

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： 计算机类 1709 班

学 号： U201714761

姓 名： 胡澳

指导教师： 甘早斌

报告日期： 2018 年 1 月 17 日

计算机科学与技术学院

目 录

1	表达式和标准输入与输出实验	1
1.1	实验目的.....	1
1.2	实验内容.....	1
1.3	实验小结.....	6
2	流程控制实验	7
2.1	实验目的.....	7
2.2	实验内容.....	7
2.3	实验小结.....	18
3	函数与程序结构实验	19
3.1	实验目的.....	19
3.2	实验内容.....	19
3.3	实验小结.....	31
4	编译预处理实验	32
4.1	实验目的.....	32
4.2	实验题目及要求.....	32
4.3	实验小结.....	39
5	数组实验	40
5.1	实验目的.....	40
5.2	实验内容及要求.....	40
5.3	实验小结.....	55
6	指针实验	56
6.1	实验目的.....	56
6.2	实验题目及要求.....	56
6.3	指定 main 函数的参数.....	71
6.4	实验小结.....	71
7	结构与联合实验	72
7.1	实验目的.....	72
7.2	实验题目及要求.....	72
7.3	实验小结.....	88
8	文件实验	89
8.1	实验目的.....	89
8.2	实验题目及要求.....	89
8.3	实验小结.....	93
	参考文献	94

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 源程序改错题

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。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

```
1  #include<stdio.h>
2  #define PI 3.14159;
3  void main( 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);

```

```

23 }

```

解答:

(1) 错误修改:

- 1) 第 2 行不能加;
正确形式为: # define PI 3.14159
- 2) 第 10 行 f 前缺少 &
正确形式为: scanf("%d", &f) ;
- 3) 第 11 行将 int 类型赋给 double 类型
正确形式为: c = 5.0/9*(f-32) ;
- 4) 第 15 行 f 前缺少 l
正确形式为: scanf("%lf", &r);
- 5) 第 17 行 s 前面多出一个 &
正确形式为: printf("\nThe acreage is %.2f\n\n",s);
- 6) 第 20 行两个 %x 不能用于输入 short 类型的变量 p,k
正确形式为: scanf("%hd %hd", &k, &p);
- 7) 第 21 行使用没有定义的变量 newint
正确形式为: 在 7、8 行之间添加 short newint;
- 8) 第 21 行语义错误
正确形式为: newint = (p&0xff00)|((k&0xff00)>>8);
- 9) 第 22 行输出 short 类型的 newint 不能使用 %x
正确形式为: printf("new int = %hd\n\n",newint);

(2) 错误修改后运行结果如图 1-1:

```
homework — -bash — 103x28
Last login: Sun Nov 19 12:21:30 on ttys003
huaodeMacBook-Air:~ huao$ cd /Users/huao/Documents/C/homework/homework
huaodeMacBook-Air:homework huao$ gcc main.c -o homework.out
huaodeMacBook-Air:homework huao$ ./homework.out
Input Fahrenheit:15

15(F) = -9.44(C)

input the radius r:3

The acreage is 28.27

input hex int k, p:115 543
new int = 512

huaodeMacBook-Air:homework huao$
```

图 1-1

1.2.2 源程序修改替换题

下面的程序利用常用的中间变量法实现两数交换，请改用不用第三个变量的交换法实现。

```
#include<stdio.h>

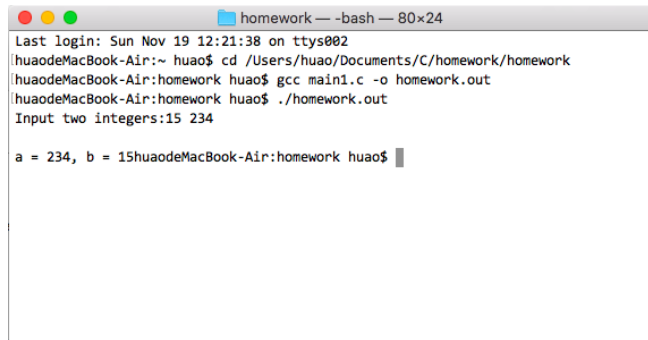
void main( )
{
    int a, b, t;
    printf("Input two integers:");
    scanf("%d %d",&a,&b);
    t=a, a=b, b=t;
    printf("\na=%d,b=%d",a,b);
}
```

解答：

(1) 替换修改后的程序如下：

```
# include <stdio.h>
int main(void)
{
    int a, b;
    printf("Input two integers:");
    scanf("%d %d", &a, &b);
    a = a + b;
    b = a - b;
    a = a - b;
    printf("\na = %d, b = %d", a, b);
    return 0;
}
```

(2) 运行结果如图 1-2:



```
homework -- bash -- 80x24
Last login: Sun Nov 19 12:21:38 on ttys002
hualdeMacBook-Air:~ huao$ cd /Users/hualde/Documents/C/homework/homework
hualdeMacBook-Air:homework huao$ gcc main1.c -o homework.out
hualdeMacBook-Air:homework huao$ ./homework.out
Input two integers:15 234

a = 234, b = 15hualdeMacBook-Air:homework huao$
```

图 1-2

1.2.3 编程设计题

上机调试运行以下程序：

- (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) 源程序清单

```
#include <stdio.h>

int main(void)
{
    char c;
    scanf("%c", &c);
    if(c >= 'A' && c <= 'Z')
        putchar(c + 0x20);
    else
        putchar(c);
    putchar('\n');
    return 0;
}
```

2) 测试

- a) 测试数据：A d
- b) 对于测试数据的运行结果截图如图 1-3

```
桌面 — -bash— 80x24
[huaodeMacBook-Air:Desktop huao$ gcc main.c -o homework.out
[huaodeMacBook-Air:Desktop huao$ ./homework.out
A
a
[huaodeMacBook-Air:Desktop huao$ ./homework.out
d
d
huaodeMacBook-Air:Desktop huao$
```

图 1-3

(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) 源程序清单

```
#include <stdio.h>
#include <limits.h>
int main(void)
{
    unsigned short int x, m, n, x1;
    scanf("%hu %hu %hu", &x, &m, &n);
    x1 = x;
    x >>= (m + n);
    x <<= (m + n);
    x1 <<= (16 - m);
    x1 >>= (16 - m - n);
    printf("%hu\n", x | x1);
    return 0;
}
```

2) 测试

a) 测试数据 125 5 3

b) 对测试数据的运行结果截图如图 1-4

```
桌面 — -bash— 80x24
[huaodeMacBook-Air:Desktop huao$ ./homework.out
125 5 3
232
huaodeMacBook-Air:Desktop huao$
```

图 1-4

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

解答：

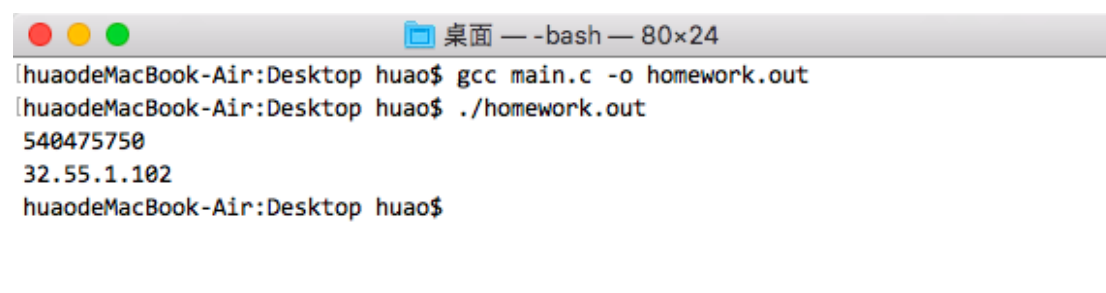
1) 源程序清单

```
#include <stdio.h>
int main(void)
{
    unsigned long int IP;
    unsigned long int a, b, c, d;
    scanf("%lu", &IP);
    a = (IP & 0xff000000) >> 24;
    b = (IP & 0x00ff0000) >> 16;
    c = (IP & 0x0000ff00) >> 8;
    d = IP & 0x000000ff;
    printf("%lu.%lu.%lu.%lu\n", a, b, c, d);
    return 0;
}
```

2) 测试

a) 测试数据：540475750

b) 对测试数据的运行结果截图如图 1-5



```
桌面 — -bash — 80x24
[huaodeMacBook-Air:Desktop huao$ gcc main.c -o homework.out
[huaodeMacBook-Air:Desktop huao$ ./homework.out
540475750
32.55.1.102
huaodeMacBook-Air:Desktop huao$
```

图 1-5

1.3 实验小结

要不断熟悉 C 语言的使用。

2 流程控制实验

2.1 实验目的

- (1) 掌握复合语句、if 语句、switch 语句的使用，熟练掌握 for、while、do-while 三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。
- (2) 练习循环结构 for、while、do-while 语句的使用。
- (3) 练习转移语句和标号语句的使用。
- (4) 使用 Turbo C 2.0 集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

2.2 实验内容

2.2.1 源程序改错题

下面是计算 $s=n!$ 的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。例如： $8!=40320$ 。

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

解答：

(1) 错误修改

- 1) 第 6 行 `scanf` 中的 `n` 前面要加上 `&`

正确形式为: `scanf("%d", &n);`

- 2) 第 7 行中 `for` 后面括号里面的表达式要用分号隔开

正确形式为: `for(i=1;i<=n;i++)`

(2) 错误修改后的运行结果如图 2-1:

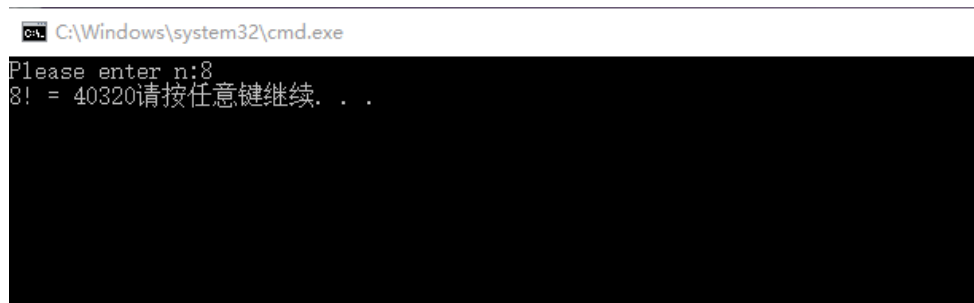


图 2-1

2.2.2 源程序修改替换题

- (1) 修改第 1 题, 分别用 `while` 和 `do-while` 语句替换 `for` 语句。
- (2) 修改第 1 题, 输入改为“整数 `S`”, 输出改为“满足 $n! \geq S$ 的最小整数 `n`”。例如输入整数 40310, 输出结果为 `n=8`。

解答:

- (1)

