3      s[0] = strcpy,s[1] = strcat,s[2] = strtok;
4      char a[80], b[80], *result;
5      int choice;
6      while(1)
7      {
8          do
9          {
10             printf("\t\t1 copy string.\n");
11             printf("\t\t2 connect string.\n");
12             printf("\t\t3 parse string.\n");
13             printf("\t\t4 exit.\n");
14             printf("\t\tinput a number (1-4) please.\n");
15             scanf("%d", &choice);
16             }while(choice<1 || choice>4);
17         if(choice != 4)
18             p = s[choice-1];
19         else
20             goto down;

```


运行结果：

```

1 copy string.
2 connect string.
3 parse string.
4 exit.
input a number (1-4) 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.
1 copy string.
2 connect string.
3 parse string.
4 exit.
input a number (1-4) please.

4
PS C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\experiment reports\experiment6>

```

图 6-5 运行结果示意图

6.2.3 跟踪调试题

请按下面的要求对源程序进行操作，并回答问题和排除错误。

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

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

```

1  #include "stdio.h"
2  char *strcpy(char *,char *);
3  void main(void)
4  {
5      char a[20],b[60]="there is a boat on the lake.";
6      printf("%s\n",strcpy(a,b));
7
8  }
9  char *strcpy(char *s,char *t)
10 {
11     while(*s++=*t++)
12         ;
13     return (s);
14 }

```

解答：

(1) 进入 strcpy 时, s 为\b 随机值, 返回 main 时, s 为\0;

(2) 函数 strcpy 改为:

```
char *strcpy(char *s,char *t)
{
    char *s0 = s;
    while(*s++=*t++);
    return (s0);
}
```

6.2.4 编程设计题

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

解答:

流程图:

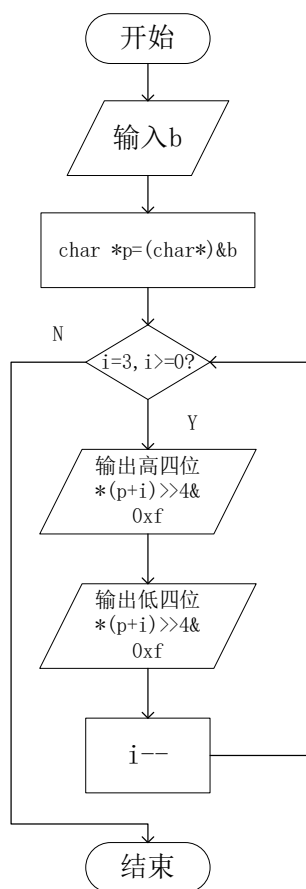


图 6-6 程序设计流程图

代码如下：

```

1  #include <stdio.h>
2  int main()
3  {
4  int i;
5      long b;
6      char *p;
7      scanf("%d", &b);
8      p =(char *)&b;
9      for(i = 3;i>=0; i--)
10     {
11         printf("第%d 个字节:\n", 4-i);
12         printf("高四位:%x\n", *(p+i)>>4 &0xf);
13         printf("低四位:%x\n", *(p+i)&0xf);
14     }
15
16     return 0;
17 }

```

运行结果示意图：

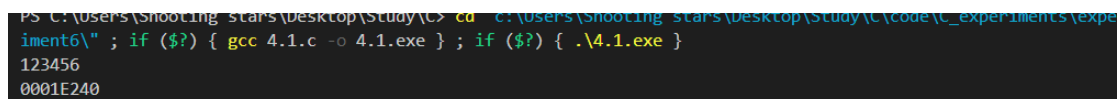


图 6-7 运行结果示意图

(2) 旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转 90° 。提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在 `main` 函数中输入图像矩阵的行数 `n` 和列数 `m`，接下来的 `n` 行每行输入 `m` 个整数，表示输入的图像。输出原始矩阵逆时针旋转 90° 后的矩阵。例如，输入：

```

2  3
1  5  3
3  2  4

```

则输出：

3 4

5 2

1 3

解答：

流程图：

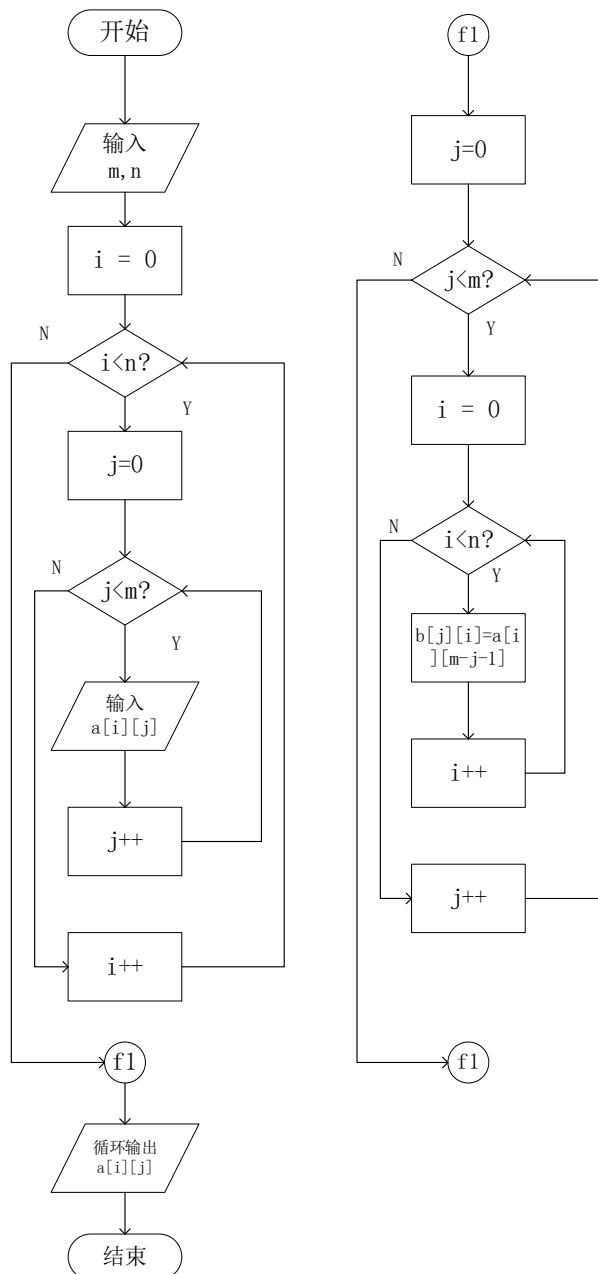
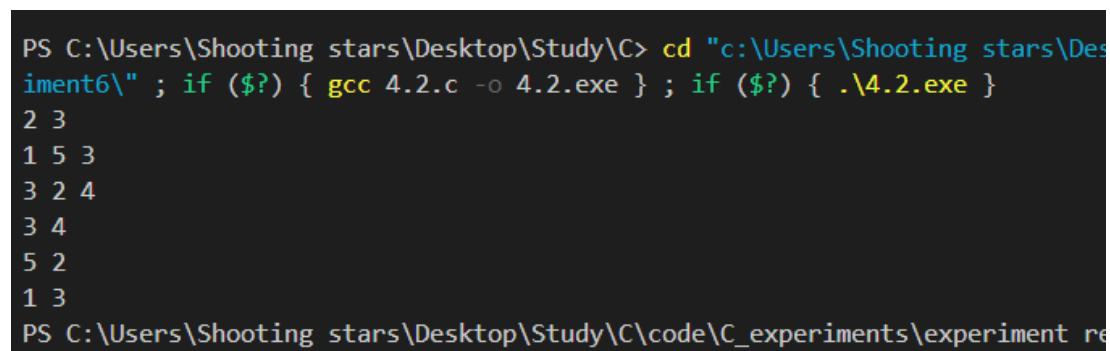


图 6-8 程序设计流程图

主要代码如下：

