

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: 计算机(二) 2108 班

学 号: U202117281

姓 名: 张骁凯

指导教师: 方少红

报告日期: 2021-12-22

软件学院

目 录

1	表达式和标准输入与输出实验	3
1.1	实验目的	3
1.2	实验内容	3
1.3	实验小结	10
2	流程控制实验	11
2.1	实验目的	11
2.2	实验内容及要求	11
2.3	实验小结	25
3	函数与程序结构实验	26
3.1	实验目的	26
3.2	实验内容	26
3.3	实验小结	44
4	编译预处理实验	45
4.1	实验目的	45
4.2	实验内容	45
4.3	实验小结	56
5	数组实验	57
5.1	实验目的	57
5.2	实验内容及要求	57
5.3	实验小结	69
6	指针实验	70
6.1	实验目的	70
6.2	实验题目及要求	70
6.3	实验小结	97
7	结构与联合实验	98
7.1	实验目的	98
7.2	实验题目及要求	98
7.3	实验小结	115
8	文件操作实验	116
8.1	实验目的	116
8.2	实验题目及要求	116
8.3	实验小结	121

1 表达式和标准输入与输出实验

1.1 实验目的

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

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

(3) 掌握简单 C 程序（顺序结构程序）的编写方法。

1.2 实验内容

1 源程序改错

下面给出了一个简单 C 语言程序例程，用来完成以下工作：

(1) 输入华氏温度 `f`，将它转换成摄氏温度 `c` 后输出；

(2) 输入圆的半径值 `r`，计算并输出圆的面积 `s`；

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

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

```
1  #include<stdio.h>
2  #define PI 3.14159;
3  voidmain( void )
4  {
5    int f;
6    short p, k;
7    double c , r , s;
8    /* for task 1 */
9    printf("Input  Fahrenheit:");
10    scanf("%d", f);
11    c = 5/9*(f-32);
12    printf( " \n %d (F) = %.2f (C)\n\n", f, c );
```

```

13  /* for task 2 */
14  printf("input the radius r:");
15  scanf("%f", &r);
16  s = PI * r * r;
17  printf("\nThe acreage is %.2f\n\n",&s);
18  /* for task 3 */
19  printf("input hex int k, p :");
20  scanf("%x %x", &k, &p );
21  newint = (p&0xff00)|(k&0xff00)<<8;
22  printf("new int = %x\n\n",newint);
    }

```

解答：

(1) 错误修改：

1) 第 2 行的符号常量定义后不能有分号，应改为

```
#define PI 3.14159
```

2) 第 5 行的 f 类型应该为 double，应改为

```
double f;
```

3) 第 10 行 f 的类型为 double，应改为

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

4) 第 11 行 c 的类型为 double，应改为

```
c = 1.0 * 5/9 * (f-32);
```

5) 第 12 行 c 的类型为 double，应改为

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

6) 第 15 行 r 的类型为 double，应改为

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

7) 第 17 行存在 printf 语法错误，应改为

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

8) 第 21 行存在 newint 类型、算法错误，应改为

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

(2) 错误修改后运行结果：

```

C:\Users\dekr\Desktop\cpp\exp 1-1.exe
Input Fahrenheit: 80
80.00 (F) = 26.67 (C)
input the radius r:1
The acreage is 3.14
new int = 0x84a1
    
```

图 1-1 程序改错题的程序运行结果示意图

2 程序设计

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

解答：

1) 算法流程如图 1.1 所示。

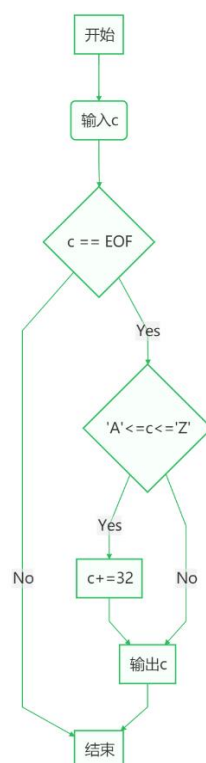


图 1-2 编程题 1 的程序流程图

2) 源程序清单

```
#include <stdio.h>

int main()
{
    char c ;
    while ( scanf("%c",&c) != EOF )
    {
        if ( c >= 'A' && c <= 'Z')
        {
            c += 32 ;
            printf("%c",tmp) ;
        }
        else printf("%c",c) ;
    }
    return 0;
}
```

3) 测试

(a) 测试数据:

A

(b) 对应测试数据的运行结果截图

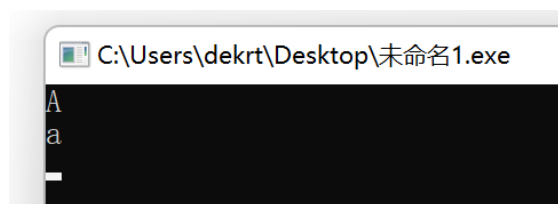


图 1-3 编程题 1 的程序运行结果示意图

(2) 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。要求: ①检查 m 和 n 的范围; ② x 的值以十六进制输入, m 和 n 以十进制输入; ③结果以十六进制输出。

解答:

1) 解题思路:

1. 输入 x, m, n, 为了方便分析测试结果, x 的输入采用 16 进制
2. 如果 $0 \leq m \leq 15, 1 \leq n \leq 16-m$, 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 $15-n-1$ 位, 即: $(x \gg m) \ll (15 - n - 1)$
 - 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>

#define re register

int main()
{
    unsigned short int x, m, n, ans;
    scanf("%hx%hd%hd",&x,&m,&n);
    // return 0;
    if (m < 0 || m > 15 || m + n < 0 || m + n > 16)
    {
        printf("error");
        return 0;
    }
    ans = (x >> m) << (15 - n - 1);
    printf("%x",ans);
    return 0;
}
```

3) 测试

- (a) 测试数据: 如表 1-1 所示。

表 1-1 编程题 3 的测试数据

测试用例	程序输入			理论结果
	X	m	N	
用例 1	0100 0110 1000 0000 (4680)	7	4	计算结果 1101 0000 0000 0000 即 D000
用例 2	1101 0101 1000 0011 (D583)	16	1	输入错误 (m 值超范围)

用例 3	1101 0101 1000 0011 (D583)	13	5	输入错误 (n 值超范围)
------	----------------------------	----	---	---------------

(b) 对应测试用例 1 的运行结果如图 1-2 所示。

```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
输入x (16进制)、m (0~15) 和n (1~16-m):
4680 7 4
ans=d000

-----
Process exited with return value 9
Press any key to continue . . .

```

图 1-4 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 1-3 所示。

```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
输入x (16进制)、m (0~15) 和n (1~16-m):
d583 16 1
输入错误!

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

```

图 1-5 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 1-4 所示。

```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
输入x (16进制)、m (0~15) 和n (1~16-m):
d583 13 5
输入错误!

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

```

图 1-6 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 (即点分十进制)，但这些地址在机器中是用一个无符号长整型数表示的。例如 3232235876，其机内二进制表示就是 11000000 10101000 00000001 01100100，按照 8 位一组用点分开，该 IP 地址就写成 192.168.1.100。

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

解答：

1) 解题思路:

1.输入 x 为了方便分析测试结果，x 的输入采用 unsigned long int 型

2.如果 x != EOF 转 2.1，否则转 3.

2.1 首先定义 mask = 0x000000ff ;

2.2 计算结果并进行输出，(x>>24)&t.(x>>16)&t.(x>>8)&t.(x&t),

然后转 3:

3. 结束

2) 程序清单

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

```
#define re register
```

```
unsigned long int x;
```

```
int main()
```

```
{
    while (scanf("%ld",&x) != EOF )
    {
        unsigned long int t = 0x000000ff ;
        printf("%d.%d.%d.%d\n", (x>>24)&t, (x>>16)&t, (x>>8)&t, (x&t)) ;
    }
    return 0;
}
```

3) 测试

(a) 测试数据:

3232235876

(b) 对应测试数据的运行结果截图

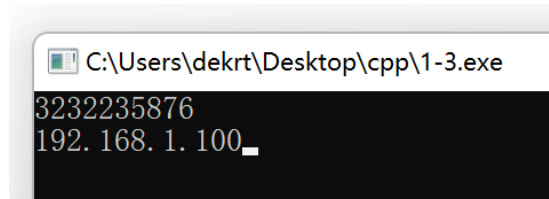


图 1-7 编程题 4 的运行结果

1.3 实验小结

实验中对于部分算法、语法存在许多疑惑，但经过查阅资料、自己摸索、借助 ide 的 debug、调试等功能，成功解决了问题。经过这次实验，我体会到了自学探索的重要性，对有关知识的理解更深了。

实验 2 流程控制实验

2.1 实验目的

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

(2) 练习循环结构 for、while、do-while 语句的使用。

(3) 练习转移语句和标号语句的使用。

(4) 使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

2.2 实验内容及要求

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      {
8          for(i=2,k=x>>1;i<=k;i++)
9              if (!x%i) {
10                 flag = 1;
11                 break;
12             }
13         if(flag=1) printf("%d是合数", x);
14         else printf("%d不是合数", x);
15     }
16     return 0;
17 }
```

解答：

(1) 错误修改：

1) 第 9 行的运算优先级出错，正确形式为：

```
if( !(x % i) )
```

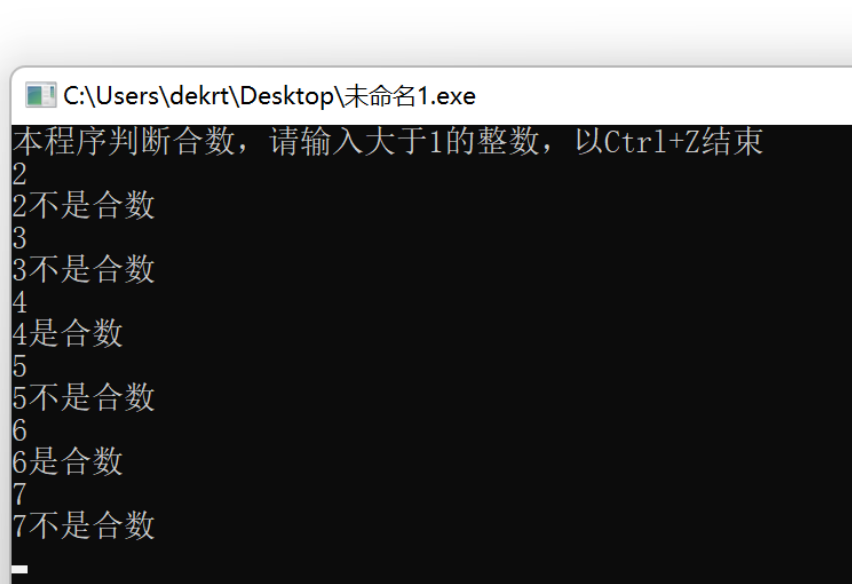
2) 第 13 行的 if(flag=1) 语法出错，正确形式为：

```
if ( flag == 1 )
```

3) 第 15 行的算法出错，正确形式为：

```
flag = 0 ;}
```

(2) 错误修改后运行结果：



```
C:\Users\dekrt\Desktop\未命名1.exe
本程序判断合数，请输入大于1的整数，以Ctrl+Z结束
2
2不是合数
3
3不是合数
4
4是合数
5
5不是合数
6
6是合数
7
7不是合数
```

图 2-1 程序改错的测试用例运行结果

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。

解答：

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

```
int main( )
{
    int i, x, k, flag = 0;

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

    while (scanf("%d", &x) != EOF)
    {
        for(i=2,k=x>>1; i<=k && ( flag == 0 );i++)

            if (!(x%i)) flag = 1;

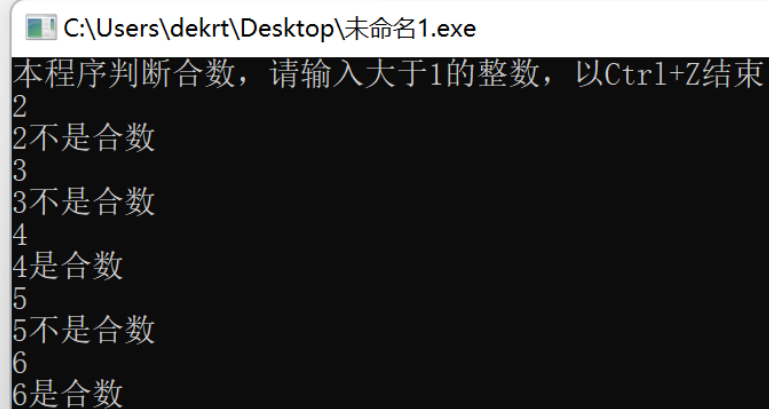
        if(flag == 1) printf("%d 是合数", x);

        else printf("%d 不是合数", x);

        flag = 0 ;
    }

    return 0;
}
```

(2) 修改后运行结果：



```
C:\Users\dekr\Desktop\未命名1.exe
本程序判断合数，请输入大于1的整数，以Ctrl+Z结束
2
2不是合数
3
3不是合数
4
4是合数
5
5不是合数
6
6是合数
```

图 2-2 程序修改替换的测试用例运行结果

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

解答：

```
#include <stdio.h>

int main( )
{
    int i, x, k, flag = 0;
    printf("本程序判断合数，请输入大于 1 的整数，以 Ctrl+Z 结束\n");
    do {
        if(x)
        {
            for(i=2,k=x>>1; i<=k && ( flag == 0 );i++)
            if (!(x%i)) flag = 1;
            if(flag == 1) printf("%d 是合数\n", x);
            else printf("%d 不是合数\n", x);
            flag = 0 ;
        }
    } while (scanf("%d", &x) !=EOF) ;
    return 0;
}
```