1) 使用 `while` 修改替换后的程序如下:

```
#include <stdio.h>

int main(void){
    int i = 1, n, s = 1;
    printf("Please enter n:");
    scanf("%d", &n);
    while(i <= n){
        s = s * i;
        i++;
    }
    printf("%d! = %d", n, s);
    return 0;
}
```

运行结果如图 2-2:



图 2-2

2) 使用 do-while 修改替换后的程序如下:

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

运行结果如图 2-3:



图 2-3

(2) 修改后的程序如下:

```
# include <stdio.h>
int main(void)
{
```

```

int i, s, a = 1;
scanf("%d", &s);
for(i = 1; ; i++)
{
    a *= i;
    if(a >= s)
        break;
}
printf("%d\n", i);
return 0;
}

```

运行结果如图 2-4:

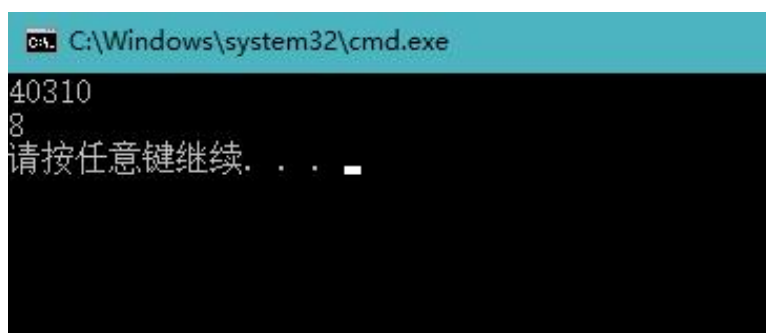


图 2-4

2.2.3 编程设计题

(1) 编写并上机调试运行能实现以下功能的程序。

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。编写一个程序, 输入工资金额, 输出应收取税金额度, 要求分别用 if 语句和 switch 语句来实现。

解答:

1) 源代码清单

a) 使用 if 语句实现

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

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

```

int salary;
scanf("%d", &salary);
if(salary < 1000)
    printf("0\n");
else if(salary < 2000)
    printf("%lf\n", salary * 0.05);
else if(salary < 3000)
    printf("%lf\n", salary * 0.1);
else if(salary < 4000)
    printf("%lf\n", salary * 0.15);
else if(salary < 5000)
    printf("%lf\n", salary * 0.20);
else
    printf("%lf\n", salary * 0.25);
return 0;
}

```

b) 使用 switch 语句实现

```

#include <stdio.h>
int main(void)
{
    int salary;
    scanf("%d", &salary);
    switch(salary/1000)
    {
        case 0:printf("0\n");
            break;
        case 1:printf("%lf\n", salary * 0.05);
            break;
        case 2:printf("%lf\n", salary * 0.1);
            break;
        case 3:printf("%lf\n", salary * 0.15);
            break;
        case 4:printf("%lf\n", salary * 0.20);
            break;
        default:printf("%lf\n", salary * 0.25);
    }
}

```

```

        break;
    }
    return 0;
}

```

2) 测试

a) 测试数据 2500

b) 对于测试数据的运行结果截图，如图 2-5



图 2-5

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

解答:

1) 源代码清单

```

#include <stdio.h>
#include <string.h>
#include <malloc.h>
#define N 65535
int main(void){
    char *str1;
    char *str;
    int i;
    str1 = (char *)malloc(sizeof(char) * N);
    scanf("%[^\\n]", str1);
    str = (char *)malloc(sizeof(char) * (strlen(str1) + 1));
    strcpy(str, str1);
    free(str1);
    for(i = 0;str[i] != '\\0'; i++){
        if(str[i] != ' ')

```

```

        putchar(str[i]);
    else{
        putchar(str[i]);
        for(; str[i] == ' '; i++);
        i--;
    }
}
putchar('\n');
return 0;
}

```

2) 测试

a) 测试数据 1432 321 13546 123421

b) 对于测试数据的运行结果截图，如图 2-6

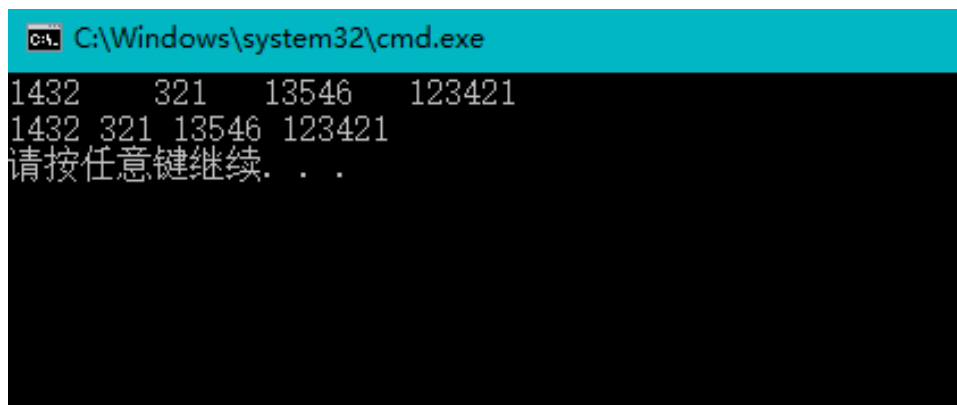


图 2-6

(3) 编写一个程序，打印如下的杨辉三角形。

1										/*第 0 行 */				
	1	1								/*第 1 行 */				
		1	2	1						/*第 2 行 */				
			1	3	3	1								
				1	4	6	4	1						
					1	5	10	10	5	1				
						1	6	15	20	15	6	1		
							1	7	21	35	35	21	7	1

```

1   8   28  56  70  56  28  8   1
1   9   36  84 126 126 84  36  9   1

```

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

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

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

本程序中为了打印出金字塔效果，要注意空格的数目。一位数之间是 3 个空格，两位数之间有 2 个空格，3 位数之间只有一个空格，程序编制过程中要注意区分。

解答：

1) 源代码清单

```

#include <stdio.h>

int C(int, int);

int main(void)
{
    int i;
    int j;
    int num;
    for(i = 1; i < 10; i++)
    {
        for(j = 0; j < (18 - 2 * i); j++)
            putchar(' ');
        for(j = 0; j < i; j++)
        {
            num = C(i - 1, j);
            printf("%d", num);
            if((num / 100) != 0)
                putchar(' ');
            else if((num / 10) != 0)

```



```

        printf(" ");
    else
        printf(" ");
    }
    putchar("\n");
}

return 0;
}

int C(int i, int j)
{
    if(j == 0)
        return 1;
    return (C(i, j - 1) * (i - j + 1) / j);
}

```

2) 测试

- a) 测试数据 无输入数据
- b) 对于测试数据的运行结果截图，如图 2-7:



图 2-7

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

解答:

1) 源代码清单

```
# include <stdio.h>
# include <string.h>
# include <malloc.h>
# define N 65535
int main(void)
{
    int i;
    char *num;
    char *str = (char *)malloc(sizeof(char) * N);
    scanf("%s", str);
    num = (char *)malloc(sizeof(char) * (strlen(str) + 1));
    strcpy(num, str);
    free(str);
    for(i = strlen(num) - 1; i >= 0; i--)
    {
        putchar(num[i]);
    }
    putchar('\n');
    return 0;
}
```

2) 测试

a) 测试数据 13654781523

b) 对于测试数据的运行结果截图，如图 2-8:

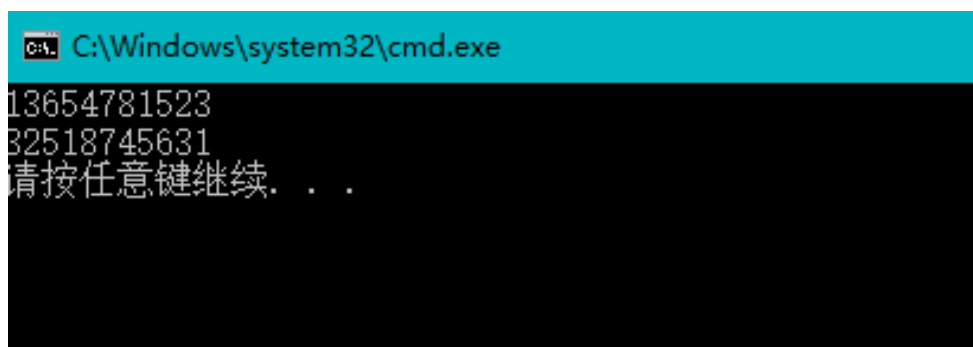


图 2-8

2.2.4 选做题

编写并上机调试运行能实现以下功能的程序。

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

牛顿迭代法求方程近似根的迭代公式为：

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

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

解答：

1) 源代码清单

```
#include <stdio.h>
#include <math.h>
double f(double);
double df(double);
int main(void){
    double x1, x2;
    scanf("%lf", &x2);
    while(1){
        x1 = x2;
        x2 = x1 - f(x1) / df(x1);
        if(fabs(x1 - x2) <= 1e-6)
            break;
    }
    printf("%lf\n", x2);
    return 0;
}
double f(double x){
    return (3 * x * x * x - 4 * x * x - 5 * x + 13);
}
double df(double x){
```

```
return (9 * x * x - 8 * x - 5);  
}
```

2) 测试

a) 测试数据 2

b) 对于测试数据的运行结果截图，如图 2-9:

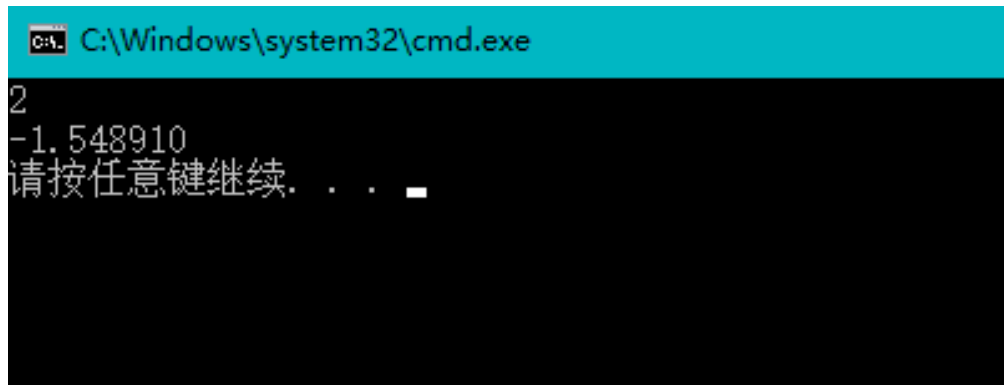


图 2-9

2.3 实验小结

本次实验时我发现写代码时要细心，注意调试。

3 函数与程序结构实验

3.1 实验目的

- (1) 熟悉和掌握函数的定义、声明；函数调用与参数传递方法；以及函数返回值类型的定义和返回值使用。
- (2) 熟悉和掌握不同存储类型变量的使用。
- (3) 熟悉多文件编译技术。