```
1 void circle(int b[m][n], int a[n][m])
2 {
3     int i,j;
4     for(j = 0; j<m; j++)
5         for(i = 0; i<n; i++)
6             b[j][i] = a[i][m-j-1];
7 }
```

运行结果：



```
PS C:\Users\Shooting stars\Desktop\Study\C> cd "c:\Users\Shooting stars\Desktop\Study\C" ; if ($?) { gcc 4.2.c -o 4.2.exe } ; if ($?) { .\4.2.exe }
2 3
1 5 3
3 2 4
3 4
5 2
1 3
PS C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\experiment re
```

图 6-9 运行结果示意图

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

解答：

流程图：

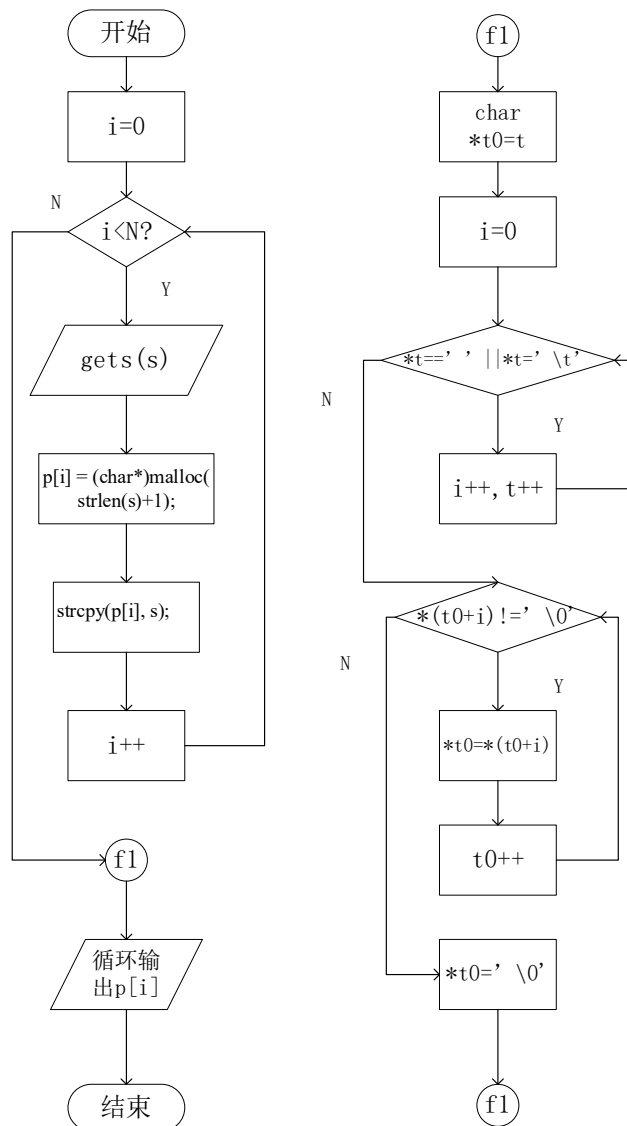


图 6-10 程序设计流程图

主要代码如下：

```

1 void delete(char *t)
2 {
3     char *t0 = t;
4     int i = 0;
5     while(*t==' ' || *t == '\t')
6         i++,t++;
7     while (*(t0+i)!= '\0')
8     {
9         *t0 = *(t0+i);
10        t0++;
    
```

```

11     }
12     *t0 = '\0';
13 }
    
```

运行结果：



```

3      abc
      s
d ssad
abc
s
d ssad
    
```

图 6-11 运行结果示意图

(4) 编写 8 个任务函数，一个 scheduler 调度函数和一个 execute 执行函数。

仅在 main 函数中调用 scheduler 函数，scheduler 函数要求用最快的方式调度执行用户指定的任务函数。

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

②scheduler 函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350 并回车，则 scheduler 函数一次调度选择 task1, task3, task5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给 execute 函数并调用 execute 函数。

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

例如，当输入 13607122 并回车，程序运行结果如下：

```

task1 is called!
task3 is called!
task6 is called!
task0 is called!
task7 is called!
task1 is called!
task2 is called!
    