(2) 修改后运行结果：

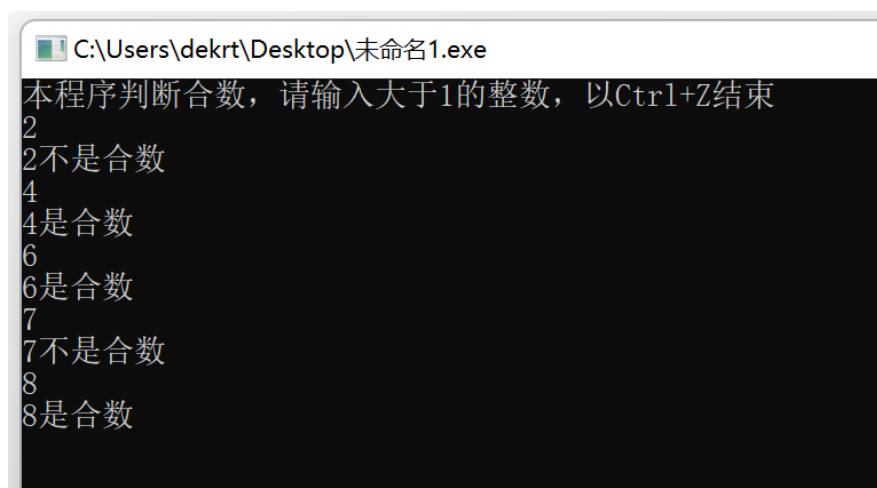


图 2-3 程序修改替换的测试用例运行结果

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

解答：

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

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

```
#define re register
```

```
int judge ( int x )
```

```
{
```

```
    int tmp = sqrt(x);
```

```
    for( re int i = 2 ; i <= tmp ; i++ )
```

```
        if( x % i == 0 ) return 1 ;
```

```
    return 0 ;
```

```
}
```



```
int main( )
{
    for( re int i = 100 ; i <= 999 ; i++ )
    {
        if ( judge (i) && judge (i / 10) && judge (i/100) )
            printf ("%d ", i );
    }
    return 0;
}
```

(2) 修改后运行结果:

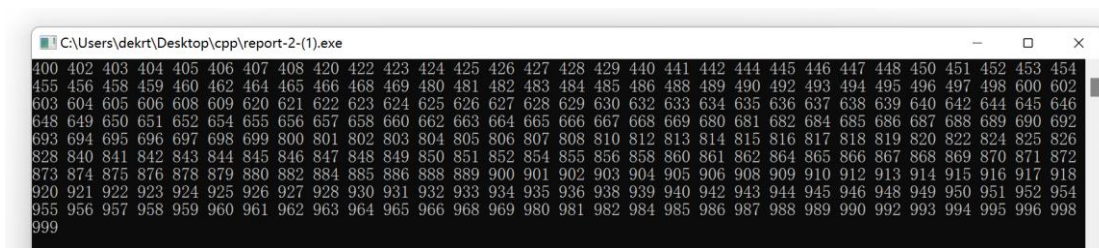


图 2-4 程序修改替换的测试用例运行结果

2.2.3 程序设计

(1) 假设工资税金按以下方法计算: $x < 1000$ 元, 不收取税金; $1000 \leq x < 2000$, 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; $5000 \leq x$, 收取 25% 的税金。(注意税金的计算按照阶梯计税法, 比如, 工资为 4500, 那么税金 = $1000 \times 5\% + 1000 \times 10\% + 1000 \times 15\% + 501 \times 20\%$)。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:

/*使用 if 版本*/

#include <stdio.h>

#define re register

double x ;

int main()

{

scanf("%lf",&x) ;

int tmp = x / 1000 ;

if(tmp == 0) printf("收税金额为%d",0) ;

else if(tmp == 1) printf("收税金额为%.2lf", (x-1000) * 0.05) ;

else if(tmp == 2) printf("收税金额为%.2lf", 50 + (x-2000) * 0.1) ;

else if(tmp == 3) printf("收税金额为%.2lf", 150 + (x-3000) * 0.15) ;

else if(tmp == 4) printf("收税金额为%.2lf", 300 + (x-4000) * 0.2) ;

else if(tmp >= 5) printf("收税金额为%.2lf", 500 + (x-2000) * 0.25) ;

return 0 ;

}

/*使用 switch 版本*/

#include <stdio.h>

#define re register

double x ;

```

int main()

{

    scanf("%lf",&x) ;

    int tmp = x / 1000 ;

    switch ( tmp )

    {

        case 0 : printf("收税金额为%d",0) ; break ;

        case 1 : printf("收税金额为%.2lf", (x-1000) * 0.05 ) ; break ;

        case 2 : printf("收税金额为%.2lf", 50 + (x-2000) * 0.1 ) ; break ;

        case 3 : printf("收税金额为%.2lf", 150 + (x-3000) * 0.15 ) ; break ;

        case 4 : printf("收税金额为%.2lf", 300 + (x-4000) * 0.2 ) ; break ;

        default : printf("收税金额为%.2lf", 500 + (x-2000) * 0.25 ) ; break ;

    }

    return 0 ;

}

```

测试数据：

程序运行截图：



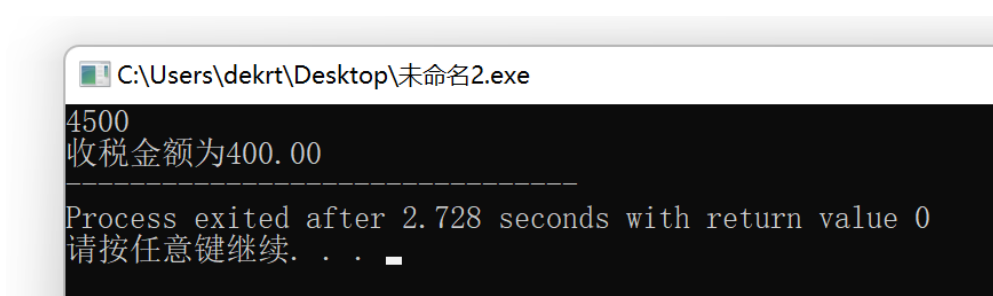
图 2-5 编程题 1 的测试用例 1 运行结果



```

C:\Users\dekrt\Desktop\未命名1.exe
2500
收税金额为100.00
-----
Process exited after 1.086 seconds with return value 0
请按任意键继续. . .
    
```

图 2-6 编程题 1 的测试用例 2 运行结果



```

C:\Users\dekrt\Desktop\未命名2.exe
4500
收税金额为400.00
-----
Process exited after 2.728 seconds with return value 0
请按任意键继续. . .
    
```

图 2-7 编程题 1 的测试用例 3 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。

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

1) 解题思路：

- 1.输入字符 c，如果输入未完成则继续输入
- 2.如果 c 为空格
 - 2.1 如果 flag 为真，则继续读取字符 c
 - 2.2 如果 flag 为假，则 flag = 1 ， 并且输出字符 c
3. 如果 c 不为空格，则 flag=0,并且输出字符 c
4. 结束

2) 程序清单

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

```
char c ;
int flag = 0 ;
```

```
int main()
{
    while ( scanf("%c",&c) != EOF )
    {
        if ( c == ' ' )
        {
            if ( flag ) continue ;
            else
            {
                flag = 1 ;
                putchar(c) ;
            }
        }
        else
        {
            flag = 0 ;
            putchar(c) ;
        }
    }
    return 0 ;
}
```

3) 测试

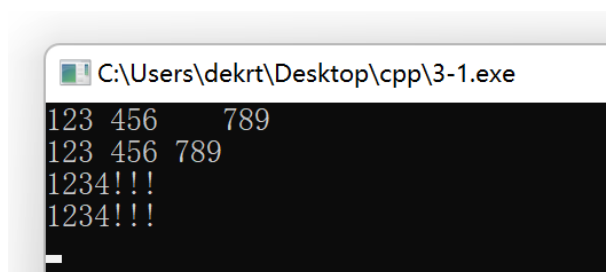


图 2-8 编程题 2 的测试用例运行结果

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) 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，输入最后一行的编号 N ($0 \leq N \leq 6$)，要求输出金字塔效果的杨辉三角形。

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

1) 解题思路：

1. 输入整数 n
2. 使用递推求出组合数 $c[i][j]$
3. 利用双层循环结合题意对杨辉三角进行输出
4. 结束

2) 程序清单

```
#include <stdio.h>

#define re register
#define maxn 10

int n ;
int c[maxn][maxn] ;

int main()
{
    scanf("%d",&n) ;
    c[0][0] = 1 ;
    for( re int i = 1 ; i <= n ; i++ )
    {
        c[i][0] = c[i][i] = 1 ;
        for( re int j = 1 ; j <= ( i >> 1 ) ; j++ )
        {
            c[i][j] = c[i][i-j] = c[i-1][j] + c[i-1][j-1] ;
        }
    }
    for ( re int i = 0 ; i <= n ; i++ )
    {
        for( re int j = 1 ; j <= n + 2*(n-i) ; j++ )
        {
            printf(" ") ;
        }
        for ( re int k = 0 ; k <= i ; k++ )
        {
            printf("%-4d",c[i][k]) ;
        }
        printf("\n") ;
    }
    return 0;
}

3) 测试
```

```

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
    
```

图 2-9 编程题 3 的测试用例运行结果

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。

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

1) 解题思路：

1. 输入字符 c，如果输入未完成则继续输入
2. 如果 c 为空格
 - 2.1 如果 flag 为真，则继续读取字符 c
 - 2.2 如果 flag 为假，则 flag = 1，并且输出字符 c
3. 如果 c 不为空格，则 flag=0,并且输出字符 c
4. 结束

2) 程序清单

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

```
#define re register
```

```
int main()
{
```



```
for (re int i = 100 ; i <= 999 ; i++ )  
{  
    if ( ( i*i % 1000 ) == i ) printf("%d\n",i);  
}  
return 0;  
}3) 测试
```

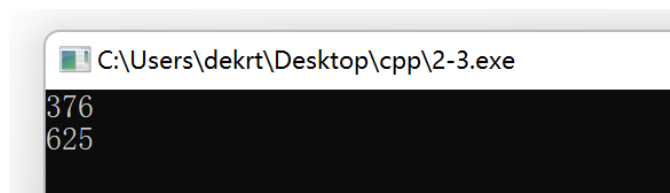


图 2-10 编程题 4 的测试用例运行结果

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。

2.3 实验小结

(1) 通过这次试验，我掌握了复合语句、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，对它们有了更深的理解。

(2) 通过这次试验，我练习了循环结构 for、while、do-while 语句的使用，熟练使用了他们。

(3) 通过这次试验，我练习了转移语句和标号语句的使用。

(4) 通过这次试验，我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，提高了 debug 效率。

实验 3 函数与程序结构实验

3.1 实验目的

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

3.2 实验内容

3.2.1. 程序改错题

下面是计算 $s=1!+2!+3!+\cdots+n!$ 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 $s=1!+2!+3!+\cdots+n!$ */

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

```
2  int main(void)
```

```
3  {
```

```
4      int k;
```

```

5      for(k=1;k<=20;k++)
6          printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
7      return 0;
8  }

9  long sum_fac(int n)
10 {
11     long s=0;
12     int i,fac;
13     for(i=1;i<=n;i++)
14         fac*=i;
15     s+=fac;
16     return s;
17 }

```

解答：

(1) 错误修改：

- 4) 第 3 行未对函数 sum_fac 进行声明，同时函数数据类型出错，正确形式为：

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

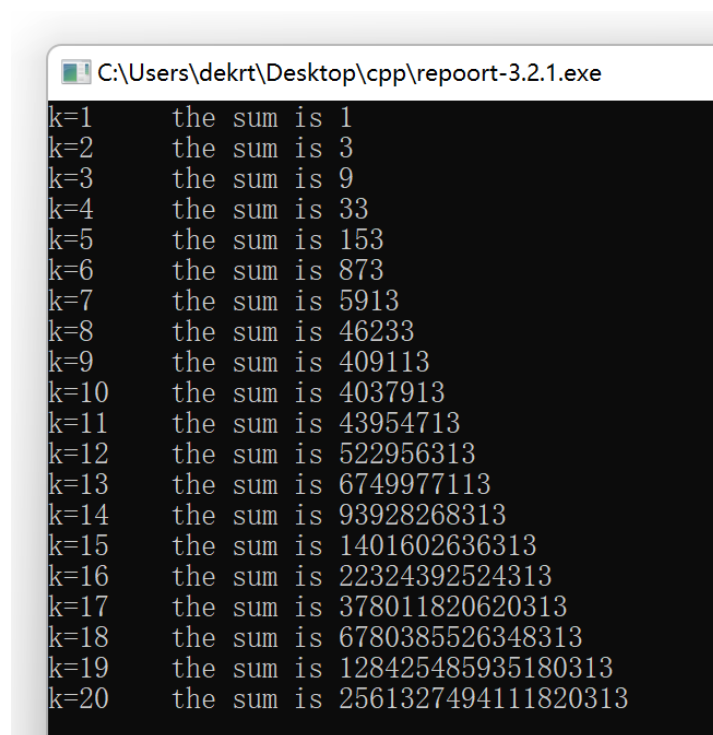
- 5) 第 11 行 s 的数据类型出错，正确形式为：

```
static long long s=0;
```

- 6) 第 12 行 fac 的数据类型、初始值出错，正确形式为：

```
long long fac = 1 ;
```

(2) 错误修改后运行结果：



```

C:\Users\dekr\Desktop\cpp\repoort-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
    
```

图 3-1 程序改错题的程序运行截图

3.3.2. 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 $\sum_{i=1}^n i!$ 。

个递归函数，用递归的方式计算 $\sum_{i=1}^n i!$ 。

解答：

(1) 替换后的程序如下：

```

#include <stdio.h>

#define re register

long long sum_fac(int n);

int main(void)
{
    int k;
    
```

```

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

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

return 0;

}