3.2 实验内容

3.2.1 源程序改错题

下面是计算 $s=1!+2!+3!+\dots+n!$ 的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

```
1  #include "stdio.h"
2  void main(void)
3  {
4      int k;
5      for(k=1;k<6;k++)
6          printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
7  }
8  long sum_fac(int n)
9  {
10     long s=0;
11     int i;
12     long fac;
13     for(i=1;i<=n;i++)
14         fac*=i;
15     s+=fac;
16     return s;
17 }
```

解答:

(1) 错误修改

- 1) 函数 `sum_fac` 缺少定义语句

正确形式为: 在 1、2 行之间添加 `long sum_fac(int n);`

- 2) 需要定义计算式中的 `n`

正确形式为: 第 4 行改为: `int k, n;`

第 4、5 行之间添加 `scanf("%d", &n);`

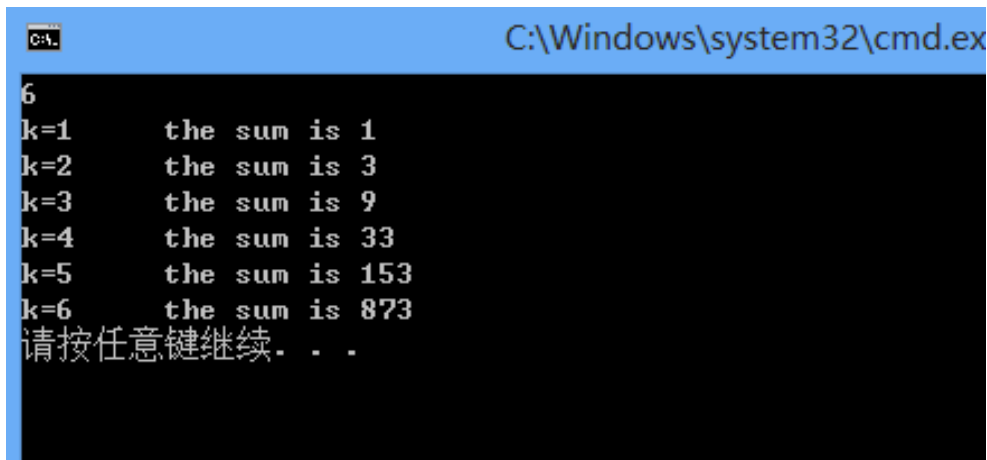
- 3) 第 12 行 `fac` 变量没有初始化

正确形式为: 第 12 行改为: `long fac = 1;`

- 4) 第 14、15 行应该都在循环体中

正确形式为: 14、15 行改为: `{fac*=i; s+=fac;}`

(2) 修改后运行结果如图 3-1



```
C:\Windows\system32\cmd.exe
6
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
请按任意键继续. . .
```

图 3-1

3.2.2 源程序修改替换题

- (1) 修改第 1 题中 `sum_fac` 函数, 使其计算量最小。

解答:

- 1) 替换后的程序如下

```
#include <stdio.h>
long sum_fac(int n);
```

```

void main(void)
{
    int k, n;
    scanf("%d", &n);
    for(k = 1; k <= n; k++)
        printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
}

long sum_fac(int n)
{
    static long s = 0, fac = 1;
    fac *= n;
    s += fac;
    return s;
}

```

2) 运行结果如图 3-2

```

C:\Windows\system...
6
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
请按任意键继续. . .

```

图 3-2

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

解答：

1) 替换后的程序如下

```

#include <stdio.h>
double sum_fac(int n);
void main(void)
{

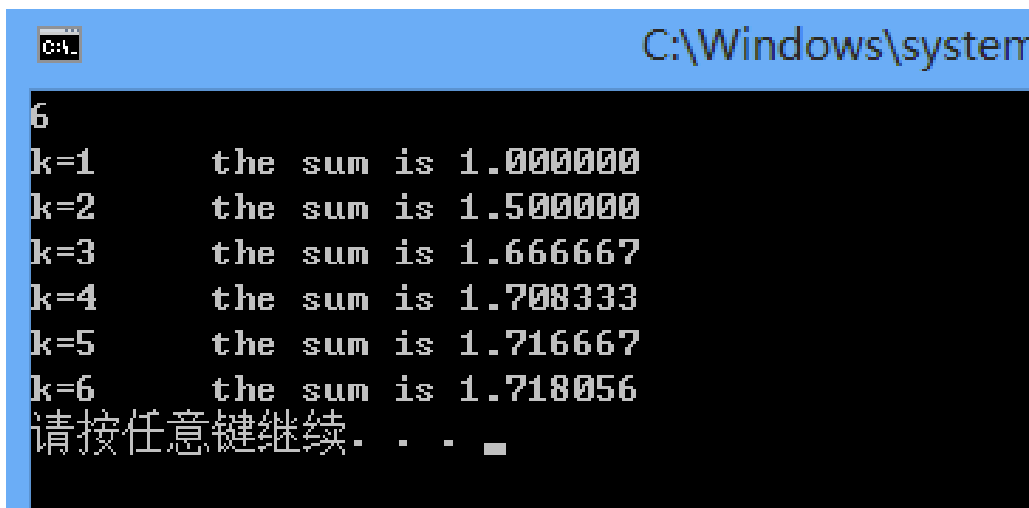
```

```

int k, n;
scanf("%d", &n);
for(k = 1; k <= n; k++)
    printf("k=%d\tthe sum is %lf\n",k,sum_fac(k));
}
double sum_fac(int n)
{
    static double s = 0;
    static long fac = 1;
    fac *= n;
    s += 1.0 / fac;
    return s;
}

```

2) 运行结果如图 3-3



```

C:\Windows\system
6
k=1    the sum is 1.000000
k=2    the sum is 1.500000
k=3    the sum is 1.666667
k=4    the sum is 1.708333
k=5    the sum is 1.716667
k=6    the sum is 1.718056
请按任意键继续. . .

```

图 3-3

3.2.3 跟踪调试题

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

- (1) 刚执行完 scanf("%d",&k);语句，p,i 值是多少？
- (2) 从 fibonacci 函数返回后光条停留在哪个语句上？
- (3) 进入 fibonacci 函数时，watch 窗口显示的是什么？
- (4) 当 i=3 时，从调用 fibonacci 函数到返回，n 值如何变化？

源程序

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

    long sum=0,*p=&sum;
    scanf("%d",&k);
    for(i=1;i<=k;i++){
        sum+=fabonacci(i);
        printf("i=%d\tthe sum is %ld\n",i,*p);
    }
}

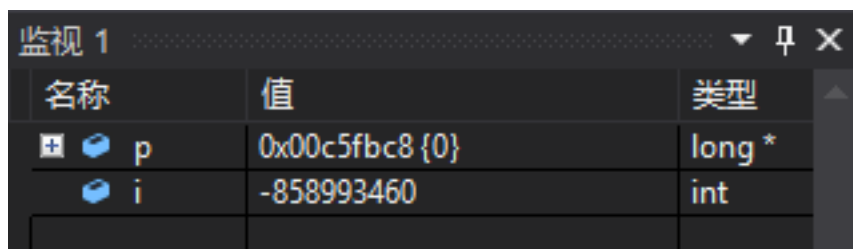
long fabonacci(int n)
{
    if(n==1 || n==2)
        return 1;
    else
        return fabonacci(n-1)+fabonacci(n-2);
}
```




其中，`long sum=0,*p=∑`声明 `p` 为长整型指针并用 `&sum` 取出 `sum` 的地址对 `p` 初始化。`*p` 表示引用 `p` 所指的变量（`*p` 即 `sum`）。

解答：

(1) `p` 的值为 `0x00c5fbc8`

`i` 的值为 `-858993460`



监视 1			
名称	值	类型	
  p	0x00c5fbc8 {0}	long *	
 i	-858993460	int	

(2) 从 `fabonacci` 函数返回后光条停留在语句 `sum += fabonacci(i);`上

(3) 进入 fabonacci 函数时 watch 窗口如图 3-4

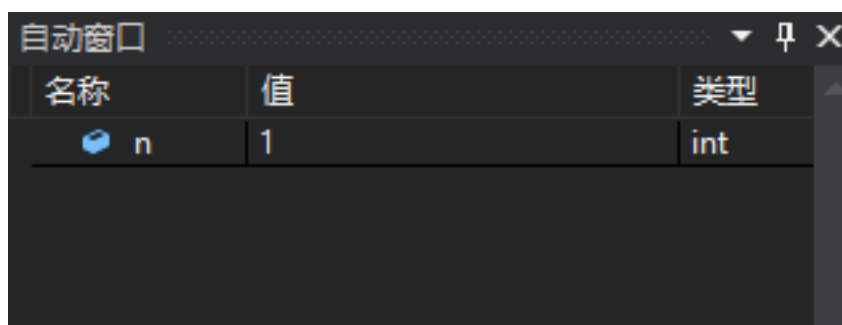


图 3-4

(4) n 由 3 变成 2 在变成 3

3.2.4 编程设计题

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

解答：

1) 源代码清单

```
#include <stdio.h>
int gcd(int, int);
int main(void)
{
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d\n", gcd(a, b));

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

```

        return gcd(b, a);
    }
}

```

2) 测试

a) 测试数据：13423 4325

b) 对测试数据的运行结果截图如图 3-5

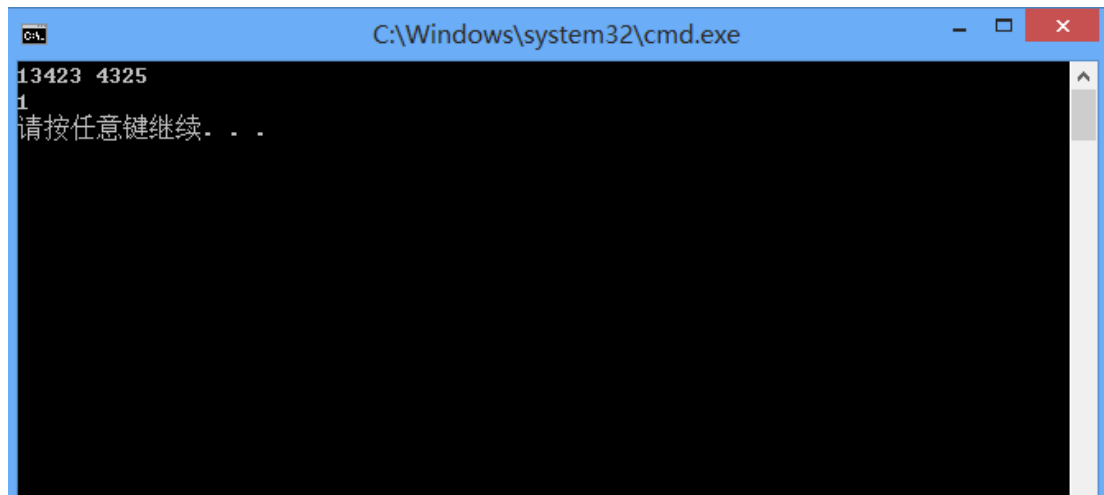


图 3-5

(2) 编程验证歌德巴赫猜想：一个大于等于 4 的偶数都是两个素数之和。

解答：

1) 源代码清单

```

#include <stdio.h>
void gold(int);
int isprime(int);
int main(void)
{
    int n = 4;
    for(; n <= 10000; n += 2)
    {
        gold(n);
    }

    return 0;
}