```

task2 is called!

解答：

流程图：

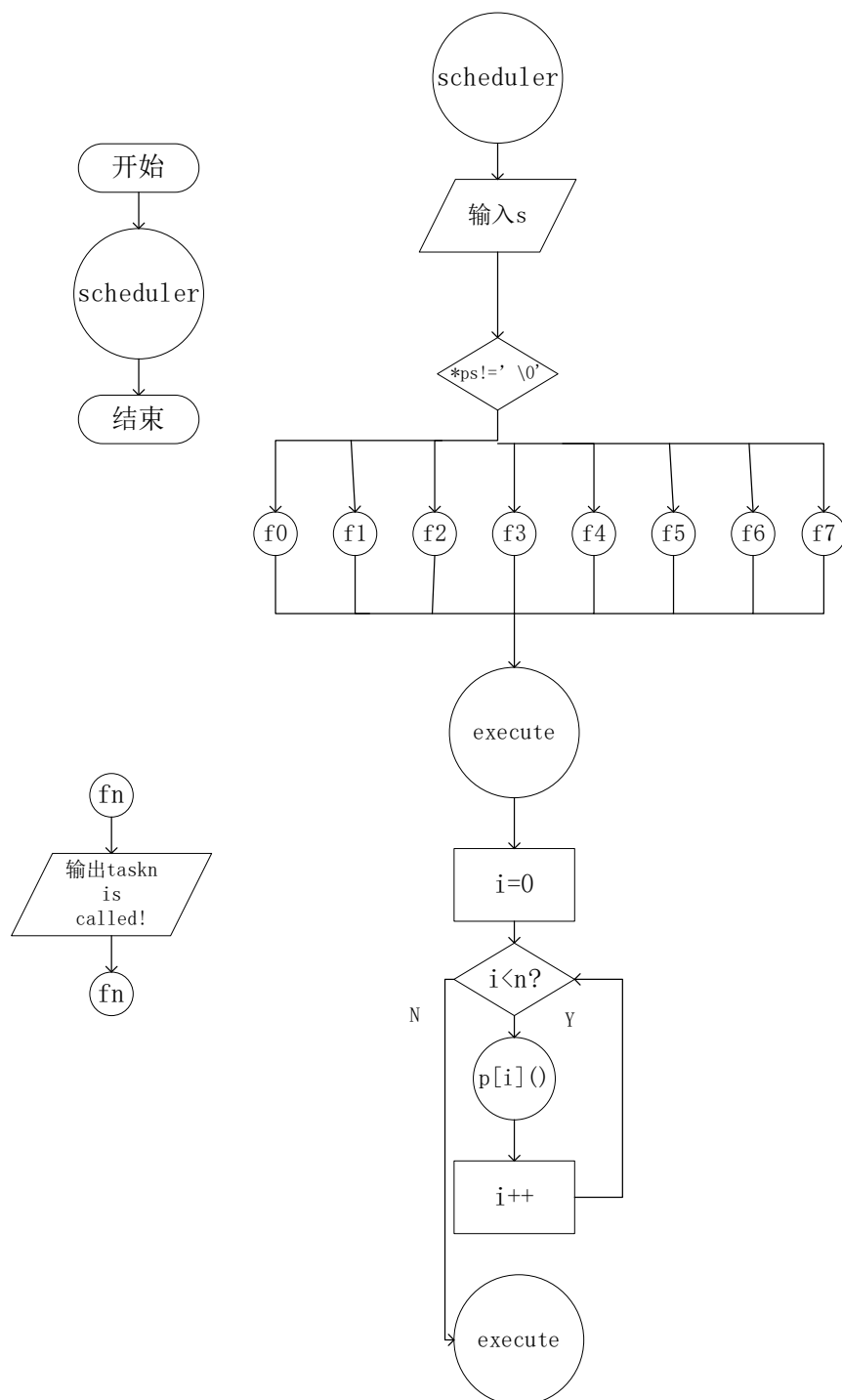


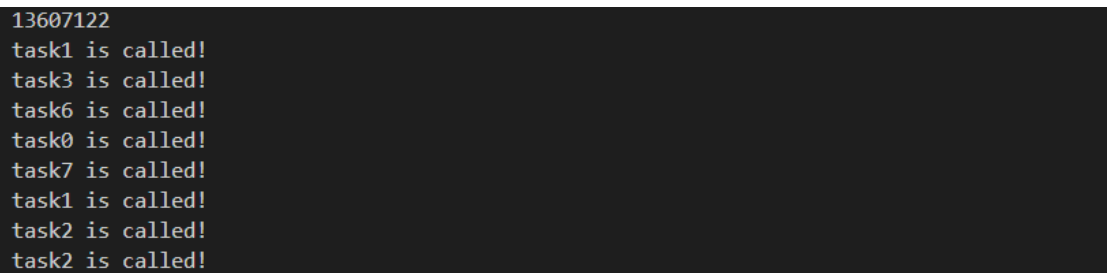
图 6-12 程序设计流程图

代码如下：


```
void scheduler(void)
{
    while(*ps != '\0')
    {
        switch (*ps)
        {
            default:printf("Error!");
            flag = 0;
            break;
        }
        assert(flag);
        i++;
        ps++;
    }
    execute(p, i);
}

void execute(void (*p[50])(void), int n)
{
    int i;
    for(i = 0; i<n; i++)
        (p[i])();
}
```

运行结果：