```

```

long long sum_fac(int n)
{
    if ( n == 1 ) return 1;

    long long fac = 1 ;

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

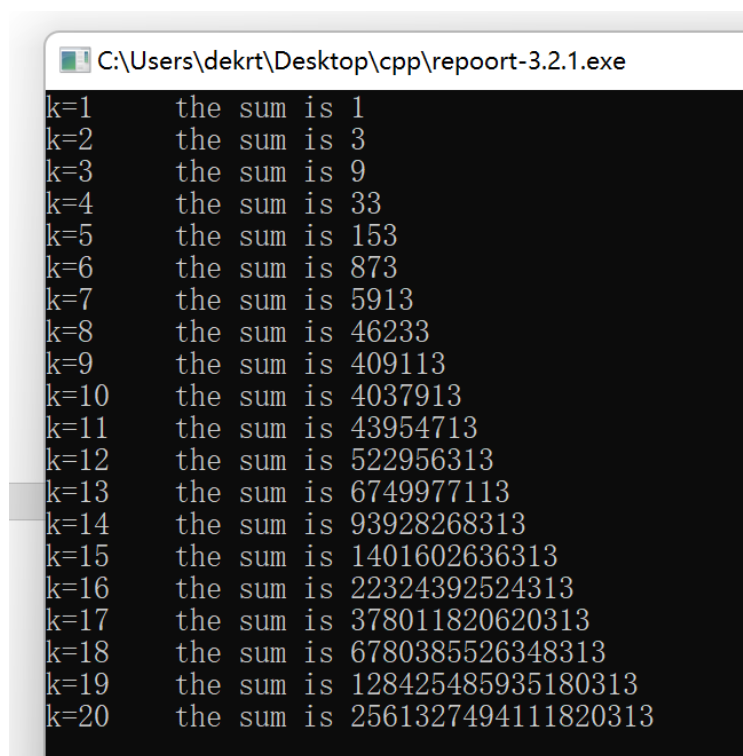
        fac *= i ;

    return fac + sum_fac( n - 1 ) ;

}

```

(2) 程序运行结果示意图:



```

C:\Users\dekr\ Desktop\cpp\repoort-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
    
```

图 3-2 程序修改替换题（1）的程序运行截图

(2) 下面是计算 $s = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$ 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 */

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

```
double mulx(double x,int n);
```

```
long fac(int n);
```

```
double sum(double x,int n)
```

```
{
```

```
    int i;
```

```
    double z=1.0;
```

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

```

    {

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

    }

    return z;

}

double mulx(double x,int n)
{

    int i;

    double z=1.0;

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

    {

        z=z*x;

    }

    return z;

}

long fac(int n)
{

    int i;

    long h=1;

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

    {

        h=h*i;
    }

```

```

    }

    return h;

}

int main()

{

    double x;

    int n;

    printf("Input x and n:");

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

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

    return 0;

}

```

解答：

(1) 替换后的程序如下：

```

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n)

{

    int i;

    double z=1.0;

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

```



```

    {

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

    }

    return z;

}

```

double mulx(double x,int n)

```

{

    int i;

    static double z=1.0;

    z *= x ;

    return z;

}

```

long fac(int n)

```

{

    if ( n == 1 ) return 1 ;

    static long f = 1 ;

    f *= n ;

    return f ;

}

```

int main()

```

{

    double x;

```

```

int n;

printf("Input x and n:");

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

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

return 0;

}

```

(2) 替换后的程序运行结果如下：

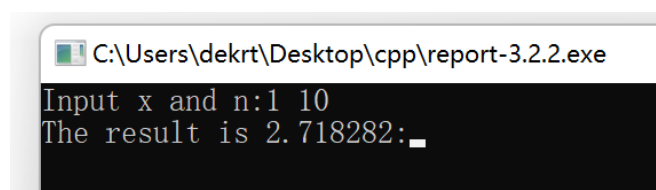


图 3-3 程序修改替换题（2）的程序运行截图

3.3.3. 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：

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

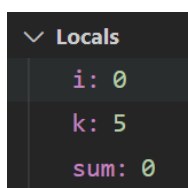


图 3-4 跟踪调试题（1）的程序运行截图

(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？

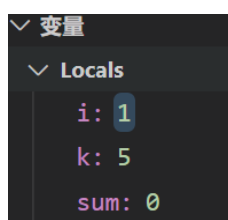


图 3-5 跟踪调试题（2）的程序运行截图

- (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？

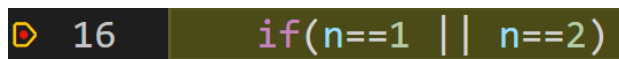


图 3-6 跟踪调试题（3）的程序运行截图

- (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？

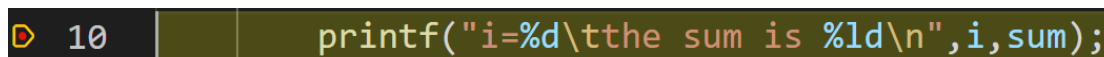


图 3-7 跟踪调试题（4）的程序运行截图

- (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？

解答：

这两个变量是开在 main()函数的局部变量, fibonacci 函数无法调用。

/*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,fibonacci(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);
}

```

3.3.4. 程序设计

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

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

例如：n=6，输出“6=3+3”。

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

1) 解题思路：

1.输入整数 n

2.如果 n 是大于或等于 4 的偶数，调用 solve(n)函数进行验证。

2.1 首先从 i=2 开始进行循环，调用 judge(int a)函数进行判断。

2.2 judge 函数：检测传入的参数是否为质数，如果是质数返回 1，否则返回 0.

2.3 如果 i, n-i 都是质数，则进行输出并终止循环。

3. 结束

2) 程序清单

```
#include <stdio.h>
#include <math.h>
#define re register
int n;
int judge (int a)
{
    for(re int i = 2 ; i <= sqrt(a) ; i++ )
    {
        if ( a % i == 0 ) return 0 ;
    }
    return 1 ;
}

int solve (int a)
{
    for( re int i = 2 ; i <= a ; i++ )
    {
        if( judge(i) && judge(a-i) )
        {
            printf("%d=%d+%d\n",a,i,a-i);
            break ;
        }
    }
    return 0;
}
```

```
int main()
{
    while (scanf("%d",&n) != EOF )
        if(n >= 4 && n % 2 == 0 ) solve(n);
    return 0;
}
```

3) 测试

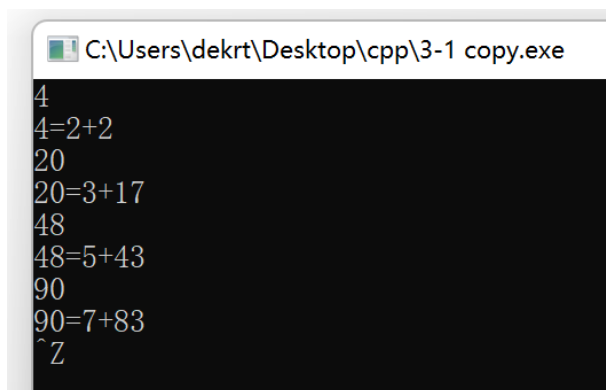


图 3-8 程序设计题（1）的程序运行截图

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

编程寻找 10000 以内的所有完全数。

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

1) 解题思路：

1. 从 $i=2$ 到 10000 进行遍历，调用 `judge(int n), print(int x)` 进行验证

2. judge 函数：

2.1 先运用循环求出传入整数的所有因子，并调用数组进行储存。

2.2 再将所有因子进行求和，判断其是否与传入的参数相等，相等返回 1，不相等返回 0。

2.3 如果 i ， $n-i$ 都是质数，则进行输出并终止循环。

3. print 函数:

3.1 如果传入参数是完全数, 调用 judge 函数中的数组对所有的因子进行输出。

3.2 对数组, cnt 进行清零。

2) 程序清单

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

#define re register
#define maxn 10005

int a[maxn];
int cnt = 0;

int judge ( int n )
{
    int sum = 0 ; cnt = 0 ;
    for ( re int i = 1 ; i <= n ; i++ )
    {
        if( n % i == 0 )
            a[++cnt] = i ;
    }
    for( re int i = 1 ; i < cnt ; i++ )
        sum += a[i] ;
    if( sum == n )
    {
        sum = 0 ;
        return 1;
    }
    sum = 0 ;
    return 0 ;
}

void print( int x )
{
    printf("%d=",x) ;
    for( re int i = 1 ; i < cnt - 1 ; i++ )
        printf("%d+", a[i] ) ;
```

```

printf("%d\n",a[cnt-1]);
cnt = 0;
memset ( a , 0 , sizeof(a) );
}

int main()
{
    for ( re int i = 2 ; i <= 10000 ; i++ )
        if ( judge (i) ) print(i);
    return 0;
}

```

3) 测试

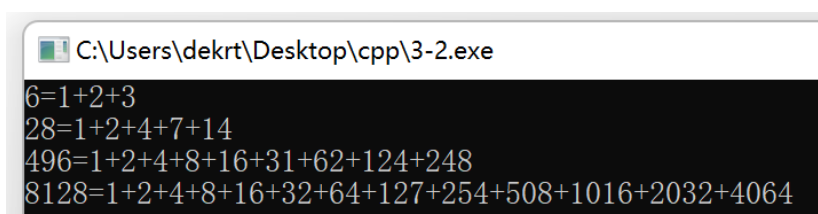


图 3-9 程序设计题（2）的程序运行截图

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

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

1) 解题思路：

1.输入 k ,调用 solve 函数进行解答。

2.solve 函数：

2.1 先运用循环从 $10^{(k-1)}$ 到 10^k-1 进行遍历，寻找符合条件的数

2.2 再将各位的数字按题目意思进行验证，如果符合则存入数组

2.3 进行输出

3.结束

2) 程序清单

```
#include <stdio.h>

#include <string.h>


#define re register

#define maxn 1005


int k , cnt ;

int a[maxn] ;


int qpow ( int x, int y )
{
    int ans = 1 ;

    while (y)
    {
        if( y & 1 ) ans *= x ;

        x = x * x ; y >>= 1 ;
    }

    return ans ;
}
```

```

int solve(int k)

{

    for( re int i = qpow(10,k-1) ; i <= qpow(10,k)-1 ; i ++ )

        {

            int tmp = i , sum = 0;

            while (tmp)

                {

                    int x = tmp%10 ;

                    sum += qpow(x,k);

                    tmp /= 10;

                }

            if( sum == i ) a[++cnt] = i ;

        }

    if ( k == 3 ) printf("3 位的水仙花数共有%d 个",cnt) ;

    if ( k == 4 ) printf("4 位的四叶玫瑰数共有%d 个",cnt) ;

    if ( k == 5 ) printf("5 位的五角星数共有%d 个",cnt) ;

    if ( k == 6 ) printf("6 位的六合数共有%d 个",cnt) ;

    if ( k == 7 ) printf("7 位的北斗星数共有%d 个",cnt) ;

    if ( k == 8 ) printf("8 位的八仙数共有%d 个",cnt) ;

    for ( re int i = 1 ; i < cnt ; i++)

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

    printf("%d\n",a[cnt]) ;

```

```

    cnt = 0 ; memset ( a , 0 , sizeof(a) ) ;

}

```

```

int main()

{

    while (1)

    {

        scanf("%d",&k);

        if ( k == 0 ) break ;

        solve(k) ;

    }

    return 0;

}

```

3) 测试

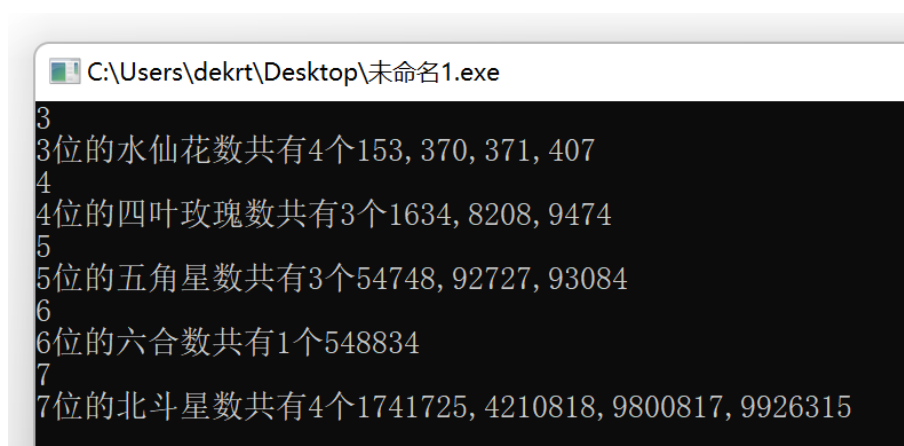


图 3-10 程序设计题（3）的程序运行截图

3.3 实验小结

- (1) 通过这次实验，我熟悉和掌握了函数的定义、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) 通过这次实验，我熟悉和掌握了不同存储类型变量的使用。
- (3) 通过这次实验，我练习使用了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。

实验 4 编译预处理实验

4.1 实验目的

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

4.2 实验内容

4.2.1. 程序改错

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

/*实验 4-1 改错与跟踪调试题程序: 计算平方差、将换两数*/

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

解答：

(1) 错误修改：

1) 第 2 行的宏定义有误，正确形式为：

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

2) 第 3 行的宏定义有误，正确形式为：

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

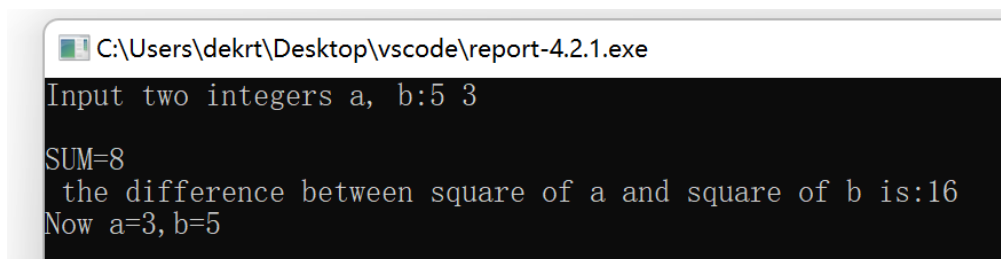
3) 第 4 行的宏定义有误，正确形式为：

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

3) 第 7 行的宏定义不全，正确形式为：

```
int a,b,t;
```

(2) 错误修改后运行结果：



```
C:\Users\dekr\Desktop\vscode\report-4.2.1.exe
Input two integers a, b: 5 3
SUM=8
the difference between square of a and square of b is:16
Now a=3, b=5
```