```

```

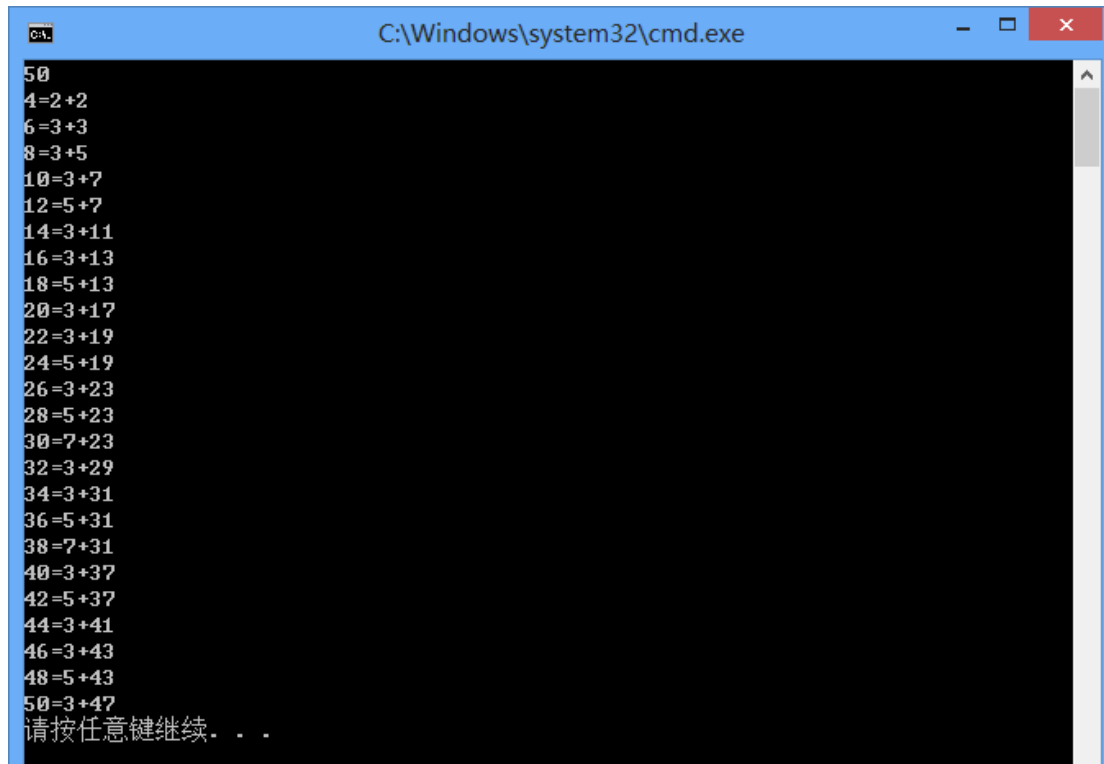
int isprime(int n)
{
    int i;
    for(i = 2; i <= n / 2; i++)
    {
        if(n % i == 0)
            return 0; //不是素数
    }
    return 1; //是素数
}

void gold(int n)
{
    int prime = 2;
    for( ; ; prime++)
    {
        if(isprime(prime) == 0)
            continue;
        else
        {
            if(isprime(n - prime) == 1)
            {
                printf("%d=%d+%d\n", n, prime, n - prime);
                break;
            }
        }
    }
}

```

2) 测试

- a) 测试数据 50
- b) 对于测试数据的运行截图如图 3-6

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. It displays a list of even numbers from 4 to 50, each followed by an equals sign and a sum of two prime numbers. The pairs of primes are: (2,2) for 4, (3,3) for 6, (3,5) for 8, (3,7) for 10, (5,7) for 12, (3,11) for 14, (3,13) for 16, (5,13) for 18, (3,17) for 20, (3,19) for 22, (5,19) for 24, (3,23) for 26, (5,23) for 28, (7,23) for 30, (3,29) for 32, (3,31) for 34, (5,31) for 36, (7,31) for 38, (3,37) for 40, (5,37) for 42, (3,41) for 44, (3,43) for 46, (5,43) for 48, and (3,47) for 50. At the bottom, it says "请按任意键继续..." (Press any key to continue...).

```
C:\Windows\system32\cmd.exe
50
4=2+2
6=3+3
8=3+5
10=3+7
12=5+7
14=3+11
16=3+13
18=5+13
20=3+17
22=3+19
24=5+19
26=3+23
28=5+23
30=7+23
32=3+29
34=3+31
36=5+31
38=7+31
40=3+37
42=5+37
44=3+41
46=3+43
48=5+43
50=3+47
请按任意键继续...
```

图 3-6

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

GOLDBACH'S CONJECTURE:

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

10=3+7

12=5+7

.....

20=3+17

解答：

1) 源代码清单：

```
# include <stdio.h>
# define BEGIN 10
# define END 20
void gold(int);
int isprime(int);
int main(void)
{
```

```

int begin;
if(BEGIN % 2 != 0)
    begin = BEGIN + 1;
else
    begin = BEGIN;
for( ; begin <= END; begin += 2)
{
    gold(begin);
}
return 0;
}
int isprime(int n)
{
    int i;
    for(i = 2; i <= n / 2; i++)
    {
        if(n % i == 0)
            return 0;//不是素数
    }
    return 1;//是素数
}
void gold(int n)
{
    int prime = 2;
    printf("GOLDBACH'S CONJECTURE:\nEvery even number n>=4 is the sum of
two primes.\n");
    for( ; ; prime++)
    {
        if(isprime(prime) == 0)
            continue;
        else
        {
            if(isprime(n - prime) == 1)
            {
                printf("%d=%d+%d\n", n, prime, n - prime);
            }
        }
    }
}

```

```

        break;
    }
}
}
}

```

2) 测试

a) 测试数据 无输入

b) 对测试数据的运行结果截图如图 3-7:

```

C:\Windows\system32\cmd.exe
Every even number n>=4 is the sum of two primes.
10=3+7
12=5+7
14=3+11
16=3+13
18=5+13
20=3+17
22=3+19
24=5+19
26=3+23
28=5+23
30=7+23
32=3+29
34=3+31
36=5+31
38=7+31
40=3+37
42=5+37
44=3+41
46=3+43
48=5+43
50=3+47
52=5+47
54=7+47
56=3+53
58=5+53
60=7+53
62=3+59
64=3+61
66=5+61
68=7+61
70=3+67
72=5+67
74=3+71
76=3+73
78=5+73
80=7+73
82=3+79
84=5+79
86=3+83
88=5+83
90=7+83
92=3+89
94=5+89
96=7+89
98=19+79
100=3+97
请按任意键继续...

```

图 3-7

3.2.5 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, 并编译和链接。然后运行生成的可执行文件。

源文件 file1.c 的内容为:

```
#include "file.h"

int x,y;          /* 外部变量的定义性说明 */
char ch;          /* 外部变量的定义性说明 */
int main(void)
{
    x=10;
    y=20;
    ch=getchar();
    printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);
    func1();
    return 0;
}
```

源文件 file2.c 的内容为:

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

解答:

编译结果如图 3-8

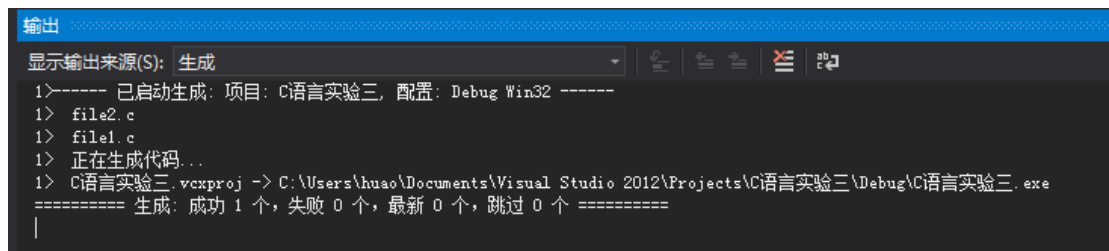


图 3-8

运行结果如图 3-9

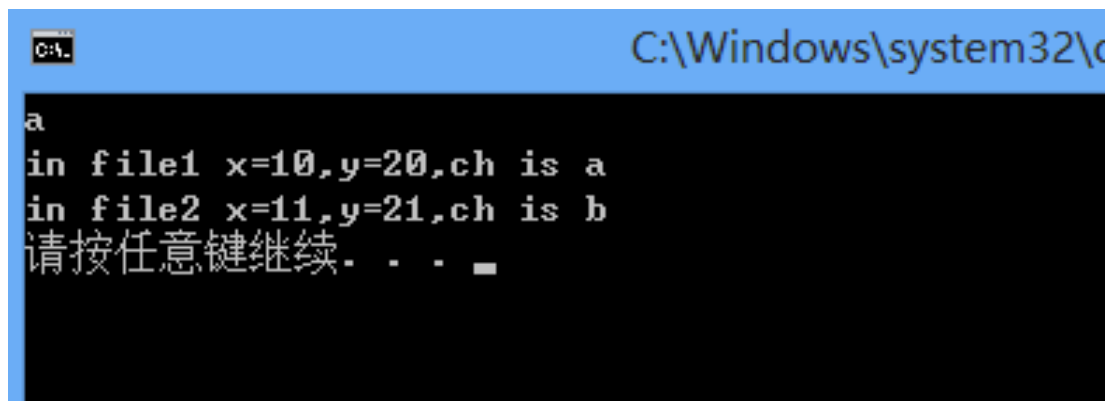


图 3-9

3.3 实验小结

注意在使用函数是形参和实参的区别。

4 编译预处理实验

4.1 实验目的

- (1) 掌握文件包含、宏定义、条件编译、assert 宏的使用；
- (2) 练习带参数的宏定义、条件编译的使用；
- (3) 练习 assert 宏的使用；
- (4) 使用 Turbo C 2.0 集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

4.2 实验题目及要求

4.2.1 源程序改错题

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语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