```
13607122
task1 is called!
task3 is called!
task6 is called!
task0 is called!
task7 is called!
task1 is called!
task2 is called!
task2 is called!
```

图 6-13 运行结果示意图

6.2.5 选做题

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

如：12345678912345678912.1234567891 + 98765432109876543210.0123456789

解答:

主要代码如下:

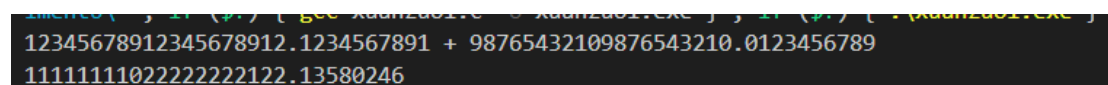
```
//初始化
for(i = 0; i<50;i++)
b[i] = '0';
b[30] = '.';
//小数部分相加,但第三十一位暂时不加
len1 = (len[1] >= len[3]) ? len[1] : len[3];
temp = len1;
t1 = &c2[len1 - 1], t2 = &d2[len1 - 1], p = &b[30+len1];
while(len1 > 1)
{
    n = *t1-- + *t2-- + *p - '0';
    *p-- = n%10 + '0';
    *p += n/10;
    len1--;
}
//第三十一位相加, 进位给个位
n = *t1 + *t2 + *p - '0';
*p = n %10 + '0';
*(p-2) += n/10;
//整数部分相加;
len2 = (len[0] >= len[2]) ? len[0] : len[2];
t1 = &c1[len2 - 1], t2 = &d1[len2 - 1], p= &b[29];
while(len2 > 0)
{
    n = *t1-- + *t2-- + *p - '0';
    *p-- = n%10 + '0';
    *p += n/10;
    len2--;
}
//构造字符串并输出
p = &b[temp + 30];
while(*p -- == '0');
```

```

*(p+1) = '\0';
p = b;
while(*p++ == '0');
printf("%s", p);
return 0;
}

```

运行结果：



```

12345678912345678912.1234567891 + 98765432109876543210.0123456789
11111111022222222122.13580246

```

图 6-14 运行结果示意图

(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：

代码如下：

```

char *(*p[2])(const char *, const char *);
p[0] = strcmp;
p[1] = strstr;
ret1= p[0](s,t);
ret2 = p[1](s,t);

```

6.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在本次实验中遇到了常量区数据不能修改的问题，在对字符串进行赋值时不能使用=，而应该使用 `strcpy` 函数进行操作，还遇到了动态分配内存的问题，在编程时尽量使用函数对代码进行封装，这样可以使得代码更简洁，可读性，实用性更高。

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)

表 7-1 各表达式的值

序号	表达式	计算值	验证值
1	$(++p) \rightarrow x$	100	100
2	$p++$, $p \rightarrow c$	B	B
3	$*p++ \rightarrow t$, $*p \rightarrow t$	'x'	'x'
4	$*(++p) \rightarrow t$	'x'	'x'
5	$*++p \rightarrow t$	'v'	'v'

6	++*p->t	‘V’	‘V’
---	---------	-----	-----

7.2.2 源程序修改替换题

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

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

源程序如下：

```
#include "stdio.h"
#include "stdlib.h"
struct s_list{
int data; /* 数据域 */
struct s_list *next; /* 指针域 */
};
void create_list (struct s_list *headp,int *p);
void main(void)
{
    struct s_list *head=NULL,*p;
    int s[]={1,2,3,4,5,6,7,8,0}; /* 0 为结束标记 */
    create_list(head,s); /* 创建新链表 */
    p=head; /*遍历指针 p 指向链头 */
    while(p){
        printf("%d\t",p->data); /* 输出数据域的值 */
        p=p->next; /*遍历指针 p 指向下一结点 */
    }
    printf("\n");
}
void create_list(struct s_list *headp,int *p)
{
    struct s_list * loc_head=NULL,*tail;
    if(p[0]==0) /* 相当于*p==0 */
        ;
```

```

else { /* loc_head 指向动态分配的第一个结点 */
    loc_head=(struct s_list *)malloc(sizeof(struct s_list));
    loc_head->data=*p++; /* 对数据域赋值 */
    tail=loc_head; /* tail 指向第一个结点 */
    while(*p){ /* tail 所指结点的指针域指向动态创建的结点 */
        tail->next=(struct s_list *)malloc(sizeof(struct s_list));
        tail=tail->next; /* tail 指向新创建的结点 */
        tail->data=*p++; /* 向新创建的结点的数据域赋值 */
    }
    tail->next=NULL; /* 对指针域赋 NULL 值 */
}

headp=loc_head; /* 使头指针 headp 指向新创建的链表 */

```

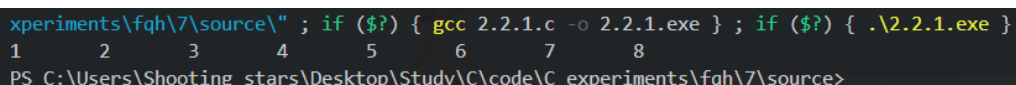
解答:

(1) 错误修改:

- 1) 第 7 行: 更改函数声明为 void create_list(struct s_list **headp, int *p);
- 2) 第 12 行: 将 head 的地址传入函数, 应改为 create_list(&head,s);
- 3) 第 20 行: 函数原型改为 void create_list(struct s_list **headp,int *p)
接收 head 的地址从而通过间访来改变 main 中 head 的值
- 4)第 36 行: 通过间访来 headp 改变 main 中 head,应改为: *headp =

loc_head;

(2) 错误修改后运行结果:



```

xperiments\fqh\7\source" ; if ($?) { gcc 2.2.1.c -o 2.2.1.exe } ; if ($?) { .\2.2.1.exe }
1      2      3      4      5      6      7      8
PS C:\Users\Shooting_stars\Desktop\Study\C\code\C_experiments\fqh\7\source>

```

图 7-1 程序运行示意图

(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:

代码如下:#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);/*更改函数声明
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 */ /*将 head 的地址传入函数
    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)/*更改函数原型，接收 head
的地址
{
    int i = 0;
    struct s_list *loc_head, *tail = NULL;
    while(*p != 0)
    {
        loc_head = (struct s_list*)malloc(sizeof(struct s_list));
        loc_head->data = *p++;
        loc_head->next = tail;
        tail = loc_head; }
    *headp = loc_head;
}

```

运行结果示意图：

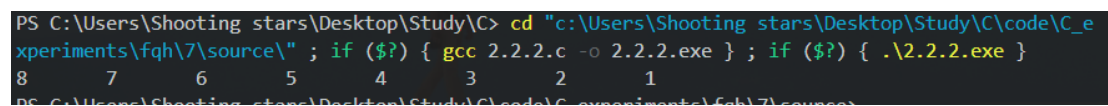


图 7-2 运行结果示意图

7.2.3 程序设计

(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，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：