图 4-1 程序修改题的运行结果图

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

```

```

25 float sum(float x, float y)
26 {
27     return x+y;
28 }

```

解答：

(1) 错误修改：

1) 第 2 行的声明有误，正确形式为：

```

int max(int x, int y, int z);
float sum(float x, float y);

```

(2) 程序替换：

```

#define max(a,b,c) ( a > b ) ? ( a > c ? a : c ) : ( c > b ? c : b )

```

(3) 错误修改后运行结果：

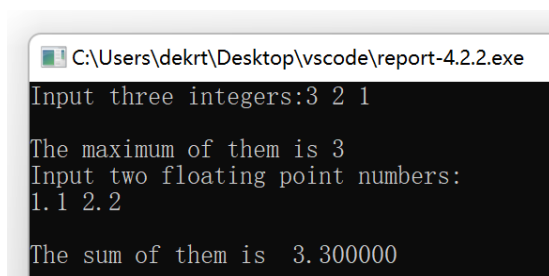


图 4-2 程序修改替换题的运行结果图

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 integerI_fraction 时，watch 窗口中 x 为何值？在返回 main 时，watch 窗口中 i 为何值？

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

/*实验 4-3 跟踪调试题程序利用 R 计算圆的面积 s*/

```
#define R

int main(void)
{
    float r, s;

    int s_integer=0;

    printf("Input a number: ");

    scanf("%f",&r);

    #ifdef R

        s=3.14159*r*r;

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

        s_integer=integer_fraction(s);

        assert((s-s_integer)<0.5);

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

    #endif

    return 0;
}

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

    return i;
}
```

(1) 错误修改:

1) 第 1 行的编译预处理有误, 正确形式为:

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

```
#include <assert.h>
```

2) 第 5 行未声明函数, 正确形式为:

```
int integer_fraction(float x);
```

(2) 单步执行:

1) 进入函数 integerl_fraction 时, watch 窗口中 x 为何值?

2) 在返回 main 时, watch 窗口中 i 为何值?

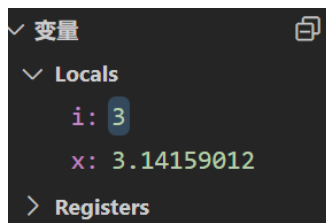


图 4-3 跟踪调试题的单步执行图

(3) 错误修改:

1) 第 16 行的 assert 有误, 正确形式为:

```
assert((s-s_integer)<1.0);
```

(4) 修改后的运行结果:

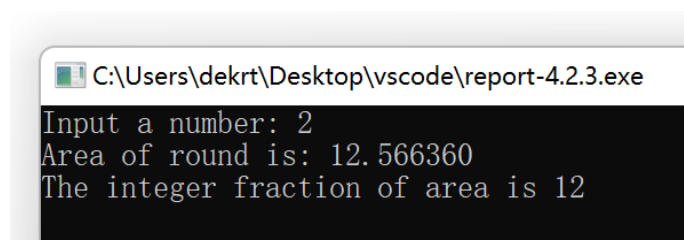


图 4-4 跟踪调试题的运行结果图

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。

1) 解题思路:

1.对 area, s 进行宏定义

2.读入 a,b,c

3.输出 area

4. 结束

2) 程序清单

```
#include <stdio.h>

#include <math.h>

#define re register

#define s ( ( a + b + c ) / 2 )

#define area ( sqrt( s * ( s - a ) * ( s - b ) * ( s - c ) ) )

double a , b , c ;

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

    printf("%lf",area) ;

    return 0 ;
}
```

3) 程序运行结果截图

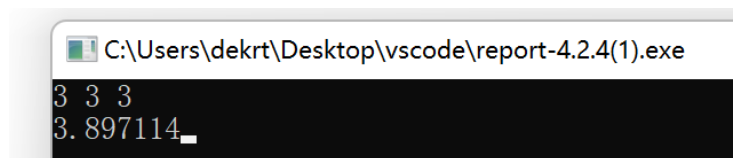


图 4-5 程序设计题（1）的运行结果图

(2) 用条件编译方法来编写程序。输入一行英文字符序列，可以任选两种方式

之一输出：一为原文输出；二为变换字母的大小写后输出。例如小写'a'变成大写'A'，大写'D'变成小写'd'，其他字符不变。用#define 命令控制是否变换字母的大小写。例如，#define CHANGE 1 则输出变换后的文字，若#define CHANGE 0 则原文输出。

1) 解题思路：

- 1.对输入的字符串用 gets()函数进行读入
- 2.使用#ifdef 命令进行判断
- 3.进入相应的程序模块进行有关处理、输出
- 4.结束

2) 程序清单

```
#include <stdio.h>

#include <math.h>

#include <string.h>


#define re register

#define maxn 10005

#define base ( 'a' - 'A' )


// #define CHANGE0

// #define CHANGE1


char s[maxn];
```

```

int main()

{

    gets(s) ;


    #ifdef CHANGE0

    {

        printf("%s",s) ;

    }

    #endif


    #ifdef CHANGE1

    {

        int cnt = strlen(s) ;

        for ( re int i = 0 ; i < cnt ; i++ )

        {

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

                printf("%c" ,s[i] + base ) ;

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

                printf("%c" ,s[i] - base ) ;

            else printf("%c",s[i] ) ;

        }

    }

}

```

```
#endif

return 0 ;

}
```

3) 程序运行结果截图

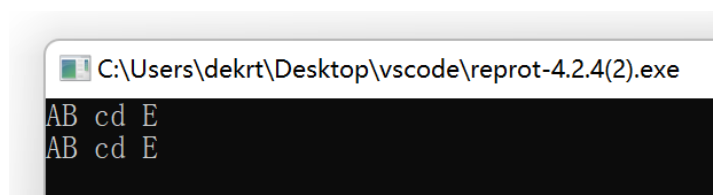


图 4-6 程序设计题（2）的运行结果图（1）

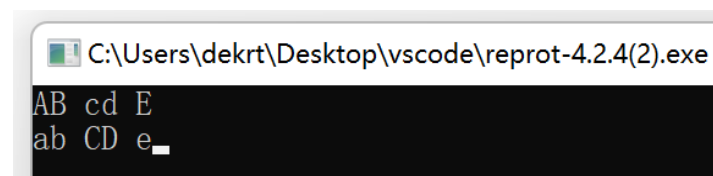


图 4-7 程序设计题（2）的运行结果图（2）

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

```
/*源文件 file1.c 的内容*/
#include "file.h"

int x,y;          /* 外部变量的定义性说明 */
char ch;          /* 外部变量的定义性说明 */
int main(void)
{
    x=10;
    y=20;
    ch=getchar();
    printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);
}
```

```
func1();
return 0;
}
```

/*源文件 file2.c 的内容为: */

```
#include "file.h"
void func1(void)
{
    x++;
    y++;
    ch++;
    printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);
}
```

1) 解题思路:

- 1.对 file.h 进行编写
- 2.对 file1.c 进行修改, 使其能够正常运行。

2) 程序清单

```
//file.h
#include <stdio.h>
int x,y;
char ch;
void func1(void)
{
    x++;
    y++;
    ch++;
    printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);
}
```

```
//file1.c
#include "file.h"
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;
}
```

3) 程序运行结果截图

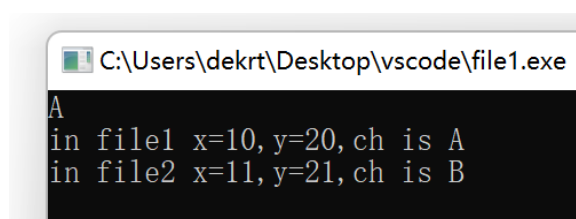


图 4-8 程序设计题（3）的运行结果图

4.3 实验小结

- (1) 通过这次实验，我掌握了文件包含、宏定义、条件编译和 assert 宏的使用；
- (2) 通过这次实验，我练习使用了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) 通过这次实验，我熟悉了多文件编译技术

实验 5 数组实验

5.1 实验目的

- (1) 掌握数组的说明、初始化和使用。
- (2) 掌握一维数组作为函数参数时实参和形参的用法。
- (3) 掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。
- (4) 掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

5.2 实验内容及要求

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

ProgrmingLngeuge

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。

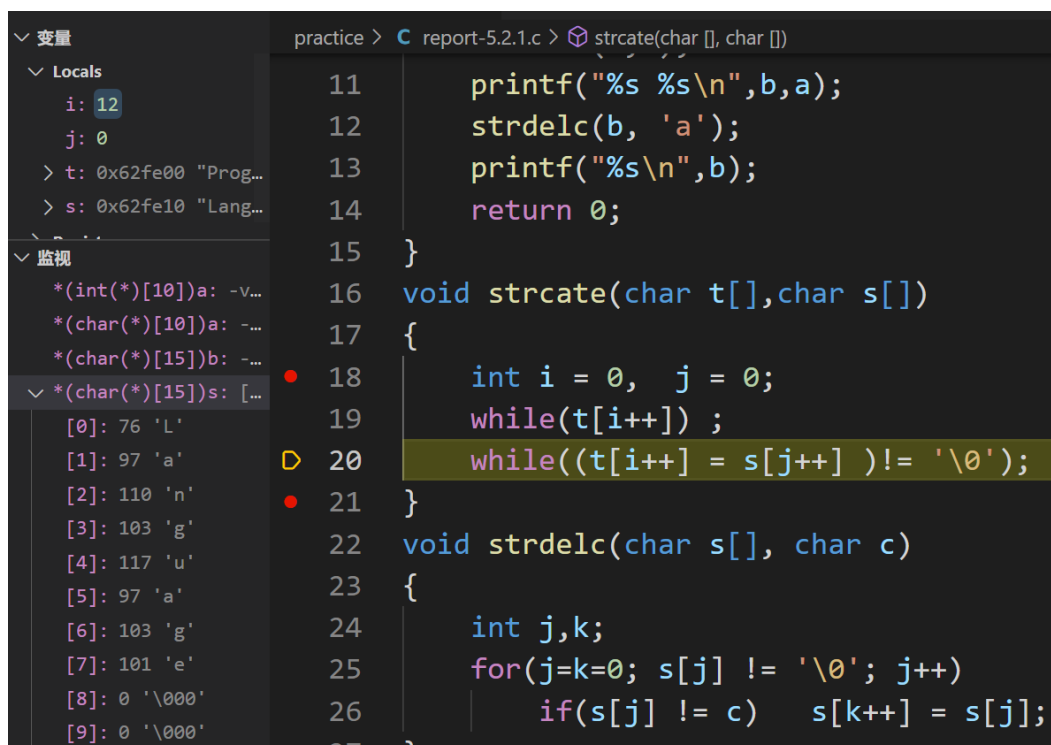


图 5-1 源程序改错与跟踪调试题的示意图 1

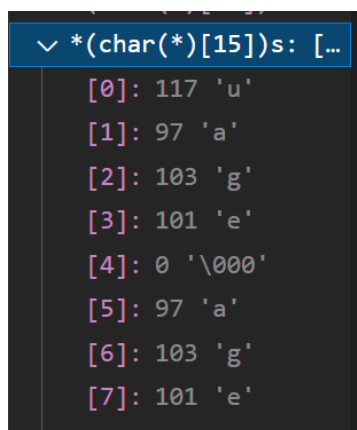


图 5-2 源程序改错与跟踪调试题的示意图 2

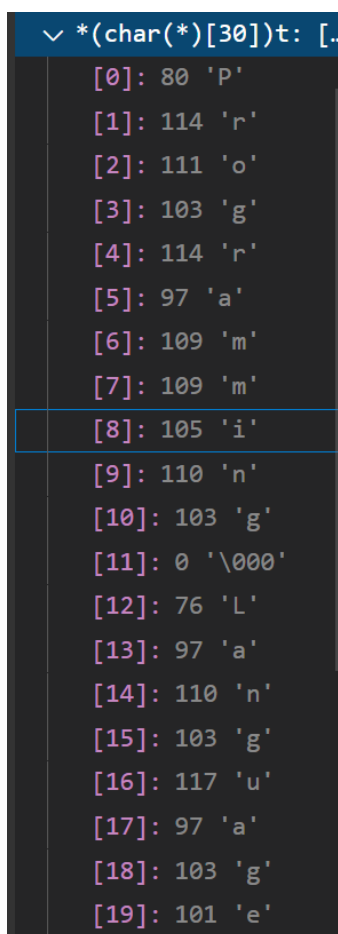


图 5-3 源程序改错与跟踪调试题的示意图 3

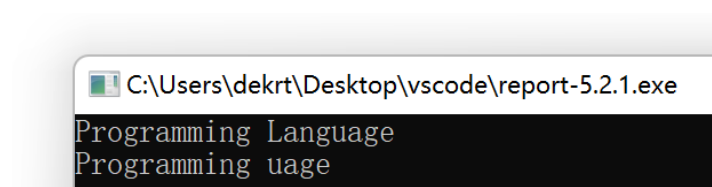


图 5-4 源程序改错与跟踪调试题的示意图 4

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。

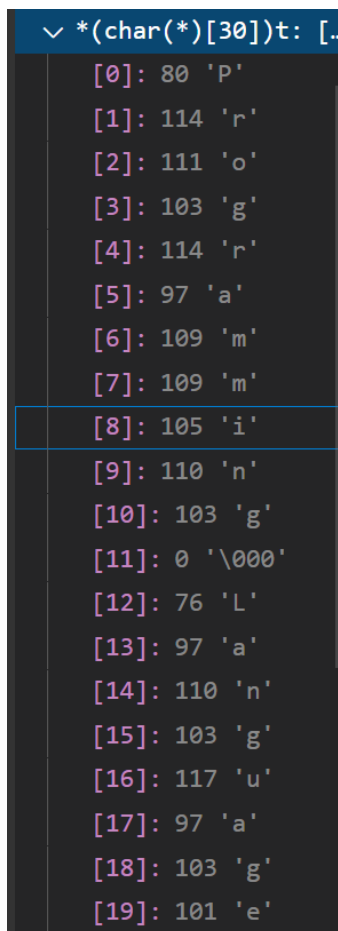


图 5-5 源程序改错与跟踪调试题的示意图 5

1) 修改后的源程序:

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