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

解答：

(1) 错误修改

- 1) 第 4 行宏定义不能实现两个变量的值的交换

正确形式为: `#define SWAP(a,b) (a=a+b,b=a-b, a=a-b)`

- 2) 第 5 行 `main` 函数后面要加括号

正确形式为: `void main(void)`

- 3) 第 9 行变量 `a` 没有定义, `%d` 和 `%d` 之间不需要加逗号

正确形式为: 将第 7 行改为: `int a, b;`

将第 9 行改为: `scanf("%d %d", &a,&b);`

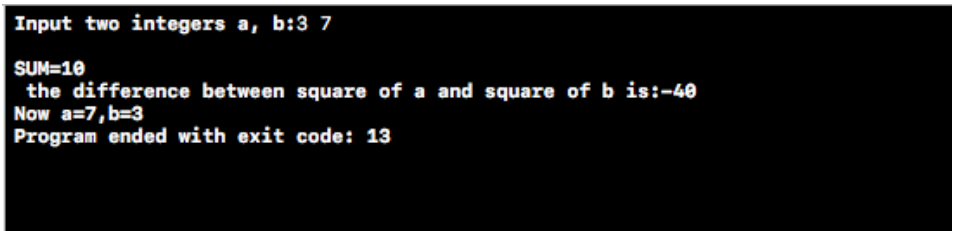
- 4) 第 10 行平方差计算错误

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

- 5) 第 12 行 `printf` 的 `p` 不能大写

正确形式为: `printf("\nNow a=%d,b=%d\n",a,b);`

(2) 修改后运行结果如图 4-1



```
Input two integers a, b:3 7
SUM=10
the difference between square of a and square of b is:-40
Now a=7,b=3
Program ended with exit code: 13
```

图 4-1

4.2.2 源程序修改替换题

下面是用函数实现求三个数中最大数、计算两数之和的程序,在这个源程序中存在若干语法和逻辑错误。

要求: (1) 对这个例子程序进行调试修改,使之能够正确完成指定任务;

(2) 用带参数的宏替换函数 `max`,来实现求最大数的功能。

```
1 void main(void)
2 {
3     int a, b, c;
4     float d, e;
```

```

5    printf("Enter three integers:");
6    scanf("%d,%d,%d",&a,&b,&c);
7    printf("\nthe maximum of them is %d\n",max(a,b,c));
8    printf("Enter two floating point numbers:");
9    scanf("%f,%f",&d,&e);
10   printf("\nthe sum of them is  %f\n",sum(d,e));
11 }

```

```

12 int max(int x, int y, int z)
13 {
14     int t;
15     if (x>y)
16         t=x;
17     else
18         t=y;
19     if (t<z)
20         t=z;
21     return t;
22 }

```

```

23 float sum(float x, float y)
24 {
25     return x+y;
26 }

```

解答：

(1) 错误修改：

1) 没有引用头文件、没有写函数声明

正确形式为：在第 1 行前添加 `#include <stdio.h>`

`float sum(float, float);`

`int max(int, int, int);`

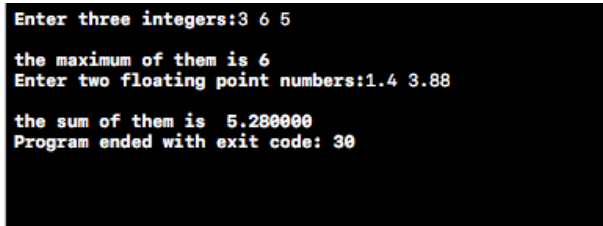
2) 第 6 行 `%d` 之间不加逗号

正确形式为： `scanf("%d %d %d",&a,&b,&c);`

3) 第 9 行%f 之间不加逗号

正确形式为: `scanf("%f %f",&d,&e);`

修改后程序运行结果如图 4-2



```
Enter three integers:3 6 5
the maximum of them is 6
Enter two floating point numbers:1.4 3.88
the sum of them is 5.280000
Program ended with exit code: 30
```

图 4-2

(2) 修改后的源代码如下

```
# include <stdio.h>
# define max(a, b, c)  ((a = ((a > b)? a : b), a = (a > c)? a : c))
float sum(float, float);
int main(void)
{
    int a, b, c;
    float d, e;
    printf("Enter three integers:");
    scanf("%d %d %d",&a,&b,&c);
    printf("\nthe maximum of them is %d\n", max(a,b,c));

    printf("Enter two floating point numbers:");
    scanf("%f %f",&d,&e);
    printf("\nthe sum of them is  %f\n",sum(d,e));
    return 0;
}
float sum(float x, float y)
{
    return x+y;
}
```

运行结果如图 4-3

```
Enter three integers:13 46 57
the maximum of them is 57
Enter two floating point numbers:2.4 5.44
the sum of them is 7.840000
Program ended with exit code: 0
```

图 4-3

4.2.3 跟踪调试题

下面程序利用 R 计算圆的面积 s，以及面积 s 的整数部分。现要求：

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

```
1  #define R
2  void main(void)
3  {
4      float r, s;
5      int s_integer=0;
6      printf("input a number: ");
7      scanf("%f",&r);
8      #ifdef R
9          s=3.14159*r*r;
10         printf("area of round is: %f\n",s);
11         s_integer= integer_fraction(s);
12         printf("the integer fraction of area is %d\n", s_integer);
13         assert((s-s_integer)<1.0);
14     #endif
15 }

int integer_fraction(float x)
{
```

```

int i=x;
return i;
}

```

解答：

(1) 修改程序：

1) 没有引用头文件

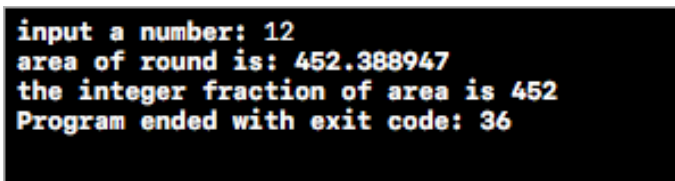
正确形式为：在第 1 行前面添加：# include <stdio.h>

include <assert.h>

2) 缺少函数声明

正确形式为：在第 1 行和第 2 行间添加：int integer_fraction(float);

修改后程序运行结果如图 4-4



```

input a number: 12
area of round is: 452.388947
the integer fraction of area is 452
Program ended with exit code: 36

```

图 4-4

(2) 进入函数时 x 的值为 452.388947，返回 main 函数时 i 的值为 452

4.2.4 编程设计题

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

解答：

1) 源代码清单：

```

# include <stdio.h>
# include <math.h>
# define S(a, b, c) ((a + b + c)/2)
# define AREA(s, a, b, c) (sqrt(s * (s - a) * (s - b) * (s - c)))
int main(void)
{

```

```

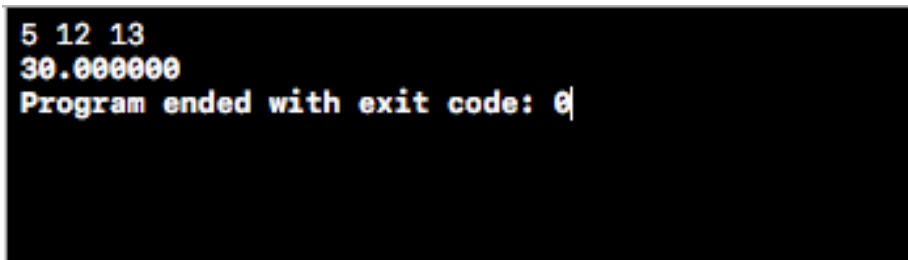
double a, b, c, s;
scanf("%lf%lf%lf", &a, &b, &c);
s = S(a, b, c);
printf("%lf\n", AREA(s, a, b, c));

return 0;
}

```

2) 测试

- a) 测试数据 5 12 13
- b) 对于测试数据的运行结果如图 4-5



```

5 12 13
30.000000
Program ended with exit code: 0

```

图 4-5

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

解答：

1) 源代码清单：

```

#include <stdio.h>
#include <string.h>
#define CHANGE 0
int main(void)
{
    char str[100];
    int len, i;
    scanf("%s", str);
    len = strlen(str);
    #if CHANGE
        for(i = 0; i < len; i++)

```



```

    {
        if(str[i] >= 'A' && str[i] <= 'Z')
            str[i] = str[i] - 'A' + 'a';
        else if(str[i] >= 'a' && str[i] <= 'z')
            str[i] = str[i] - 'a' + 'A';
    }
#endif

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

return 0;
}

```

2) 测试

- a) 测试数据 abgvfGFGHVJ4143324
- b) 对于测试数据的运行结果如图 4-6

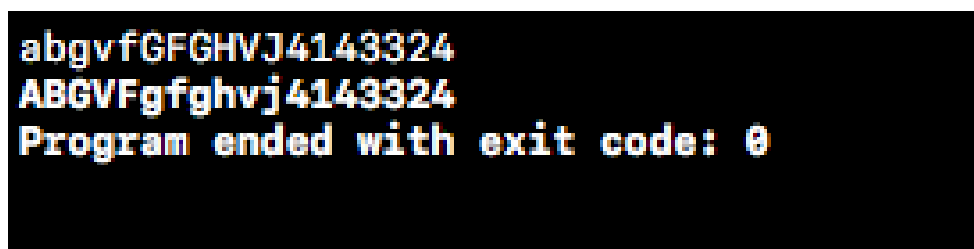


图 4-6

4.3 实验小结

熟悉各种编译预处理的指令，注意在宏定义时括号的使用。

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

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

源程序

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

```

16     for (i = 0; i < n - 1; i++)
17         for (j = 0; j < n - i - 1; j++)
18             if(b[j] < b[j+1])
19                 t = b[j], b[j] = b[j+1], b[j+1] = t;
20 }

```

解答：

(1) 错误修改

1) 第 7 行 sort 函数调用时第一个参数传递错误

正确形式为：sort(a, 10);

2) 第 18 行存在逻辑错误导致结果为降序排列

正确形式为：if(b[j] > b[j + 1])

(2) 修改后运行结果截图如图 5-1

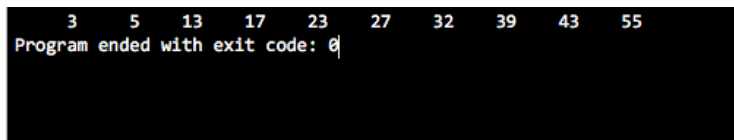


图 5-1

5.2.2 源程序完善、修改、替换

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

源程序：

```

#include<stdio.h>
#define M 10
#define N 3
int main(void)
{
    int a[M], b[M]; /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int i, j, k;
    for(i = 0; i < M; i++)                /* 对圈中人按顺序编号 1—M */
        a[i] = i + 1;
    for(i = M, j = 0; i > 1; i--){

```

```

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

```

(2) 上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。因此，请采用做标记的办法修改（1）中的程序，并使修改后的程序与（1）中的程序具有相同的功能。

解答：

1) 修改后的程序如下

```

#include<stdio.h>
#define M 10
#define N 3
int main(void)
{
    int a[M], b[M]; /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int i, j, k;
    for(i = 0; i < M; i++) /* 对圈中人按顺序编号 1—M */
    {
        a[i] = i + 1;
        b[i] = 0;
    }
}