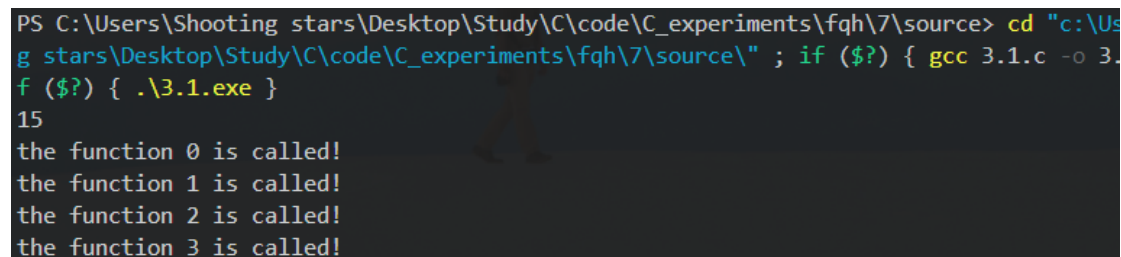
```
void f0( int n)
{
    printf("the function %d is called!\n",n);
}
```

解答：

主要代码如下：

```
1  struct  bits
2  {
3      unsigned  bit0:1;
4      unsigned  bit1:1;
5      unsigned  bit2:1;
6      unsigned  bit3:1;
7      unsigned  bit4:1;
8      unsigned  bit5:1;
9      unsigned  bit6:1;
10     unsigned  bit7:1;
11 };
```

运行结果：



```
PS C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\fqh\7\source> cd "C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\fqh\7\source\" ; if ($?) { gcc 3.1.c -o 3.1.exe }
15
the function 0 is called!
the function 1 is called!
the function 2 is called!
the function 3 is called!
```

图 7-3 运行结果示意图

(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：

流程图：

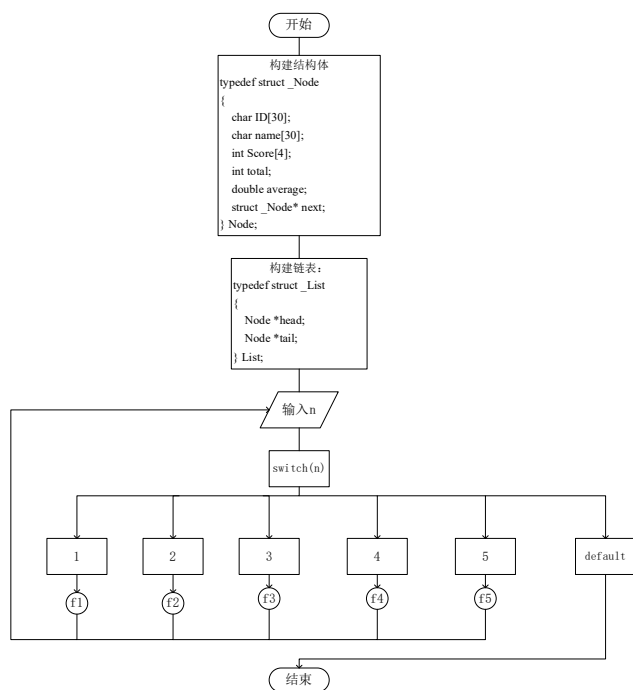


图 7-4 程序设计流程图（1）

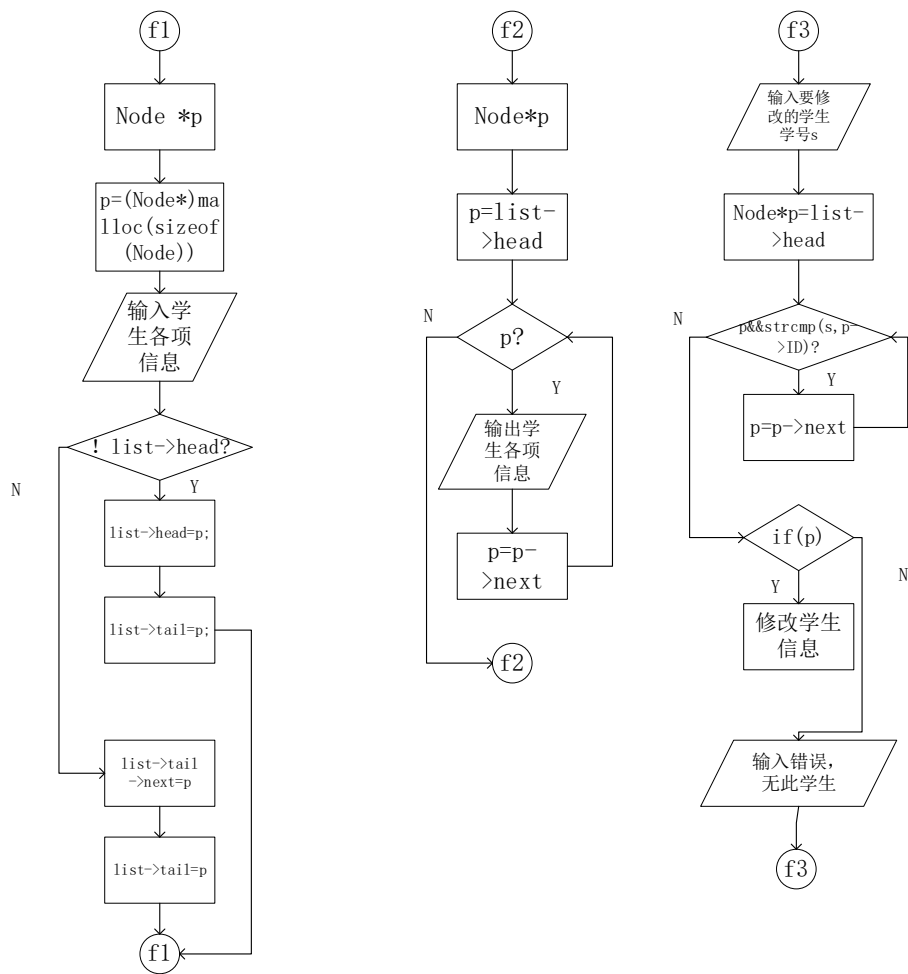


图 7-5 程序设计流程图 (2)