```
void strcat(char [],char []);
```

```
void strdelc(char [],char );
```

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

```
int main(void)
```

```
{
```

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

```
    strcat(b,a);
```

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

```
    strdelc(b , 'a');
```

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

```
    return 0;
```

```
}
```

```
void strcat(char t[],char s[])
```

```
{
```

```
    int i = 0, j = 0;
```

```

        while( t[i++] != '\0' );
        i -- ;
        while((t[i++] = s[j++]) != '\0') ;
    }
void strdelc(char s[], char c)
{
    int j,k;
    for( j = 0 , k = 0 ; s[j] != '\0' ; j++)
        if(s[j] != c)    s[k++] = s[j] ;
    while ( k <= j ) s[k++] = 0 ;
}
/*实验 5 程序改错与跟踪调试题程序*/
#include<stdio.h>
void strcate(char [],char []);
void strdelc(char [],char );
int main(void)
{
    char a[]="Language", b[]="Programming";
    printf("%s %s\n", b,a);
    strcate(b,a); printf("%s %s\n",b,a);
    strdelc(b, 'a'); printf("%s\n",b);
    return 0;
}
void strcate(char t[],char s[])
{
    int i = 0, j = 0;
    while(t[i++]) ;
    while((t[i++] = s[j++]) != '\0');
}
void strdelc(char s[], char c)
{
    int j,k;
    for(j=k=0; s[j] != '\0'; j++)
        if(s[j] != c)    s[k++] = s[j];
}

```

5.2.2 源程序完善和修改替换

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

序。

```
#include<stdio.h>
#define M 10
#define N 3
int main(void)
{
    int a[M], b[M];    /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int i, j, k;
    for(i = 0; i < M; i++)          /* 对圈中人按顺序编号 1—M */
        a[i] = i + 1;
    for(i = M, j = 0; i > 1; i--){
        /* i 表示圈中人个数，初始为 M 个，剩 1 个人时结束循环；j 表示当前报数人的位置 */
        for(k = 1; k <= N; k++)      /* 1 至 N 报数 */
            if(++j > i - 1) j = 0; /* 最后一个人报数后第一个人接着报，形成一个圈 */
        b[M-i] = j ? _____:_____; /* 将报数为 N 的人的编号存入数组 b */
        if(j)
            for(k = --j; k < i; k++) /* 压缩数组 a，使报数为 N 的人出圈 */
                _____;
    }
    for(i = 0; i < M-1; i++)          /* 按次序输出出圈人的编号 */
        printf("%6d", b[i]);
    printf("%6d\n", a[0]);           /* 输出圈中最后一个人的编号 */
    return 0;
}
```

填空后的程序：

```
#include<stdio.h>
#define M 10
#define N 3
int main(void)
{
    int a[M], b[M];    /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int i, j, k;
    for(i = 0; i < M; i++)          /* 对圈中人按顺序编号 1—M */
        a[i] = i + 1;
    for(i = M, j = 0; i > 1; i--){
        /* i 表示圈中人个数，初始为 M 个，剩 1 个人时结束循环；j 表示当前报数人的位置 */
        for(k = 1; k <= N; k++)      /* 1 至 N 报数 */
            if( ++j > i - 1) j = 0; /* 最后一个人报数后第一个人接着报，形成一个圈 */
        b[M-i] = ( j ? a[j-1] : a[1] ); /* 将报数为 N 的人的编号存入数组 b */
        if(j)
```

```

        for(k = --j ; k < i; k++)/* 压缩数组 a, 使报数为 N 的人出圈 */
            a[k] = a[k+1];
    }
    for(i = 0 ; i < M - 1 ; i++) /* 按次序输出出圈人的编号 */
        printf("%6d", b[i]);
    printf("%6d\n", a[0]); /* 输出圈中最后一个人的编号 */
    return 0;
}

/*
3 6 9 2 7 1 8 5 10 4
*/

```

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。

优化后的程序:

```

#include<stdio.h>

#define M 10
#define N 3
#define re register

int main(void)
{
    int vis[M+1] = {0}, now = 0;;
    for( re int i = 1 ; i <= M ; i++)
    {
        for ( re int j = 1 ; j <= N ; j++)
        {
            if ( ++ now > M ) now = 1 ;
            if ( vis[now] ) j-- ;
        }
        printf("%6d",now);
        vis[now] = 1 ;
    }
    return 0;
}

```

5.2.3 程序设计


以下 (1) 至 (3) 题对应 Educoder 教学平台 “C 语言实验”课程, 实验 5, 第 10 关实

1) 解题思路:

1. 将字符数组全部初始化为字符 0
2. 输入整数 x
3. 如果整数 x 的二进制末位为 1，则字符数组对应的位置为 1，反之为 0，并将整数 x 右移一位。
4. 输出字符数组 c
5. 结束

2) 程序清单

```
int main()
{
    scanf("%d",&x) ;
    for( re int i = 1 ; i <= 32 ; i++ )
    {
        c[i] = x&1 ? '1' : '0' ;
        x >>= 1 ;
    }
    for ( re int i = 32 ; i >= 1 ; i-- )
        printf ("%c",c[i]) ;
    return 0 ;
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\dekr...". The command prompt shows the command "7" followed by a series of zeros and the number "111", indicating the execution of a program that outputs a sequence of zeros followed by "111".

64

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

- ①“成绩输入”，输入 n 个学生的姓名和 C 语言课程的成绩。
- ②“成绩排序”，将成绩按从高到低的次序排序，姓名同时进行相应调整。
- ③“成绩输出”，输出排序后所有学生的姓名和 C 语言课程的成绩。
- ④“成绩查找”，输入一个 C 语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和 C 语言课程的成绩；否则，输出提示“not found!”。

1) 解题思路：

1.输入 order

2.对 order 进行判断：

2.1 如果 order 为 1，调用函数进行数据录入

2.2 如果 order 为 2，调用函数进行归并排序

2.3 如果 order 为 3，调用函数进行姓名、成绩的输出

2.4 如果 order 为 4，调用函数进行二分查找

2.5 如果 order 为 0，则跳出 while 循环

3. 结束

2) 程序清单

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

```
#define re register
```

```
#define maxn 1005
```

```
struct student
```

```
{
```

```
    char name[15];
```

```
    int score;
```

```
}stu[maxn],r[maxn];
```

```
int n,order, first = 1, last, cnt=0;
```

```
void func1()
```

```
{
```

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

```
    cnt += n;
```

```
    last = first + n - 1;
```

```
    for (re int i = first; i <= last; i++)
```

```
        scanf("%s%d",stu[i].name, &stu[i].score);
```

```
    printf("%d records were input!\n",n);
```

```
    first = last + 1;
```

```
}
```

```
void swap ( struct student *x, struct student *y )
```

```
{
```

```
    struct student tmp;
```

```
    tmp = *x;
```

```
    *x = *y;
```

```
    *y = tmp;
```

```
}
```

```

void msort ( int s , int t )
{
    if ( s == t ) return ;
    re int mid = ( s + t ) >> 1 ;
    msort ( s , mid ) ;
    msort ( mid + 1 , t ) ;
    int i = s , j = mid + 1 , k = s ;
    while ( i <= mid && j <= t )
    {
        if( stu[i].score >= stu[j].score )
            swap ( &r[k++] , &stu[i++] ) ;
        else
            swap ( &r[k++] , &stu[j++] ) ;
    }
    while ( i <= mid )
        swap ( &r[k++] , &stu[i++] ) ;
    while ( j <= t )
        swap ( &r[k++] , &stu[j++] ) ;
    for(re int i = s ; i <= t ; i ++ )
        swap ( &r[i] , &stu[i] ) ;
}

void func2()
{
    msort( 1 , cnt ) ;
    printf("Reorder finished!\n");
}

void func3()
{
    for( re int i = 1 ; i <= cnt ; i ++ )
        printf("%s %d\n",stu[i].name,stu[i].score);
}

void func4()
{
    int x , sign = 0 ;
    scanf("%d",&x) ;
    for ( re int i = 1 ; i <= cnt ; i ++ )
    {
        if ( stu[i].score == x )
        {
            printf("%s %d\n",stu[i].name,stu[i].score) ;
            sign = 1 ;
        }
    }
    if ( !sign ) printf("not found!\n") ;
}

```

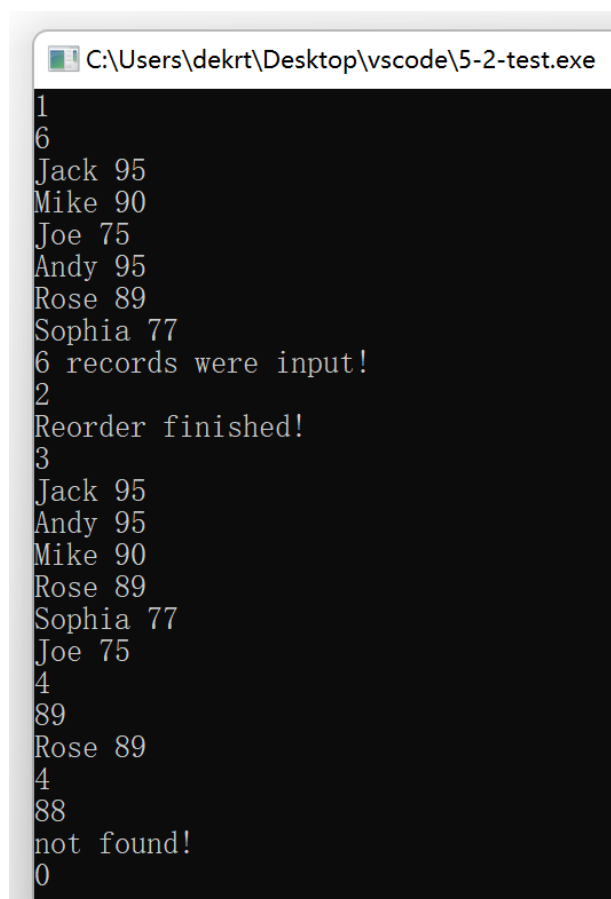
```

}

int main()
{
    while ( 1 )
    {
        scanf("%d",&order);
        if( order == 1 ) func1();
        if( order == 2 ) func2();
        if( order == 3 ) func3();
        if( order == 4 ) func4();
        if( order == 0 ) break ;
    }
    return 0 ;
}

```

3) 程序运行结果截图



```

C:\Users\dekr\Desktop\vscode\5-2-test.exe
1
6
Jack 95
Mike 90
Joe 75
Andy 95
Rose 89
Sophia 77
6 records were input!
2
Reorder finished!
3
Jack 95
Andy 95
Mike 90
Rose 89
Sophia 77
Joe 75
4
89
Rose 89
4
88
not found!
0

```

图 5-7 程序设计题 2 的程序运行截图

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

1) 解题思路：

1. 输入 n
2. 调用 search 函数进行搜索
3. 输出结果
4. 结束

2) 程序清单

```
#include<stdio.h>

int crax[2000],cray[2000],cx[2000];
int ans[2000];
int n,tot;

void search(int x)
{
    if (x>n) tot++ ;
    for (int y=1;y<=n;y++)
    {
        if (crax[x+y]) continue;
        if (cray[x-y+10]) continue;
        if (cx[y]) continue;
        ans[x]=y;
        crax[x+y]=1;
        cray[x-y+10]=1;
        cx[y]=1;
        search(x+1);
        crax[x+y]=0;
        cray[x-y+10]=0;
        cx[y]=0;
    }
}

int main(void)
{
    scanf("%d",&n);
    if ( n == 2 || n == 3 )
    {
        printf("无解");
        return 0;
    }
    search(1);
    printf("%d",tot);
    return 0;
}
```

3) 程序运行结果示意图

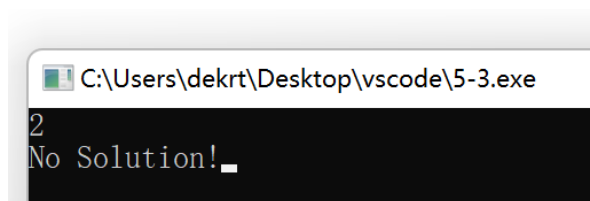


图 5-8 程序设计题 3 的程序运行截图

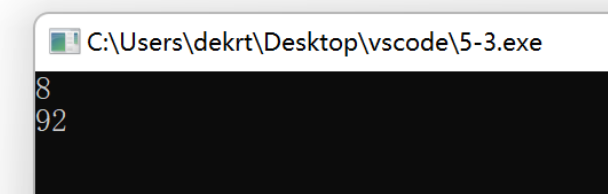


图 5-9 程序设计题 3 的程序运行截图

5.3 实验小结

- (1) 通过这次实验，我掌握了数组的说明、初始化和使用。
- (2) 通过这次实验，我掌握了一维数组作为函数参数时实参和形参的用法。
- (3) 通过这次实验，我掌握了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) 通过这次实验，我掌握了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。

实验 6 指针实验

6.1、实验目的

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

6.2、实验题目及要求

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     /*将字符串 s 复制给字符串 t, 并且返回串 t 的首地址*/

17     char * strcpy(char *t, const char *s)

18     {

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

20         return (t);