```

```

    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(b[j])
            {
                k--;
                j++;
            }
            else if(++j > M - 1) j = 0; /* 最后一个人报数后第一个人接着报，形成一
            个圈 */
        }
        if(j)
        {
            b[j - 1] = 1;
            printf("%6d", a[j - 1]);
        }
        else
        {
            b[M - 1] = 1;
            printf("%6d", a[M - 1]);
        }
    }
    for(i = 0; i < M; i++)
    {
        if(b[i] == 0)
        {
            printf("%6d\n", a[i]);
            break;
        }
    }
    return 0;
}

```

2) 修改后运行结果截图如图 5-2

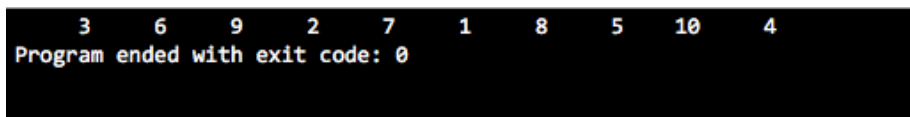


图 5-2

5.2.3 跟踪调试源程序

在下面所给的源程序中，函数 `strncat(s,t,n)` 本来应该将字符数组 `t` 的前 `n` 个字符连接到字符数组 `s` 中字符串的尾部。但函数 `strncat` 在定义时代码有误，不能实现上述功能。请按下面的要求进行操作，并回答问题和排除错误。

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

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

源程序：

```
1  #include<stdio.h>
2  void strncat(char [],char [],int);
3  int main(void)
4  {
5      char      a[50]="The      adopted      symbol      is      ",
6      b[27]="abcdefghijklmnopqrstuvwxyz";
7      strncat(a, b, 4);
8      printf("%s\n",a);
9      return 0;
10 }
11 void strncat(char s[],char t[], int n)
12 {
13     int I = 0, j;
14     while(s[i++]) ;
15     for(j = 0; j < n && t[j] != '\0')
16         s[i++] = t[j++];
17     s[i] = '\0';
```

17 }

解答：

(1) 当光条落在 for 语句所在行时，i = 23

当光条落在 strcat 函数块结束标记所在行时，

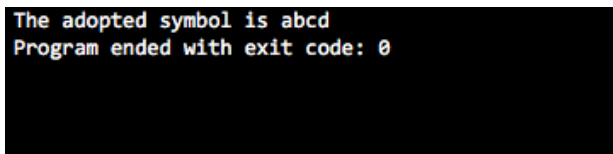
s = "The adopted symbol is " 0x00007ffefbfff5c0

t = "abcdefghijklmnopqrstuvwxyz" 0x00007ffefbfff5a0

(2) 函数出错的原因：没有使用字符串 t 中的元素覆盖掉字符串 s 的结束符\0，
导致字符串 s 始终在原来的位置遇到结束符结束

正确形式为：在 13 行和 14 行之间添加一行 i--;

修改后的运行结果如图 5-3



```
The adopted symbol is abcd
Program ended with exit code: 0
```

图 5-3

5.2.4 程序设计

(1) 编写并上机调试运行能实现以下功能的程序。

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

解答：

1) 源代码清单

```
#include <stdio.h>
int main(void)
{
    int mat1[3][4];
    int mat2[4][3];
    int i, j;
    for(i = 0; i < 3; i++)
        for(j = 0; j < 4; j++)
            scanf("%d", &mat1[i][j]);
    for(i = 0; i < 3; i++)
        for(j = 0; j < 4; j++)
```

```

        mat2[j][i] = mat1[i][j];
    for(i = 0; i < 3; i++)
    {
        for(j = 0; j < 4; j++)
            printf("%6d", mat1[i][j]);
        putchar('\n');
    }
    putchar('\n');
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 3; j++)
        {
            printf("%6d", mat2[i][j]);
        }
        putchar('\n');
    }
    return 0;
}

```

2) 测试

a) 测试数据

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

b) 对测试数据的运行结果截图如图 5-4

```

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

1 5 9
2 6 10
3 7 11
4 8 12
Program ended with exit code: 0

```

图 5-4

- (2) 编写一个程序，其功能要求是：输入一个整数，将它在内存中二进制表示的每一位转换成为对应的数字字符，存放到一个字符数组中，然后输出该整数的二进制表示。

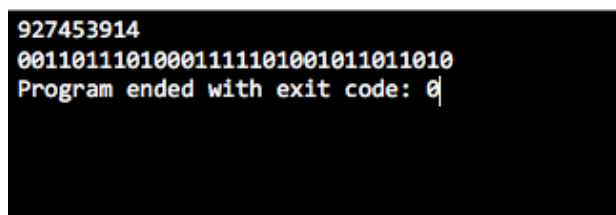
解答：

1) 源代码清单

```
#include <stdio.h>
int main(void)
{
    int n;
    char num[32];
    int i;
    unsigned int mask = 1;
    scanf("%d", &n);
    mask <= 31;
    for(i = 0; i < 32; i++)
    {
        num[i] = (mask & n) >> (31 - i);
        mask >>= 1;
    }
    for(i = 0; i < 32; i++)
        printf("%d", num[i]);
    putchar('\n');
    return 0;
}
```

2) 测试

- a) 测试数据 927453914
- b) 对测试数据的运行结果截图 s 如图 5-5



```
927453914
00110111010001111101001011011010
Program ended with exit code: 0
```

图 5-5

- (3) 编写一个程序, 其功能要求是: 输入 n 个学生的姓名和 C 语言课程的成绩, 将成绩按从高到低的次序排序, 姓名同时作相应调整, 输出排序后学生的姓名和 C 语言课程的成绩。然后, 输入一个 C 语言课程成绩值, 用二分查找进行搜索。如果查找到有该成绩, 输出该成绩同学的姓名和 C 语言课程的成绩; 否则输出提示 “not found!”。

解答:

1) 源代码清单

```
#include <stdio.h>
#include <stdlib.h>
struct student
{
    char name[20];
    int score;
};
int main(void)
{
    int n, i, j, score, begin, end;
    struct student *stu;
    struct student t;
    scanf("%d", &n);
    stu = (struct student *)malloc(sizeof(struct student) * n);
    for(i = 0; i < n; i++)
        scanf("%s %d", stu[i].name, &stu[i].score);
    for(i = 0; i < n; i++)
    {
        for(j = i; j < n - 1; j++)
        {
            if(stu[j].score < stu[j + 1].score)
            {
                t = stu[j];
                stu[j] = stu[j + 1];
                stu[j + 1] = t;
            }
        }
    }
}
```

```

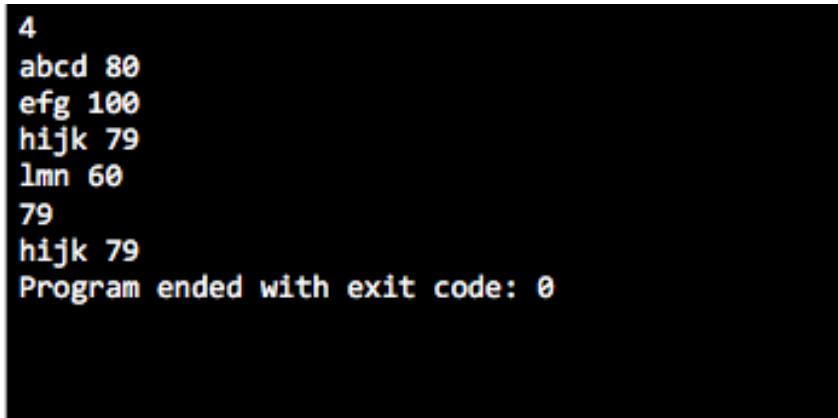
}
scanf("%d", &score);
begin = 0;
end = n - 1;
while(1){
    if(score < stu[(begin + end) / 2].score)
        begin = (begin + end) / 2;
    else if(score > stu[(begin + end) / 2].score)
        end = (begin + end) / 2;
    else
    {
        printf("%s %d\n", stu[(begin + end) / 2].name, stu[(begin + end) / 2].score);
        break;
    }
    if(end - begin == 1)
    {
        if(score == stu[begin].score)
            printf("%s %d\n", stu[begin].name, stu[begin].score);
        else if(score == stu[end].score)
            printf("%s %d\n", stu[end].name, stu[end].score);
        else
            printf("not found!\n");
        break;
    }
}
return 0;
}

```

2) 测试

a) 测试数据	4
	abcd 80
	efg 100
	hijk 79
	lmn 60
	79

b) 对测试数据的运行结果截图如图 5-6



```
4
abcd 80
efg 100
hijk 79
lmn 60
79
hijk 79
Program ended with exit code: 0
```

图 5-6

5.2.5 选做题程序设计

编写并上机调试运行能实现以下功能的函数和程序。

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

解答:

1) 源代码清单

```
void strnins(char s[], char t[], int n)
{
    int len = strlen(t);
    int len2 = strlen(s);
    int i;
    for(i = len2 - n; i >= 0; i--)
        s[i + len + n] = s[i + n];
    for(i = 0; i < len; i++)
        s[n + i] = t[i];
}
```

2) 测试

a) 测试数据 `s[30] = "abcdxyz"`
 `t[10] = "efghijkl"`

b) 对于测试数据的运行截图如图 5-7

```
abcdefghijklmnopqrstuvwxyz
Program ended with exit code: 0
```

图 5-7

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

解答：

1) 源代码清单

```
#include <stdio.h>

int sum(int queen[8][8], int x, int y);    //计算 第 x 行 或者 第 j 列 或者 (x, y)
                                           所在的两个斜线上 所有数字之和 从而判断是否成立
//用 1 代表有皇后 0 代表没有皇后 那么当 sum 中计算的四个值均为 1 时，
//表明该状态符合要求
//当判断时若 某行 或 某列 出现非 1 的和时 即可判定该情况不成立
int isqueen(int queen[8][8]);              //isqueen()函数用于判断当前传入的
queen 数组是否满足八皇后的条件
//若满足则返回 1 不满足则返回 0
void show(int queen[8][8]);                //show()函数用于输出当前传入的
queen 数组的内容

int main(void)
{
    int queen[8][8];
    int i, j;
    int a, b, c, d, e, f, g, h;           //这八个变量用于控制循环 完成遍历的过程
    int num = 0;                           //sum 用于存放存在的种数

    for(i = 0; i < 8; i++)
        for(j = 0; j < 8; j++)
            queen[i][j] = 0;
    //两个 for 循环套嵌将 queen 数组的所有元素全部赋值为 0 //0 为初始状态
```

```

for(a = 0; a < 8; a++)
{
    queen[0][a] = 1;
    for(b = 0; b < 8; b++)
    {
        if(b == a)
            continue;
        queen[1][b] = 1;
        for(c = 0; c < 8; c++)
        {
            if(c == b || c == a)
                continue;
            queen[2][c] = 1;
            for(d = 0; d < 8; d++)
            {
                if(d == c || d == b || d == a)
                    continue;
                queen[3][d] = 1;
                for(e = 0; e < 8; e++)
                {
                    if(e == d || e == c || e == b || e == a)
                        continue;
                    queen[4][e] = 1;
                    for(f = 0; f < 8; f++)
                    {
                        if(f == e || f == d || f == c || f == b || f == a)
                            continue;
                        queen[5][f] = 1;
                        for(g = 0; g < 8; g++)
                        {
                            if(g == f || g == e || g == d || g == c || g == b || g == a)
                                continue;
                            queen[6][g] = 1;
                            for(h = 0; h < 8; h++)
                            {