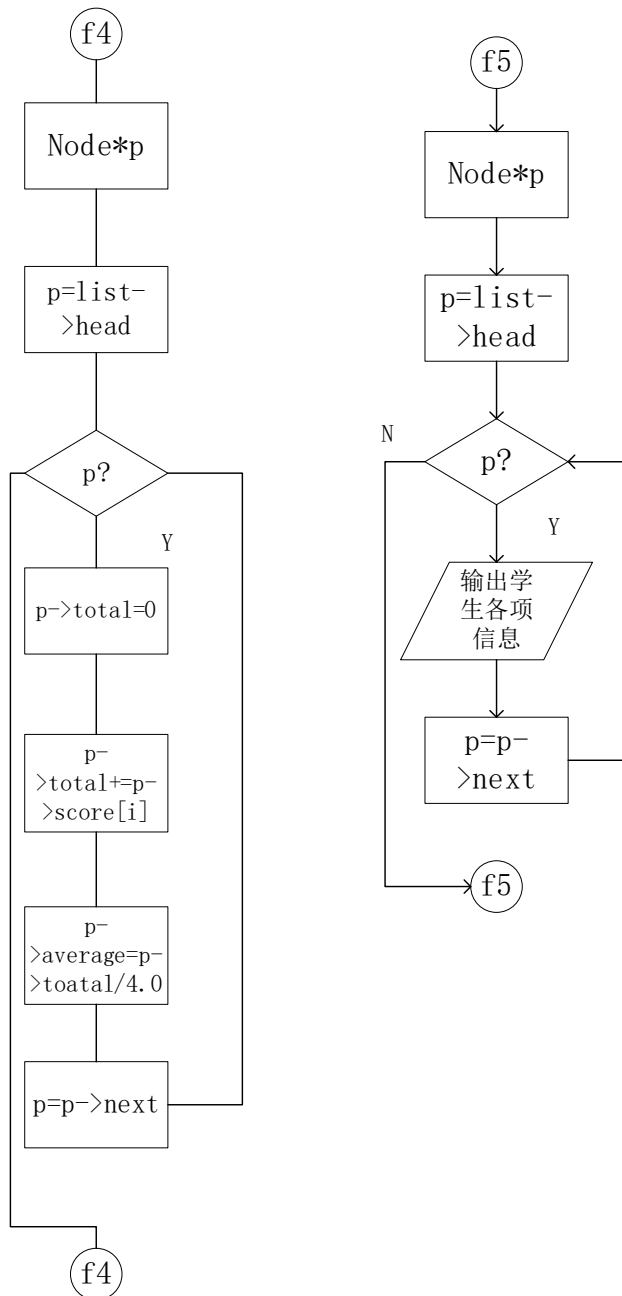


图 7-6 程序设计流程图（3）

1 }

运行结果：

```

***** 班级成绩系统 *****
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
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
输入其他字符退出本系统
请输入功能代码:
5
第1个学生学号:u001
姓名:qqq
英语成绩:      10
高等数学成绩:  10
普通物理成绩:  10
C语言程序设计成绩:10
总成绩为:      40
平均成绩为:    10.00
第2个学生学号:u002
姓名:www
英语成绩:      8
高等数学成绩:  8
普通物理成绩:  8
C语言程序设计成绩:8
总成绩为:      32
平均成绩为:    8.00
第3个学生学号:u003
姓名:www
英语成绩:      6
高等数学成绩:  6
普通物理成绩:  6
C语言程序设计成绩:6
总成绩为:      24
平均成绩为:    6.00
*****
***** 班级成绩系统 *****

```

图 7-7 运行结果示意图

7.2.4 选做题

(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。

(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。

(3) 采用双向链表重做编程设计题第(2)题。

解答：

(1) :

排序部分代码如下：

```

1 void sort(List *list)
2 {
3     Node *p, *temp;
4     temp = (Node *)malloc(sizeof(Node));
5     int len, i,j,k;
6     //统计学生
7     for(p = list->head,len = 0; p ; p = p->next,len++);
8     //冒泡排序
9     for(i = 0, p = list->head; i<len - 1;i++, p= p->next)
10    {
11        for(j = 0, p = list->head; j<len - i-
12    1;j++, p= p->next)
13        {
14            if(p->average > p->next->average)
15            {
16                //学号交换
17                strcpy(temp->ID, p->ID);
18                strcpy(p->ID, p->next->ID);
19                strcpy(p->next->ID, temp->ID);
20                //姓名交换
21                strcpy(temp->name, p->name);
22                strcpy(p->name, p->next->name);
23                strcpy(p->next->name, temp->name);
24                //各项成绩交换
25                for(k = 0; k<4; k++)
26                {
27                    temp->Score[k] = p->Score[k];
28                    p->Score[k] = p->next->Score[k]
29 ;

```

```

30             p->next->Score[k] = temp->Scor
31 e[k];
32         }
33         //总成绩交换
34         temp->total = p->total;
35         p->total = p->next->total;
36         p->next->total = temp->total;
37         //平均成绩交换
38         temp->average = p->average;
39         p->average = p->next->average;
40         p->next->average = temp->average;
41     }
42 }
43 }
44 free(temp);
45 }

```

(2)

排序部分代码如下：

```

1 void sort(List *list)
2 {
3     average(list);
4     Node *p,*prior,*after,*q;
5     int len, i,j,k;
6     //统计学生
7     for(p = list->head,len = 0; p ; p = p->next,len++);
8     //冒泡排序
9     for(i = 0; i<len-1; i++)
10    {
11        p = list->head;
12        if(p->average > p->next->average)
13        {
14            after = p->next->next;
15            list->head = p->next;