21     }

```

解答：

(1) 错误修改：

1) 第 5 行的字符数组定义错误，正确形式为：

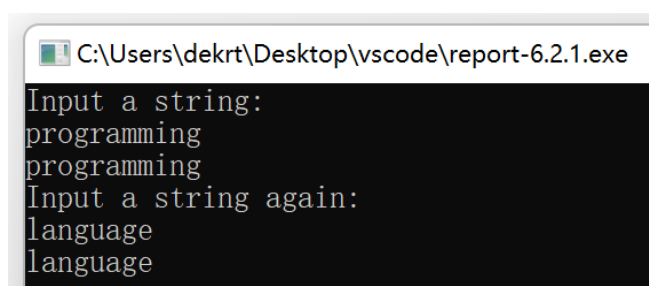
```
#define maxn 10005

char s1[maxn], s2[maxn], s3[maxn];
```

2) 第 12 行的语句错误，正确形式为：

```
strcpy(s3, s2);
```

(2) 错误修改后运行结果：



```
C:\Users\dekrt\Desktop\vscode\report-6.2.1.exe
Input a string:
programming
programming
Input a string again:
language
language
```

图 6-1 源程序改错题的程序运行结果示意图

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（字符串长度不超过 49）存入通过 `malloc` 动态分配的存储空间，然后调用 `strsort` 对这 N 个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

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

```
#include<_____>
```

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

```
#define N 4
```

```
/*对指针数组 s 指向的 size 个字符串进行升序排序*/
```

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



```

{
    _____temp;

    int i, j;

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

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

            if (_____)

            {

                temp = s[j];

                _____;

                s[j+1] = temp;

            }

}

int main()

{

    int i;

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

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

    {

        gets(t);

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

        strcpy(_____);
    }
}

```

```

    }

    strsort(_____);

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

    return 0;

}

```

解答：替换后的程序如下所示：

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define N 4
/*对指针数组 s 指向的 size 个字符串进行升序排序*/
void strsort(char *s[], int size)
{
    char *temp ;
    int i, j;
    for(i=0; i<size-1; i++)
        for (j=0; j<size-i-1; j++)
            if ( strcmp( s[j] , s[j+1]) > 0 )
            {
                temp = s[j];
                s[j] = s[j+1] ;
                s[j+1] = temp;
            }
}

int main()
{
    int i;
    char *s[N], t[50];
    for ( i=0 ; i<N ; i++)

```

```

{
    gets(t);
    s[i] = (char *)malloc(strlen(t)+1);
    strcpy( s[i] , t );
}
strsort( s , N );
for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}
return 0;
}
    
```

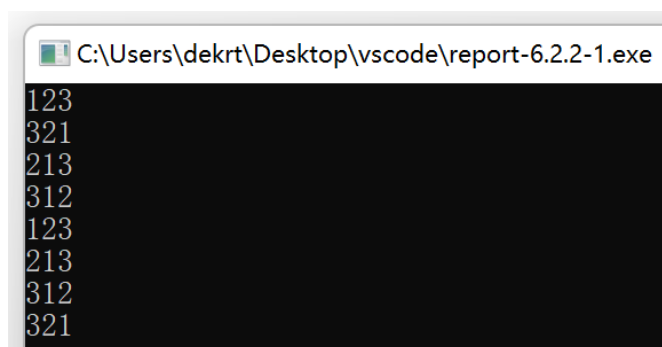


图 6-2 源程序替换题的程序运行结果示意图

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：替换后的程序如下所示：

```

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/*对指针数组 s 指向的 size 个字符串进行升序排序*/

void strsort(char **s , int size)

{
    
```

```

char *temp ;

int i, j;

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

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

        if ( strcmp( *( s + j ) , *( s + j + 1 ) ) > 0 )

            {

                temp = *( s + j ) ;

                ( *( s + j ) ) = *( s + j + 1 );

                *( s + j + 1 ) = temp;

            }

}

```

```

int main()

{

    int i;

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

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

    {

        gets(t);

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

        strcpy( s[i] , t );

    }

```

```

strsort( s , N );

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

return 0;

}

```

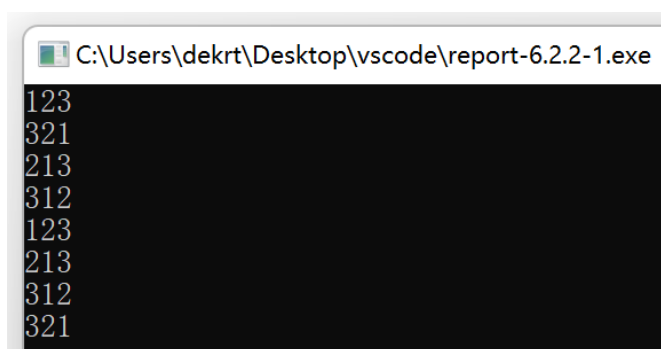


图 6-3 源程序替换题的程序运行结果示意图

(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.

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

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

```
int main (void)
```

```
{
```

```
    _____;
```

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

```
    int choice;
```

```
    while(1)
```

```
    {
```

```
        do
```

```
        {
```

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

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

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

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

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

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

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

```

switch(choice)
{
    case 1: p = strcpy; break;
    case 2: p = strcat; break;
    case 3: p = strtok; break;
    case 4: goto down;
}

getchar();

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

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

result = _____(a, b);

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

}

down:

return 0;

}

```

解答：替换后的程序如下所示：

```

#include<stdio.h>

#include<string.h>

```

```

int main (void)

{

    char* (*p)( char a[] , const char b[] );

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

    int choice;

    while(1)
    {
        do
        {

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

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

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

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

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

            scanf("%d", &choice);

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

        switch(choice)
        {

            case 1:    p = strcpy; break;

            case 2:    p = strcat; break;

            case 3:    p = strtok; break;

            case 4:    goto down;

```



```

    }

    getchar();

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

    scanf("%s",a);

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

    scanf("%s",b);

    result = p(a, b);

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

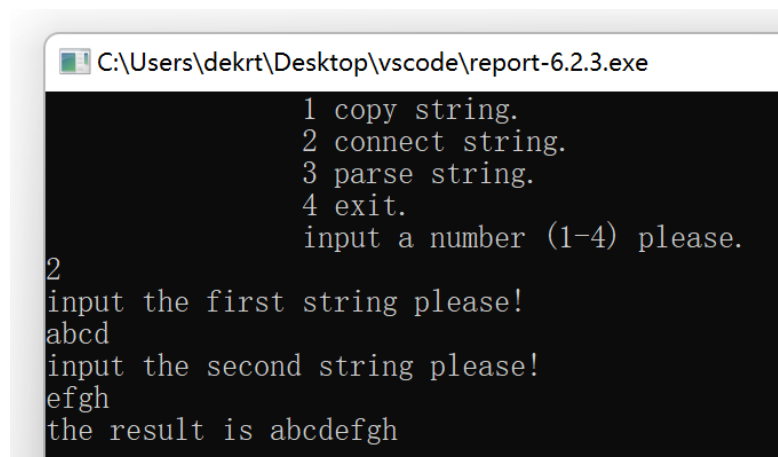
}

    down:

    return 0;

}

```



```

C:\Users\dekr\Desktop\vscode\report-6.2.3.exe
1 copy string.
2 connect string.
3 parse string.
4 exit.
input a number (1-4) please.
2
input the first string please!
abcd
input the second string please!
efgh
the result is abcdefgh

```

图 6-4 源程序修改替换题的程序运行结果示意图

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：替换后的程序如下所示：

```
# include<stdio.h>

# include<string.h>

char* (*ope[])( char a[] , const char b[] ) = { strcpy , strcat , strtok } ;

int main (void)
{
    char* (*p)( char a[] , const char b[] );

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

    int choice;

    while(1)
    {
        do
        {
            printf("\t\t1 copy string.\n");

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

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

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

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

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

```

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

    if ( choice == 4 )    goto down;

    getchar();

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

    scanf("%s",a);

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

    scanf("%s",b);

    result = (ope[choice-1]) (a, b);

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

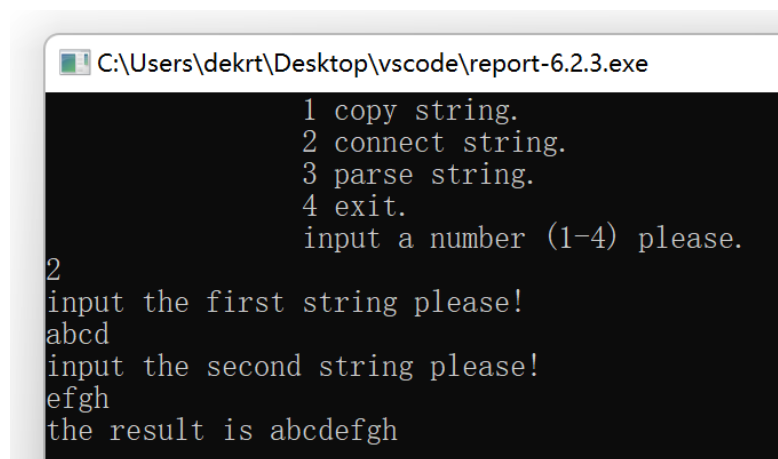
}

down:

return 0;

}

```



```

C:\Users\dekrt\Desktop\vscode\report-6.2.3.exe
1 copy string.
2 connect string.
3 parse string.
4 exit.
input a number (1-4) please.
2
input the first string please!
abcd
input the second string please!
efgh
the result is abcdefgh

```

图 6-5 源程序替换题的程序运行结果示意图

3、跟踪调试题

请按下面的要求对源程序进行操作，并回答问题和排除错误。

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

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

```
#include "stdio.h"

char *strcpy(char *,char *);

void main(void)
{
    char a[20],b[60]="there is a boat on the lake.";

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

}

char *strcpy(char *s,char *t)
{
    while(*s++=*t++)
        ;

    return (s);
}
```

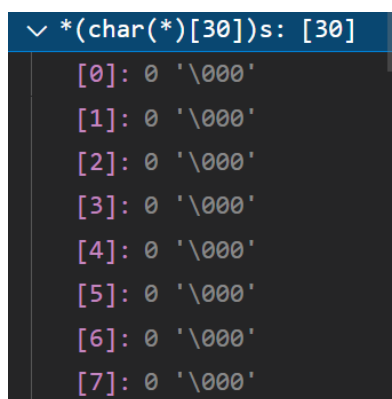


图 6-6 跟踪调试题的程序运行结果示意图

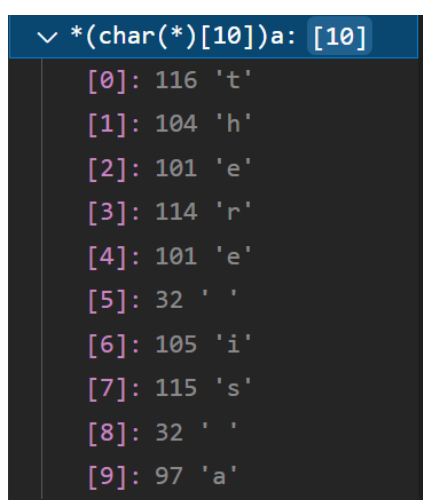


图 6-7 跟踪调试题的程序运行结果示意图

替换后的程序如下所示：

```
#include "stdio.h"

char *strcpy(char *,char *);

int main(void)
{
    char a[20],b[60]="there is a boat on the lake.";

    strcpy(a,b);

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

    return 0 ;
}
```

```

}

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

{

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

    return (s);

}

```

4、编程设计题

(1) 指定 main 函数的参数

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

1) 解题思路：

- 1.输入 n 个字符串
- 2.输出 n 个字符串
- 3.结束

2) 程序清单

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

```
#define re register
```

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

```
{
    for( re int i = 0 ; i < argc ; i++ )
    {
        printf("%s\n",argv[i]) ;
    }
    return 0 ;
}
```

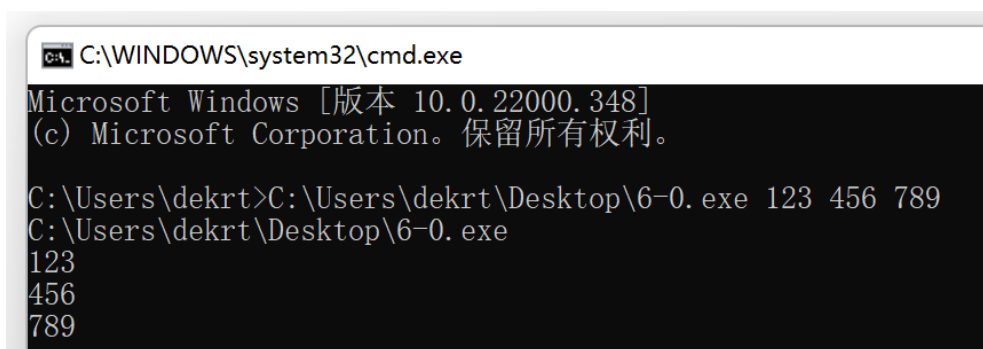


图 6-8 程序设计题 1 的程序运行结果示意图

以下 (2) 至 (5) 题对应 Educoder 教学平台“C 语言实验”课程，实验 6，第 13 关实验 6-1、第 14 关实验 6-2、第 15 关实验 6-3，以及第 16 关实验 6-4。

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

样例输入：15

样例输出：0000000F

1) 解题思路：

1.输入 k

2.定义指针 p，类型为 unsigned short

3.依次取出对应位置上的数字并输出

4.结束

2) 程序清单

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

```
#define re register
```

```
unsigned long int k ;
```

```
int main()
```

```
{
```

```
    scanf ("%u",&k) ;
```

```
    unsigned short *p = (unsigned short*) &k ;
```

```
    printf("%.4X%.4X",*(p+1),*p) ;
```

```
    return 0 ;
```

```
}
```

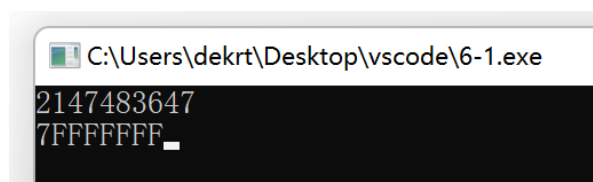


图 6-9 程序设计题 2 的程序运行结果示意图

(3) 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。

样例输入：