```

```

        if(h == g || h == f || h == e || h == d || h == c || h == b
|| h == a)

            continue;
        queen[7][h] = 1;
        //在这个地方会生成每一种可能出现的情况    用
函数来实现对该情况是否成立做判断
        if(isqueen(queen))        //isqueen(queen) 为 1 时
该情况满足条件

        {
            num++;
            show(queen);        //输出 queen 数组的内容
            putchar('\n');
        }
        queen[7][h] = 0;
    }
    queen[6][g] = 0;
}
queen[5][f] = 0;
}
queen[4][e] = 0;
}
queen[3][d] = 0;
}
queen[2][c] = 0;
}
queen[1][b] = 0;
}
queen[0][a] = 0;
}
printf("\n%d\n", num);

return 0;
}
void show(int queen[8][8])        //输出传入的 queen 数组
{

```

```

int i, j;
for(i = 0; i < 8; i++)
{
    for(j = 0; j < 8; j++)
        printf("%3d", queen[i][j]);
    putchar('\n');
}
}

int sum(int queen[8][8], int x, int y)                //返回 0 表明在该点处的某个方
向不成立
{
    int i;
    int s = 0;
    //对行进行计算
    for(i = 0; i < 8; i++)
        s += queen[x][i];
    //此时 s 为 第 x 行 的 8 个元素之和
    if(s != 1)
        return 0;                                //返回 0 表示不满足题意
    //对列进行计算
    s = 0;
    for(i = 0; i < 8; i++)
        s += queen[i][y];
    //此时 s 为 第 y 列 的 8 个元素之和
    if(s != 1)
        return 0;
    //对斜线进行计算 从左下到右上
    s = 0;
    for(i = 0; (x - i >= 0) && (y - i >= 0); i++)
        s += queen[x - i][y - i];
    for(i = 1; (x + i < 8) && (y + i < 8); i++)
        s += queen[x + i][y + i];
    if(s > 1)
        return 0;
    //从左上到右下

```



```

    s = 0;
    for(i = 0; (x - i >= 0) && (y + i < 8); i++)
        s += queen[x - i][y + i];
    for(i = 1; (x + i < 8) && (y - i >= 0); i++)
        s += queen[x + i][y - i];
    if(s > 1)
        return 0;
    return 1;    //在此处返回 1 的是全部满足条件的情况
}
int isqueen(int queen[8][8])
{
    int i;
    for(i = 0; i < 8; i++)
    {
        if(!sum(queen, 0, i))    //成立表明和该点相关的位置不满足
            return 0;
        if(!sum(queen, i, 0))
            return 0;
        if(!sum(queen, i, 7))
            return 0;
        if(!sum(queen, 7, i))
            return 0;
    }
    return 1;
}

```

2) 测试

- a) 测试数据
 - b) 对测试数据的运行结果
- 共有 92 中可行情况

5.3 实验小结

综合考虑各种问题，在数组声明时创建足够并且合适大小的数组。

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

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

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

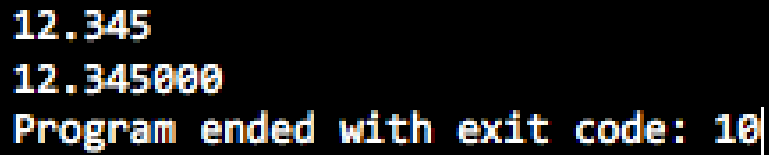
解答：

(1) 错误修改

1) 第 4 行指针 p 声明后没有初始化

正确形式为：在 3、4 行之间添加 float f;将第四行改为：float *p = &f;

(2) 修改后运行结果如图 6-1



```
12.345
12.345000
Program ended with exit code: 10
```

图 6-1

6.2.2 源程序完善、修改、替换题

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

```
#include "stdio.h"
#include "string.h"
void main(void){
    char>(*p)(char *, const char *);
    char a[80],b[80],c[160],*result=c;
    int choice,i;
    do{
        printf("\t\t1 copy string.\n");
        printf("\t\t2 connect string.\n");
        printf("\t\t3 exit.\n");
        printf("\t\tinput a number (1-3) please!\n");
        scanf("%d",&choice);
    }while(choice<1 || choice>5);
    switch(choice){
    case 1:
        p=strcpy;
        break;
    case 2:
        p=strcat;
        break;
    case 3:
        goto down;
    }
    getchar();
}
```

```

    printf("input the first string please!\n");
    i=0;
for(; (a[i] = getchar()) != '\n'; i++);a[i] = '\0';
    printf("input the second string please!\n");
    i=0;
for(; (b[i] = getchar()) != '\n'; i++);b[i] = '\0';
    result=    (*p)(a,b);
    printf("the result is %s\n",result);
down:
    ;
}

```

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

```

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

```

2 (输入)

```

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

```

解答:

- 1) 填空代码见代码部分
- 2) 输入样例后输出结果截图如图 6-2

```

1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!
2
input the first string please!
the more you learn,
input the second string please!
the more you get.
the result is the more you learn,the more you get.
Program ended with exit code: 177

```

图 6-2

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  char *strcpy(char *s,char *t)
9  {
10     while(*s++=*t++)
11         ;
12     return (s);
13 }
```

解答：

(1) 进入 strcpy 时 watch 窗口中 s 的值为：s ""0x00007ffeefbfff680

返回 main 时 watch 窗口中 s 的值为：

s char * 0x7ffeefbfff69d 0x00007ffeefbfff69d

(2) 错误修改：

a) 第 5 行声明的 a 字符数组长度不够

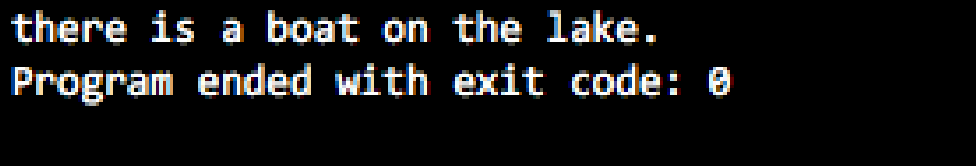
正确形式为：将第 5 行改为：char a[60],b[60]="there is a boat on the lake.";

b) 函数 strcpy 中 s 的值被改变了，返回的结果不符合要求

正确形式为：在第 9 行和第 10 行之间添加：char *p = s;

将第 12 行改为：return (p);

错误修改后运行结果截图如图 6-3



```
there is a boat on the lake.  
Program ended with exit code: 0
```

图 6-3

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 位并以数字字符的形式进行显示。

解答：

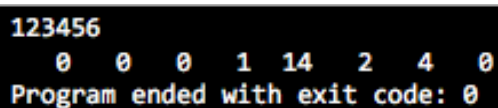
1) 源代码清单

```
#include <stdio.h>
int main(void)
{
    int n, i;
    unsigned int mask = 15 << 28;
    scanf("%d", &n);
    for(i = 7; i >= 0; i--)
    {
        printf("%4d", (mask & n) >> (i * 4));
        mask >>= 4;
    }
    printf("\n");
    return 0;
}
```

2) 测试

a) 测试数据 123456 (0000 0000 0000 0001 1110 0010 0100 0000)

b) 对测试数据的运行结果截图如图 6-4



```
123456
0 0 0 1 14 2 4 0
Program ended with exit code: 0
```

图 6-4

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

解答：

1) 源代码清单

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void out(char *);
int main(void)
{
    int n, i;
    char **str;
    scanf("%d", &n);
    getchar();
    str = (char **)malloc(sizeof(char *) * n);
    for(i = 0; i < n; i++)
        str[i] = (char *)malloc(sizeof(char) * 81);
    for(i = 0; i < n; i++)
        gets(str[i]);
    for(i = 0; i < n; i++)
        out(str[i]);

    return 0;
}
void out(char *str)
{
    if(str[0] == 0)
        return;
    int len = strlen(str);
    int i;
    for(i = 0; i < len; i++)
    {
        if(str[i] != ' ')
```

```

        putchar(str[i]);
    else
    {
        putchar(' ');
        while(str[i] == ' ')
            i++;
        i--;
    }
}
putchar('\n');
}

```

2) 测试

a) 测试数据

3

aaaaaa bbbbbb ccccc ddddd

hhhhh aaaaa xxxxxxxx

b) 对测试数据的运行结果截图如图 6-5

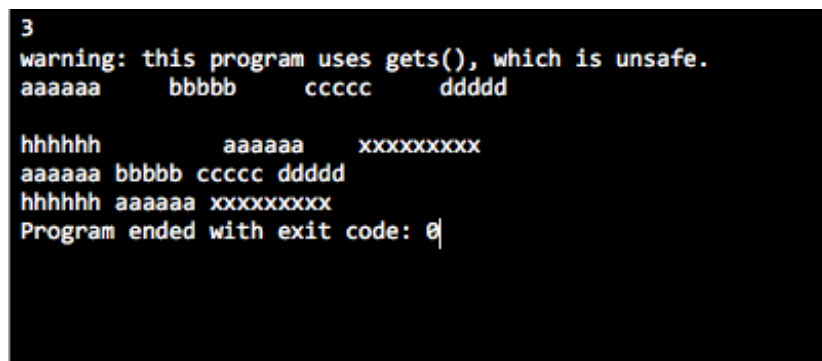


图 6-5

(3) 编写一个程序，输入 n 个整数，排序后输出。排序的原则由命令行可选参数 $-d$ 决定，并且有参数 $-d$ 时按递减顺序排序，否则按递增顺序排序。要求将排序算法定义成函数，利用指向函数的指针使该函数实现递增或递减排序。(main 函数参数处理见下面的第 (三) 大点)

解答：

1) 源代码清单

```

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

```



```

# define SWAP(a, b) (a = a + b, b = a - b, a = a - b)
void sort(int *, int, int(int, int));
int dz(int a, int b);
int dj(int a, int b);
int main(int argc, const char * argv[])
{
    int n, i;
    int *num;
    scanf("%d", &n);
    num = (int *)malloc(sizeof(int) * n);
    for(i = 0; i < n; i++)
    {
        scanf("%d", &num[i]);
    }
    if(argc == 1)
        sort(num, n, dz);
    //递减输出
    else
        sort(num, n, dj);
    //递增输出
    for(i = 0; i < n - 1; i++)
        printf("%d  ", num[i]);
    printf("%d\n", num[i]);
    return 0;
}
int dz(int a, int b)
{
    return (a - b);
}
int dj(int a, int b)
{
    return b - a;
}
void sort(int *num, int len, int compare(int, int))
{