```

```

16         list->head->next = p;
17         p->next = after;
18     }
19     for(j = 1, prior = list->head, p = prior->next; j<len
20 -i-1;j++)
21     {
22         if(p->average > p->next->average)
23         {
24             q = p->next;
25             after = q->next;
26             prior->next = q;
27             q->next = p;
28             p->next = after;
29             prior = prior->next;
30         }
31         else
32         {
33             p = p->next;
34             prior = prior->next;
35         }
36     }
37 }
38
39 }

```

(3)

结构声明部分:

```

1 typedef struct _Node
2 {
3     char ID[30];
4     char name[30];
5     int Score[4];
6     int total;
7     double average;

```

```
8      struct _Node*  prior;
9      struct _Node*  next;
10 } Node;
11 链表创建:
12      if(!(list->head))
13      {
14          list->head = p;
15          list->tail = p;
16          p ->prior = list->tail;
17      }
18      else
19      {
20          list->tail->next = p;
21          p->prior = list->tail;
22          list->tail = p;
23      }
```

7.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在本次实验中主要应用到了结构，联合和字段的知识，在使用结构时需要主要各个运算符的优先级，并对其进行判断才能读懂他人的程序，使自己的程序更加简洁明了。在使用结构构建链表过程中，要灵活运用指针进行操作。在必要时可以多创建一些指针来进行灵活的操作。在对链表排序是用交换数据域与交换指针域两种方法，当数据域较少时，选择交换数据域更加方便，而当数据域较多时，往往选择交换指针域使代码更加简洁，运行效率更高。因此掌握好两种方法并灵活应用能够使程序更加高效。

8 文件操作实验

8.1 实验目的

- (1) 熟悉文本文件和二进制文件在磁盘中的存储方式;
- (2) 熟练掌握流式文件的读写方法。

8.2 实验内容

8.2.1 文件类型的程序验证题

设有程序:

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  int main(void)
4  {
5      short a=0x253f,b=0x7b7d;
6      char ch;
7      FILE *fp1,*fp2;
8      fp1=fopen("d:\\abc1.bin","wb+");
9      fp2=fopen("d:\\abc2.txt","w+");
10     fwrite(&a,sizeof(short),1,fp1);
11     fwrite(&b,sizeof(short),1,fp1);
12     fprintf(fp2,"%hx %hx",a,b);
13
14     rewind(fp1); rewind(fp2);
15     while((ch = fgetc(fp1)) != EOF)
16         putchar(ch);
17     putchar('\n');
18
19     while((ch = fgetc(fp2)) != EOF)
20         putchar(ch);
21     putchar('\n');
22
23     fclose(fp1);
24     fclose(fp2);
25     return 0;
26 }
```

- (1) 请思考程序的输出结果，然后通过上机运行来加以验证。
- (2) 将两处 `sizeof(short)` 均改为 `sizeof(char)` 结果有什么不同，为什么？
- (3) 将 `fprintf(fp2,"%hx %hx",a,b)` 改为 `fprintf(fp2,"%d %d",a,b)` 结果有什么不同。

解答：

(1) ? (3f) %(25) } (7d) { (7b)

253f 7d7d

(2) ? (文本为 3f) } (文本为 7D)

以 char 型存入进行了类型转换，只保留了低字节。

(3) 9535 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`

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。

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  int main(int argc, char* argv[])
4  {
5      char ch;
6      FILE *fp;
7      if(argc!=2){
8          printf("Arguments error!\n");
9          exit(-1);
10     }
11     if((fp=fopen(argv[1],"r"))==NULL){          /* fp 指向 filename */
12         printf("Can't open %s file!\n",argv[1]);
13         exit(-1);
14     }
15     while(ch=fgetc(fp)!=EOF)                    /* 从 filename 中读字符 */
16         putchar(ch);                            /* 向显示器中写字符 */
17     fclose(fp);                                /* 关闭 filename */

```

```
18     return 0;
19 }
```

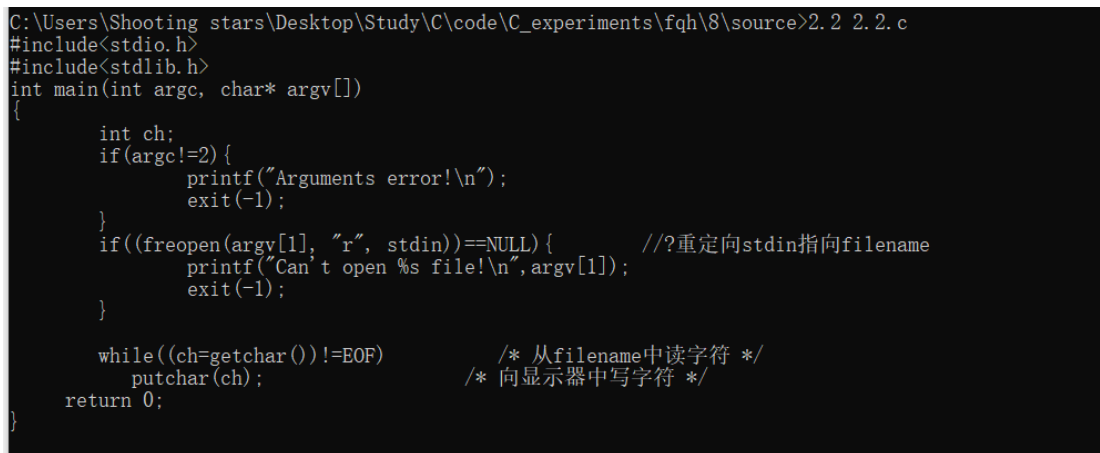
(2) 用输入输出重定向 `freopen` 改写 `main` 函数。

1.解答:

(1) 错误修改:

- 1) 第 5 行 `ch` 应定义为 `int`
- 2) 第 16 行应该为 `while((ch=fgetc(fp))!=EOF)`

(2) 错误修改后运行结果:



```
C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\fqh\8\source>2.2 2.2. c
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char* argv[])
{
    int ch;
    if(argc!=2){
        printf("Arguments error!\n");
        exit(-1);
    }
    if((freopen(argv[1], "r", stdin))==NULL){          /*重定向stdin指向filename
        printf("Can't open %s file!\n",argv[1]);
        exit(-1);
    }

    while((ch=fgetc())!=EOF)                          /* 从filename中读字符 */
        putchar(ch);                                  /* 向显示器中写字符 */
    return 0;
}
```