```
2 3
1 5 3
3 2 4
```

样例输出：

```
3 4
5 2
1 3
```

1) 解题思路：

- 1.输入 n,m
- 2.输入对应的矩阵
- 3.对矩阵进行旋转并输出
- 4.结束

2) 程序清单

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

```
#define re register
```

```
#define maxn 10
```

```
int n , m ;

int a[maxn][maxn] ;

int main()
{
    scanf("%d%d",&n,&m) ;

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

        for( re int j = 1 ; j <= m ; j++)

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

    for ( re int i = m ; i >= 1 ; i-- )
    {
        for( re int j = 1 ; j <= n-1 ; j++ )

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

        printf("%d\n",a[n][i]) ;
    }

    return 0 ;
}
```

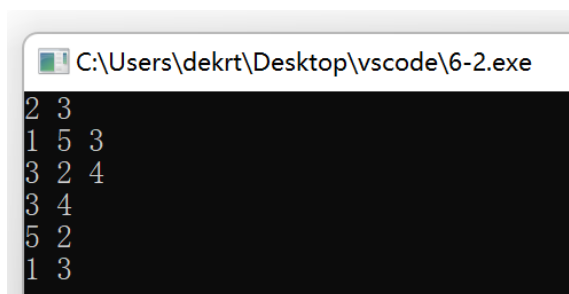


图 6-10 程序设计题 3 的程序运行结果示意图

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

1) 解题思路：

- 1.读入字符串 s
- 2.调用 del()函数进行删除
- 3.在主函数中对字符串进行输出
- 4.结束

2) 程序清单

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

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

```
#define re register
```

```
#define maxn 100
```

```
char s[maxn] ;
```

```
int cnt ;
```

```
void del( char s[] )
```

```
{
```

```
    cnt = 0 ;
```

```
    char tmp[maxn] ;
```

```

    re int len = strlen(s) , i = 0 , j , k = 0 ;

    while ( s[i++] == ' ' );

    i-- ;

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

    if ( s[j] != '\t' )

        tmp[k++] = s[j] ;

    cnt = strlen(tmp) ;

    for( re int i = 0 ; i < cnt ; i++ )

        s[i] = tmp[i] ;

}

int main()

{

    while ( gets(s) )

    {

        del(s) ;

        for ( re int i = 0 ; i < cnt ; i++ )

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

        putchar('\n') ;

    }

    return 0 ;

}

```

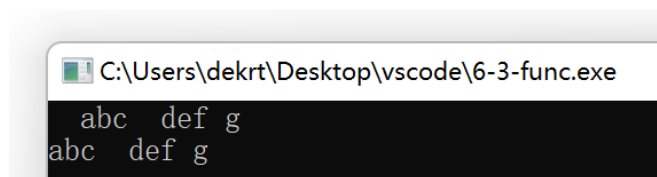


图 6-11 程序设计题 4 的程序运行结果示意图

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

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

②scheduler 函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350 并回车，则 scheduler 函数一次调度选择 task1, task3, 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此调用执行选定的任务函数。

例如，当输入 13607122 并回车，程序运行结果如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

1) 解题思路：

- 1.在 main()函数中调用 scheduler()函数
- 2.读入字符串，并将其转化为数字
- 3.调用对应的 task 函数
- 4.结束

2) 程序清单

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

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

```
#define re register
```

```
#define maxn 1005
```

```
char s[maxn] ;
```

```
int a[maxn] ;
```

```
inline void task0()
```

```
{puts("task0 is called!") ;}
```

```
inline void task1()
```

```
{puts("task1 is called!") ;}
```

```
inline void task2()
```

```
{puts("task2 is called!") ;}
```

```
inline void task3()
```

```
{puts("task3 is called!") ;}
```

```
inline void task4()
```

```
{puts("task4 is called!") ;}
```

```
inline void task5()
```

```
{puts("task5 is called!") ;}
```

```
inline void task6()
```

```
{puts("task6 is called!") ;}
```

```
inline void task7()
```

```
{puts("task7 is called!") ;}
```

```
inline void execute ( int arr[] , int cnt )
```

```
{
    for( re int i = 0 ; i < cnt ; i++ )
    {
        if ( a[i] == 0 ) task0() ;
        if ( a[i] == 1 ) task1() ;
        if ( a[i] == 2 ) task2() ;
        if ( a[i] == 3 ) task3() ;
        if ( a[i] == 4 ) task4() ;
        if ( a[i] == 5 ) task5() ;
        if ( a[i] == 6 ) task6() ;
        if ( a[i] == 7 ) task7() ;
    }
}
```

```
inline void scheduler()
{
    scanf("%s",s) ;
    int cnt = strlen(s) ;
    for ( re int i = 0 ; i < cnt ; i++ )
        a[i] = s[i] - '0' ;
    execute ( a , cnt ) ;
}
```



```
int main()

{

    scheduler();

    return 0 ;

}
```

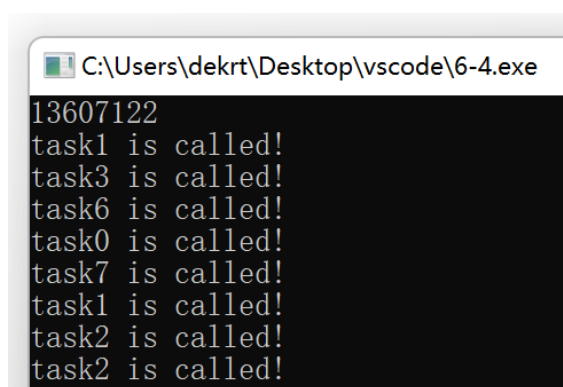


图 6-12 程序设计题 5 的程序运行结果示意图

6.3 实验小结

- (1) 通过这次实验，我熟练掌握了指针的说明、赋值、使用。
- (2) 通过这次实验，我掌握了用指针引用数组的元素，熟悉指向数组的指针的使用。
- (3) 通过这次实验，我熟练掌握了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) 通过这次实验，我掌握了指针函数与函数指针的用法。
- (5) 通过这次实验，我掌握了带有参数的 main 函数的用法。

实验 7 结构与联合实验

7.1 实验目的

1. 通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。
2. 通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。
3. 了解字段结构和联合的用法。

7.2 实验题目及要求

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	$(++p) \rightarrow x$	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	$++*p \rightarrow t$	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：

替换后的程序如下所示：

```

#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);
int 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");
    return 0 ;
}
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 */
}

```

程序运行截图如图所示：

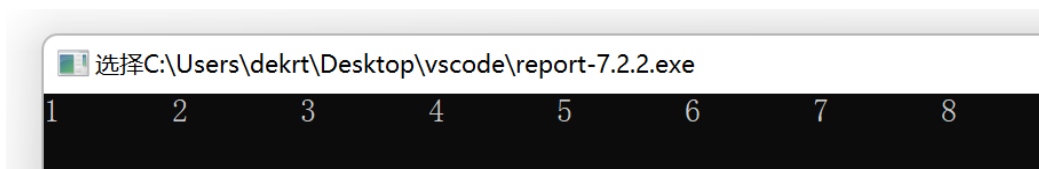


图 7-1 源程序修改替换题的程序运行结果示意图

(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：

替换后的程序如下所示

```

#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);
int 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");
    return 0 ;
}
void create_list(struct s_list **headp,int *p)
{
    struct s_list * loc_head=NULL, * tail , * tmp = NULL ;
    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 */
            tmp = (struct s_list *)malloc(sizeof(struct s_list));
            tmp->data = *p++;
            tmp->next = loc_head ;
            loc_head = tmp ;
            // loc_head->data = tmp->data ;
        }
        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 */
}

```

程序运行截图如图所示：

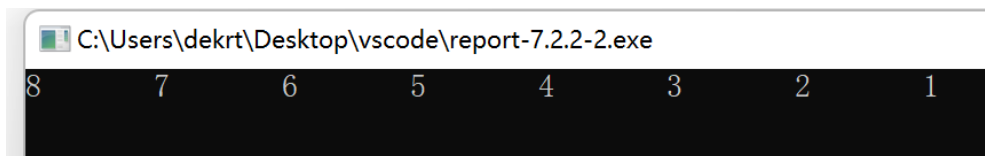


图 7-2 源程序修改替换题的程序运行结果示意图

7.2.3 程序设计

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

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

- ① 输入每个学生的各项信息。
- ② 输出每个学生的各项信息。
- ③ 修改指定学生的指定数据项的内容。
- ④ 统计每个同学的平均成绩（保留 2 位小数）。
- ⑤ 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

1) 解题思路：

- 1.输入指令 order，读入 0 或 EOF 时停止读入
- 2.调用相应的函数进行处理
 - 2.1 如果 order = 1，进行数据的输入
 - 2.2 如果 order = 2，进行数据的输出
 - 2.3 如果 order = 3，进行数据的修改
 - 2.4 如果 order = 4，进行数据的平均值的计算并输出

2.5 如果 order = 5，进行数据的输出

3. 结束

2) 程序清单

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

#define re register
#define maxn 20

struct node
{
    int c_lan , English , math , physics ;
    char num[maxn] , name[maxn] ;
    struct node *next ;
    double average ;
} *head = NULL , *tail , *p ;

void fun1()
{
    int n ;
    scanf("%d",&n) ;
    head = (struct node *) malloc ( sizeof ( struct node ) ) ;
    scanf("%s%s%d%d%d%d", head->num , head->name , &head->English ,
&head->math , &head->physics , &head->c_lan ) ;
    tail = head ;
    for( re int i = 1 ; i < n ; i++ )
    {
        tail->next = (struct node *) malloc ( sizeof ( struct node ) ) ;
        tail = tail->next ;
        scanf("%s%s%d%d%d%d", tail->num , tail->name , &tail->English ,
&tail->math , &tail->physics , &tail->c_lan ) ;
    }
}
```



```

tail->next = NULL ;
struct node * tmp = head ;
while ( tmp )
{
    int sum = tmp->English + tmp->math + tmp->physics +
tmp->c_lan ;
    tmp->average = 1.0*sum/4 ;
    tmp = tmp->next ;
}
// puts("finish!") ;
printf("完成了%d 位同学的成绩输入。\\n",n) ;
}

```

```

void fun2()
{
    struct node * tmp = head ;
    while ( tmp )
    {
        printf("%s %s %d %d %d %d\\n", tmp->num , tmp->name ,
tmp->English , tmp->math , tmp->physics , tmp->c_lan ) ;
        tmp = tmp->next ;
    }
}

```

```

void fun3()
{
    struct node * tmp = head ;
    char ctmp[maxn] ;
    int task , target ;
    scanf("%s%d" , ctmp , &task ) ;
    while ( tmp )
    {
        if( !strcmp( ctmp , tmp->num ) )
        {

```

```

        if( task == 0 )
        {
            char ntmp[maxn] ;
            scanf("%s",ntmp) ;
            strcpy( tmp->name , ntmp ) ;
        }
        if( task == 1 )
        {
            scanf("%d",&target) ;
            tmp->English = target ;
        }
        if( task == 2 )
        {
            scanf("%d",&target) ;
            tmp->math = target ;
        }
        if( task == 3 )
        {
            scanf("%d",&target) ;
            tmp->physics = target ;
        }
        if( task == 4 )
        {
            scanf("%d",&target) ;
            tmp->c_lan = target ;
        }
        printf("%s %s %d %d %d %d\n", tmp->num , tmp->name ,
tmp->English , tmp->math , tmp->physics , tmp->c_lan ) ;
        break ;
    }
    tmp = tmp->next ;
}
}

```

```

void fun4()
{
    struct node * tmp = head ;
    while ( tmp )
    {
        int sum = tmp->English + tmp->math + tmp->physics +
tmp->c_lan ;
        tmp->average = 1.0*sum/4 ;
        printf("%s %s %.2lf\n",tmp->num,tmp->name,tmp->average) ;
        tmp = tmp->next ;
    }
}

```

```

void fun5()
{
    struct node * tmp = head ;
    while ( tmp )
    {
        int sum = tmp->English + tmp->math + tmp->physics +
tmp->c_lan ;
        printf("%s %s %d %.2lf\n",tmp->num,tmp->name,sum,tmp->average) ;
        tmp = tmp->next ;
    }
}

```

```

void fun6()
{
    struct node *tmp = head ;
    while ( tmp->next )
    {
        struct node *now = tmp ;
        while ( now->next )
        {
            if( now->average > now->next->average )

```

```

        {
            struct node tmpp = *now ;
            tmpp.next = now->next->next ;
            now->next->next = now->next ;
            *now = *now->next ;
            *now->next = tmpp ;
        }
        now = now->next ;
    }
    tmp = tmp->next ;
}
// puts("YES!!!") ;
}

int main()
{
    int order ;
    while ( scanf("%d",&order) != EOF )
    {
        if( order == 1 ) fun1() ;
        if( order == 2 ) fun2() ;
        if( order == 3 ) fun3() ;
        if( order == 4 ) fun4() ;
        if( order == 5 ) fun5() ;
        if( order == 6 ) fun6() ;
        if( order == 0 ) break ;
    }
    return 0 ;
}

```

3)程序运行截图

```

C:\Users\dekr\Desktop\7-1.exe
1
5
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
2021004 Andy 95 97 92 95
2021005 Rose 90 87 88 89
完成了5位同学的成绩输入。
2
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
2021004 Andy 95 97 92 95
2021005 Rose 90 87 88 89
3
2021003 0 Joan
2021003 Joan 77 86 90 75
4
2021001 Jack 91.00
2021002 Mike 80.00
2021003 Joan 82.00
2021004 Andy 94.75
2021005 Rose 88.50
5
2021001 Jack 364 91.00
2021002 Mike 320 80.00
2021003 Joan 328 82.00
2021004 Andy 379 94.75
2021005 Rose 354 88.50
    
```

图 7-3 程序设计题 1 的程序运行结果示意图

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

1) 解题思路：

1.输入指令 order，读入 0 或 EOF 时停止读入

2.调用相应的函数进行处理

2.1 如果 order = 1，进行数据的输入

2.2 如果 order = 2，进行数据的输出

2.3 如果 order = 3，进行数据的修改

2.4 如果 order = 4，进行数据的平均值的计算并输出

2.5 如果 order = 5，进行数据的输出

2.6 如果 order = 6，进行数据的排序，运用冒泡排序交换节点域

3. 结束

2) 程序清单

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

#define re register
#define maxn 20

struct node
{
    int c_lan , English , math , physics ;
    char num[maxn] , name[maxn] ;
    struct node *next ;
    double average ;
} *head = NULL , *tail , *p ;

void fun1()
{
    int n ;
    scanf("%d",&n) ;
    head = (struct node *) malloc ( sizeof ( struct node ) ) ;
    scanf("%s%s%d%d%d%d", head->num , head->name , &head->English ,
    &head->math , &head->physics , &head->c_lan ) ;
    tail = head ;
    for( re int i = 1 ; i < n ; i++ )
    {
        tail->next = (struct node *) malloc ( sizeof ( struct node ) ) ;
        tail = tail->next ;
        scanf("%s%s%d%d%d%d", tail->num , tail->name , &tail->English ,
        &tail->math , &tail->physics , &tail->c_lan ) ;
```

```

    }
    tail->next = NULL ;
    struct node * tmp = head ;
    while ( tmp )
    {
        int sum = tmp->English + tmp->math + tmp->physics +
tmp->c_lan ;
        tmp->average = 1.0*sum/4 ;
        tmp = tmp->next ;
    }
    // puts("finish!") ;
    printf("完成了%d 位同学的成绩输入。 \n",n) ;
}

void fun2()
{
    struct node * tmp = head ;
    while ( tmp )
    {
        printf("%s %s %d %d %d %d\n", tmp->num , tmp->name ,
tmp->English , tmp->math , tmp->physics , tmp->c_lan ) ;
        tmp = tmp->next ;
    }
}

void fun3()
{
    struct node * tmp = head ;
    char ctmp[maxn] ;
    int task , target ;
    scanf("%s%d" , ctmp , &task ) ;
    while ( tmp )
    {
        if( !strcmp( ctmp , tmp->num ) )

```

```

{
    if( task == 0 )
    {
        char ntmp[maxn] ;
        scanf("%s",ntmp) ;
        strcpy( tmp->name , ntmp ) ;
    }
    if( task == 1 )
    {
        scanf("%d",&target) ;
        tmp->English = target ;
    }
    if( task == 2 )
    {
        scanf("%d",&target) ;
        tmp->math = target ;
    }
    if( task == 3 )
    {
        scanf("%d",&target) ;
        tmp->physics = target ;
    }
    if( task == 4 )
    {
        scanf("%d",&target) ;
        tmp->c_lan = target ;
    }
    printf("%s %s %d %d %d %d\n", tmp->num , tmp->name ,
tmp->English , tmp->math , tmp->physics , tmp->c_lan ) ;
    break ;
}
tmp = tmp->next ;
}
}

```



```

void fun4()
{
    struct node * tmp = head ;
    while ( tmp )
    {
        int sum =    tmp->English + tmp->math + tmp->physics +
tmp->c_lan ;
        tmp->average = 1.0*sum/4 ;
        printf("%s %s %.2lf\n",tmp->num,tmp->name,tmp->average) ;
        tmp = tmp->next ;
    }
}

void fun5()
{
    struct node * tmp = head ;
    while ( tmp )
    {
        int sum =    tmp->English + tmp->math + tmp->physics +
tmp->c_lan ;
        printf("%s %s %d %.2lf\n",tmp->num,tmp->name,sum,tmp->average) ;
        tmp = tmp->next ;
    }
}

void fun6()
{
    struct node *tmp = head ;
    while ( tmp->next )
    {
        struct node *now = tmp ;
        while ( now->next )
        {

```

```

        if( now->average > now->next->average )
        {
            struct node tmpp = *now ;
            tmpp.next = now->next->next ;
            now->next->next = now->next ;
            *now = *now->next ;
            *now->next = tmpp ;
        }
        now = now->next ;
    }
    tmp = tmp->next ;
}
// puts("YES!!!") ;
fun4() ;
}

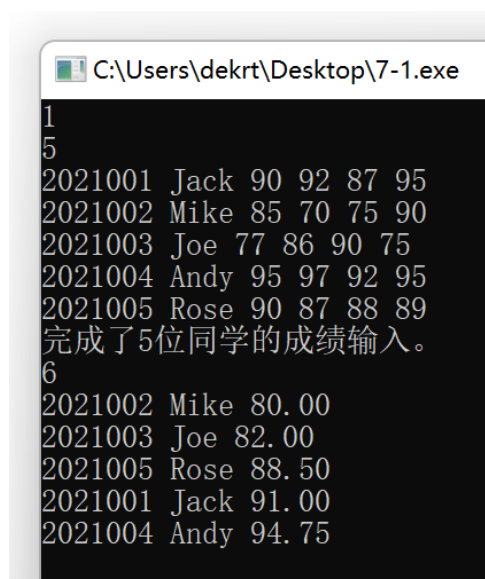
```

```

int main()
{
    int order ;
    while ( scanf("%d",&order) != EOF )
    {
        if( order == 1 ) fun1() ;
        if( order == 2 ) fun2() ;
        if( order == 3 ) fun3() ;
        if( order == 4 ) fun4() ;
        if( order == 5 ) fun5() ;
        if( order == 6 ) fun6() ;
        if( order == 0 ) break ;
    }
    return 0 ;
}

```

3)程序运行截图



```

C:\Users\dekr\Desktop\7-1.exe
1
5
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
2021004 Andy 95 97 92 95
2021005 Rose 90 87 88 89
完成了5位同学的成绩输入。
6
2021002 Mike 80.00
2021003 Joe 82.00
2021005 Rose 88.50
2021001 Jack 91.00
2021004 Andy 94.75
    
```

图 7-4 程序设计题 2 的程序运行结果示意图

7.3 实验小结

- 1 .通过实验,我熟悉和掌握了结构的说明和引用、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,我掌握动态了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 .通过实验,我了解了字段结构和联合的用法。

实验 8 文件操作实验

8.1 实验目的

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

8.2 实验题目及要求

1. 文件类型的程序验证题

设有程序:

```
#include <stdio.h>
int main(void)
{
    short a=0x253f,b=0x7b7d;
    char ch;
    FILE *fp1,*fp2;
    fp1=fopen("d:\\abc1.bin","wb+");
    fp2=fopen("d:\\abc2.txt","w+");
    fwrite(&a,sizeof(short),1,fp1);
    fwrite(&b,sizeof(short),1,fp1);
    fprintf(fp2,"%hx %hx",a,b);

    rewind(fp1); rewind(fp2);
    while((ch = fgetc(fp1)) != EOF)
        putchar(ch);
    putchar('\n');

    while((ch = fgetc(fp2)) != EOF)
        putchar(ch);
    putchar('\n');

    fclose(fp1);
    fclose(fp2);
    return 0;
}
```

- (1) 请思考程序的输出结果, 然后通过上机运行来加以验证。

预测结果:

? %X

253f 7b7d

实际结果:

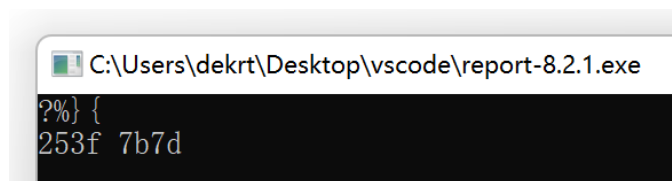


图 8-1 程序验证题的程序运行结果示意图

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

结果：

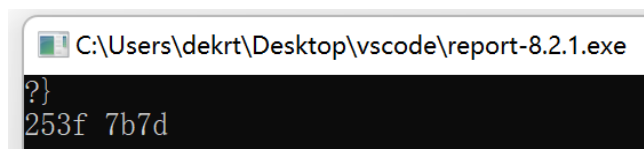


图 8-2 程序验证题的程序运行结果示意图

原因：char 大小为一个字节，写入 fp1 时只能写入 8 位，故只能对应显示高位对应的字符

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

将以十进制的形式写入文件

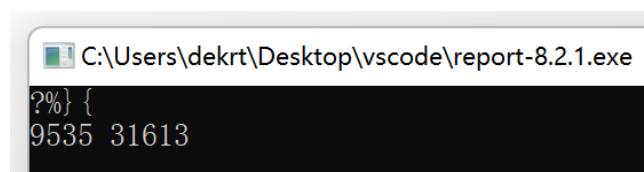


图 8-3 程序验证题的程序运行结果示意图

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 */

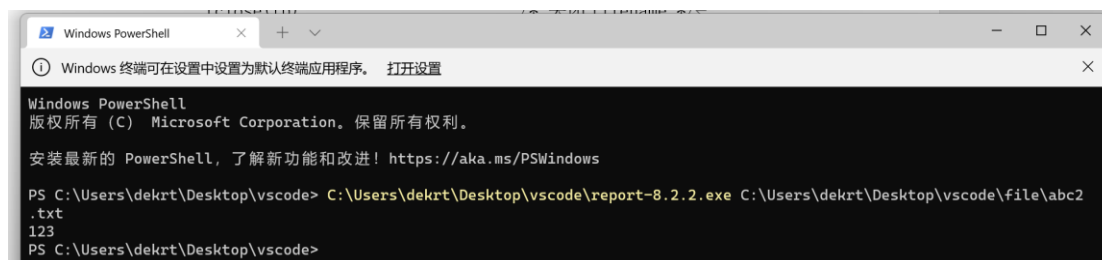
    putchar(ch);                            /* 向显示器中写字符 */

fclose(fp);                                /* 关闭 filename */

return 0;
}

```

修改：将 while(ch=fgetc(fp)!=EOF) 中 ch=fgetc(fp) 加上括号



```

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！ https://aka.ms/PSWindows

PS C:\Users\dekart\Desktop\vscode> C:\Users\dekart\Desktop\vscode\report-8.2.2.exe C:\Users\dekart\Desktop\vscode\file\abc2.txt
123
PS C:\Users\dekart\Desktop\vscode>

```

图 8-4 源程序修改替换题的程序运行结果示意图

(4) 用输入输出重定向 freopen 改写 main 函数。

修改后的程序：

```

#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((freopen(argv[1],"r",stdin))==NULL)
    {
        /* fp 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;
}

```

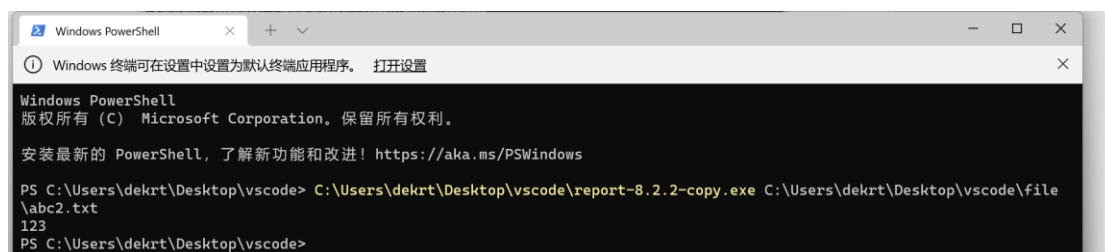


图 8-5 源程序修改替换题的程序运行结果示意图

3. 程序设计

以下 (1) 题对应 Educoder 教学平台“C 语言实验”课程，实验 8，第 20 关实验 8-1。

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

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

若文件为

```

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

```

样例输入：`you they`

样例输出：

```

`3`

```

```

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

```

1) 解题思路：

1. 读入需要替换的字符串 s1,s2
2. 使用 freopen 函数进行重定向，读取相应文件中的字符串
3. 使用字符串处理函数进行查找，替换，拼接
4. 进行处理后的字符串的输出

2) 程序清单

```

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

#define re register
#define maxn 10005

char s[maxn] , s1[maxn] , s2[maxn] , ans[maxn] , space[]=" " , tmp[maxn] , *p ;
int cnt ;

int main()
{
    scanf("%s%s" , s1 , s2 ) ;
    // freopen("experiment/src/step8/source.txt","r",stdin) ;
    while ( scanf("%s",s) != EOF)
    {
        if( ( p=strstr(s,s1) ) != NULL )
        {
            re int num = 0 ;
            for( num = 0 ; s[num] != *p ; num++ )
                tmp[num] = s[num] ;
            strcat( ans , tmp ) ;
            memset( tmp , 0 , sizeof(tmp) ) ;
            strcat( ans , s2 ) ;
            for ( re int i = 0 , j = strlen(s1) + num ; s[j] ; j++,i++ )
                tmp[i] = s[j] ;
            strcat( ans , tmp ) ;
            memset( tmp , 0 , sizeof(tmp) ) ;
            cnt++ ;
        }
        else
        {
            strcat(ans,s) ;
        }
        strcat(ans,space) ;
    }
    printf("%d\n",cnt) ;
    printf("%s",ans) ;
    return 0 ;
}

```

3) 程序运行结果示意图


```

C:\Users\dekr\Desktop\vscode\8-1.exe
you y
May you have enough happiness to make you sweet,enough trials to make you strong,enough sorrow to
keep you human,enough hope to make you happy? Always put yourself in others' shoes.If you feel that
it hurts you,it probably hurts the other person, too.
^Z
8
May y have enough happiness to make y sweet,enough trials to make y strong,enough sorrow to keep y
human,enough hope to make y happy? Always put yrself in others' shoes.If y feel that it hurts y,it
probably hurts the other person, too.
    
```

图 8-6 程序设计题的程序运行结果示意图

8.3 实验小结

- (1) 通过这次实验，我熟悉了文本文件和二进制文件在磁盘中的存储方式；
- (2) 通过这次实验，我熟练掌握了流式文件的读写方法。