图 8-1 运行结果示意图

2.解答:

主要代码如下:

```
1     if((freopen(argv[1], "r", stdin))==NULL){/*重定向 stdin 指向 filename
2         printf("Can't open  %s  file!\n",argv[1]);
3         exit(-1);
4     }
5
6     while((ch=fgetc())!=EOF)                /* 从 filename 中读字符 */
7         putchar(ch);                        /* 向显示器中写字符 */
8     return  0;
9 }
```

运行结果:

```
C:\Users\Shooting stars\Desktop\Study\C\code\C_experiments\fqh\8\source>2.2 2.2.c
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char* argv[])
{
    int ch;
    if(argc!=2){
        printf("Arguments error!\n");
        exit(-1);
    }
    if((freopen(argv[1], "r", stdin))==NULL){           /*重定向stdin指向filename*/
        printf("Can't open %s file!\n",argv[1]);
        exit(-1);
    }

    while((ch=getchar())!=EOF)                          /* 从filename中读字符 */
        putchar(ch);                                    /* 向显示器中写字符 */
    return 0;
}
```

图 8-2 运行结果示意图

8.2.3 程序设计题

1.编写一个程序 `replace`，采用命令行方式，用给定的字符串替换指定文件中的目标字符串，并显示输出替换的个数。例如，命令行：

`replace filename.txt you they`

`you like your self`

解答：

代码如下：

```
1
2 int  replace(char  *p,char*temp, char  *t1, char  *t2)
3 {
4     int  flag  =  0;
5     char  *p0  =  p;
6     while(*temp++);
7     temp--;
8     while(*p++  ==  *t1++);
9     t1--;
10    if(*t1  ==  '\0')
11    {
12
13        while(*temp++  =  *t2++);
14        flag  =  1;
```

```

15     }
16     else
17     {
18         *temp++ = *p0;
19         *temp = '\0';
20     }
21     return flag;
22 }

```

运行结果：替换前



图 8-3 运行结果示意图

运行结果：

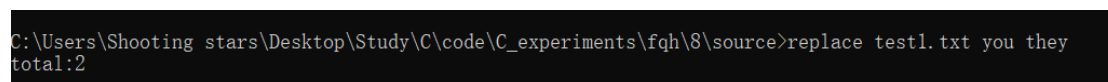


图 8-4 运行结果示意图

替换后：

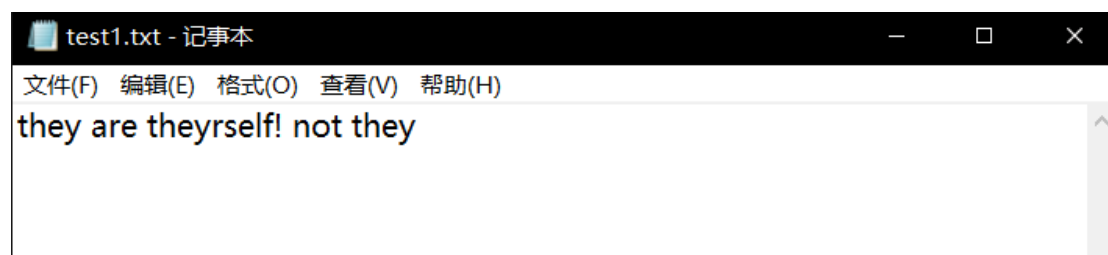


图 8-5 运行结果示意图

(2) 从键盘输入 10 个单精度浮点数，以二进制形式存入文件 float.dat 中。再从文件中读出这 10 个单精度浮点数显示在屏幕上。之后要求将 float.dat 中的单精度浮点数按字节读出来，观察写入文件的浮点数字节数据是不是和计算机内存中表示的浮点数字节数据一致。

解答:

代码如下:

```
1      fwrite(a, sizeof(float), N, fp);
2      freopen("float.dat", "rb", fp);
3      fread(b,sizeof(float), N, fp);
4      rewind(fp);
5      fread(c, sizeof(char), M, fp);
6      for(i = 0; i<N; i++)
7          printf("%f\n", b[i]);
8      printf("字节比较:");
9
10     }
11     fclose(fp);
12     return 0;
13 }
```

运行结果:

```

1.1
2.2
3.3
4.4
5.5
6.6
7.7
8.8
9.9
1.2
1.100000
2.200000
3.300000
4.400000
5.500000
6.600000
7.700000
8.800000
9.900000
1.200000
字节比较:第0个小数的字节比较:
第1个字节:
源文件中:cd      二进制文件中:cd
第2个字节:
源文件中:cc      二进制文件中:cc
第3个字节:
源文件中:8c      二进制文件中:8c
第4个字节:
源文件中:3f      二进制文件中:3f
第1个小数的字节比较:
第1个字节:
源文件中:cd      二进制文件中:cd
第2个字节:
源文件中:cc      二进制文件中:cc
第3个字节:
源文件中:c       二进制文件中:c
第4个字节:
源文件中:40      二进制文件中:40
第2个小数的字节比较:
第1个字节:
源文件中:33      二进制文件中:33
第2个字节:
源文件中:33      二进制文件中:33
第3个字节:
源文件中:53      二进制文件中:53

```

图 8-6 运行结果示意图

8.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在本次实验中，了解到了有关计算机内部工作原理以及存储的相关知识，学习到了大量有关文件操作的函数，在实际编程过程中常常遇到函数的用法，作用，参数不清楚的情况，这就需要我们理解记忆有关函数，并多加练习才能灵活应用，实现对文件的灵活操作。

参考文献

- [1] 曹计昌,卢萍,李开. C 语言程序设计,北京: 科学出版社,2013
- [2] 李开,卢萍,曹计昌. C 语言实验与课程设计, 北京: 科学出版社,2011