```

```

int i, j;
for(i = 0; i < len; i++)
{
    for(j = i; j < len; j++)
    {
        if(compare(num[i], num[j]) < 0)
            SWAP(num[i], num[j]);
    }
}
}

```

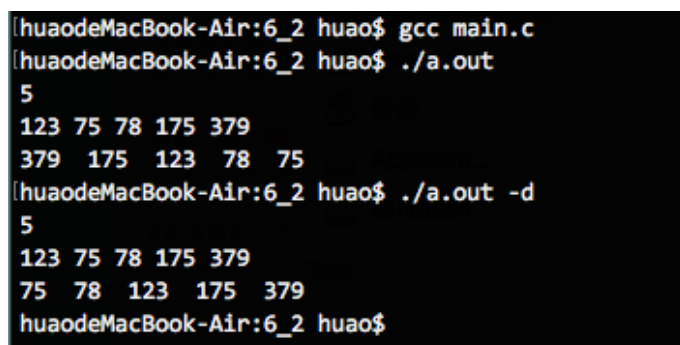
2) 测试

a) 测试数据

5

123 75 78 175 379

b) 对测试数据的运行结果截图如图 6-6



```

[huaodeMacBook-Air:6_2 huaos$ gcc main.c
[huaodeMacBook-Air:6_2 huaos$ ./a.out
5
123 75 78 175 379
379 175 123 78 75
[huaodeMacBook-Air:6_2 huaos$ ./a.out -d
5
123 75 78 175 379
75 78 123 175 379
huaodeMacBook-Air:6_2 huaos$

```

图 6-6

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

a. 计算每个学生各门课程平均成绩；

b. 计算全班每门课程的平均成绩；

c. 分别统计低于全班各门课程平均成绩的人数；

d. 分别统计全班各门课程不及格的人数和 90 分以上（含 90 分）的人数。

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

解答：

1) 源代码清单

```
#include <stdio.h>
#define N 5          //N 个学生
#define M 3          //M 门课程
void a(int stu[N][M]);      //每个学生课程的平均成绩
void b(int stu[N][M]);      //每门课程的平均成绩
void c(int stu[N][M]);      //低于平均成绩的人数
void d(int stu[N][M]);      //不及格人数 及 90 分以上的人数
int main(void)
{
    int stu[N][M];
    char class[M][20];
    int i, j;
    for(i = 0; i < M; i++)
    {
        fgets(*(class + i), 20, stdin);
        for(j = 0; (*(class + i) + j) != '\n'; j++);
        (*(class + i) + j) = '\0';
    }
    for(i = 0; i < N; i++)
        for(j = 0; j < M; j++)
            scanf("%d", (*(stu + i) + j));
    a(stu);
    b(stu);
    c(stu);
    d(stu);
    return 0;
}
void a(int stu[N][M])
{
    int j;
    for(j = 0; j < N; j++)
    {
        int sum = 0;
        int i;
```

```

        for(i = 0; i < M; i++)
            sum += (*(stu + j) + i);
        printf("%4.2lf    ", 1.0 * sum / N);
    }
    putchar("\n");
}

void b(int stu[N][M])
{
    int i;
    for(i = 0; i < M; i++)
    {
        int sum = 0;
        int j;
        for(j = 0; j < N; j++)
            sum += (*(stu + j) + i);
        printf("%4.2lf    ", 1.0 * sum / N);
    }
    putchar("\n");
}

void c(int stu[N][M])
{
    int i, j;
    int sum, num;
    double avg;
    for(i = 0; i < M; i++)
    {
        sum = 0;
        num = 0;
        for(j = 0; j < N; j++)
            sum += (*(stu + j) + i);
        avg = 1.0 * sum / N;
        for(j = 0; j < N; j++)
            if(*(stu + j) + i < avg)
                num++;
        printf("%4d", num);
    }
}

```

```

    }
    putchar('\n');
}
void d(int stu[N][M])
{
    int num_low = 0, num_high = 0;
    int i, j;
    for(i = 0; i < M; i++)
    {
        num_low = 0;
        num_high = 0;
        for(j = 0; j < N; j++)
        {
            if(*(stu + j) + i >= 90)
                num_high++;
            else if(*(stu + j) + i < 60)
                num_low++;
        }
        printf("%4d%4d\n", num_low, num_high);
    }
}

```

2) 测试

a) 测试数据	english
	math
	chinese
	100 100 100
	60 60 60
	50 50 59
	80 90 30

b) 对测试数据的运行结果截图如图 6-7

```

english
math
chinese
10 20 30
100 100 100
60 60 60
50 50 59
80 90 30
12.00      60.00      36.00      31.80      40.00
60.00      64.00      55.80
  2   3   2
  2   1
  2   2
  3   1
Program ended with exit code: 0

```

图 6-7

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:

1) 源代码清单

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

```

int main(void)
{
    char num1[31];
    char num2[31];
    int i;
    num1[0] = 0;
    num2[0] = 0;
    for(i = 1; i < 31; i++)
    {
        num1[i] = getchar() - '0';
        if(num1[i] == ('.' - '0'))
            i--;
    }
    getchar();
    for(i = 1; i < 31; i++)
    {

```

```

        num2[i] = getchar() - '0';
        if(num2[i] == ('.' - '0'))
            i--;
    }
    for(i = 1; i < 31; i++)
        num1[i] += num2[i];
    for(i = 30; i > 0; i--)
    {
        if(num1[i] >= 10)
        {
            num1[i - 1]++;
            num1[i] %= 10;
        }
    }
    for(i = 0; i < 31; i++)
    {
        if(num1[0] == 0)
            ;
        else
            printf("%d", num1[i]);
        if(i == 20)
            putchar('.');
    }
    putchar('\n');
    return 0;
}

```

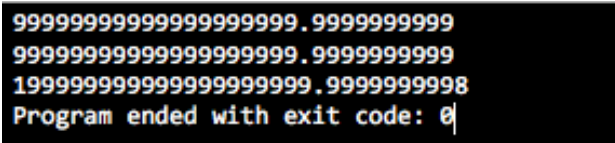
2) 测试

a) 测试数据

99999999999999999999.9999999999

99999999999999999999.9999999999

b) 对测试数据的运行结果截图如图 6-8



```

99999999999999999999.9999999999
99999999999999999999.9999999999
19999999999999999999.9999999998
Program ended with exit code: 0

```

图 6-8

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

提示: `p` 中元素可为 `strcmp`、`strstr` 等函数名。

解答:

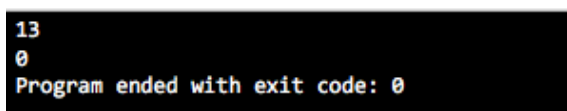
1) 源代码清单

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(void)
{
    char str1[40] = "Hello world! Hello C programming!";
    char str2[40] = "Hello C";
    char *str;
    char>(*p[2])(const char *,const char *);
    p[0] = strstr;
    p[1] = strpbrk;
    str = (*p[0])(str1, str2);
    if(str == NULL)
        printf("没有字串\n");
    else
        printf("%ld\n", str - str1);
    str = (*p[1])(str1, str2);
    if(str == NULL)
        printf("没有 str2 中的字符\n");
    else
        printf("%ld\n", str - str1);
    return 0;
}
```

2) 测试

a) 测试数据

b) 对测试数据的运行结果截图如图 6-9



```
13
0
Program ended with exit code: 0
```

图 6-9

6.3 指定 main 函数的参数

选择“project/ set programs' arguments...”菜单命令，即可打开图 6-10 所示的对话框，在“Program arguments”文本框中输入 main 函数的参数。注意只输入命令行中文件名后的参数，文件名不输入。

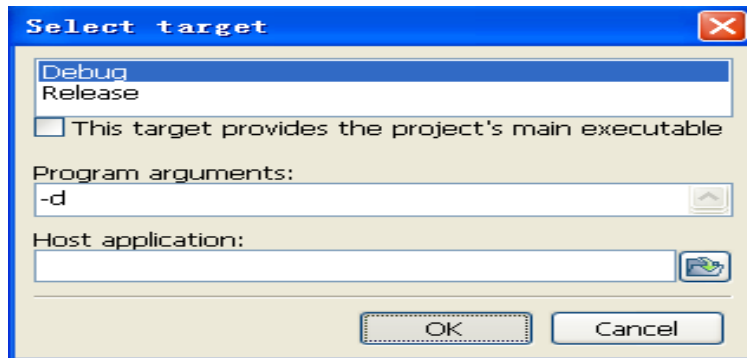


图 6-10 输入 main 函数的参数

6.4 实验小结

在使用指针的过程中，要时刻注意不能通过指针访问未分配给程序的内存空间，否则可能导致程序崩溃。

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)->x	100	100

2	p++,p->c	B	B
3	*p++->t,*p->t	x	x
4	*(++p)->t	x	x
5	*++p->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：

```

1  #include "stdio.h"
2  #include "stdlib.h"
3  struct s_list{
4      int data; /* 数据域 */
5      struct s_list *next; /* 指针域 */
6  };
7  void create_list (struct s_list *headp,int *p);
8  void main(void)
9  {
10     struct s_list *head=NULL,*p;
11     int s[]={1,2,3,4,5,6,7,8,0}; /* 0 为结束标记 */
12     create_list(head,s); /* 创建新链表 */
13     p=head; /*遍历指针 p 指向链头 */
14     while(p){
15         printf("%d\t",p->data); /* 输出数据域的值 */
16         p=p->next; /*遍历指针 p 指向下一结点 */
17     }
18     printf("\n");

```

```

19 }
20 void create_list(struct s_list *headp,int *p)
21 {
22     struct s_list * loc_head=NULL,*tail;
23     if(p[0]==0) /* 相当于*p==0 */
24         ;
25     else { /* loc_head 指向动态分配的第一个结点 */
26         loc_head=(struct s_list *)malloc(sizeof(struct s_list));
27         loc_head->data=*p++; /* 对数据域赋值 */
28         tail=loc_head; /* tail 指向第一个结点 */
29         while(*p){ /* tail 所指结点的指针域指向动态创建的结点 */
30             tail->next=(struct s_list *)malloc(sizeof(struct s_list));
31             tail=tail->next; /* tail 指向新创建的结点 */
32             tail->data=*p++; /* 向新创建的结点的数据域赋值 */
33         }
34         tail->next=NULL; /* 对指针域赋 NULL 值 */
35     }
36     headp=loc_head; /* 使头指针 headp 指向新创建的链表 */
37 }

```

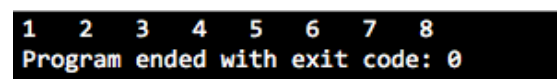
(2) 修改替换 create_list 函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

解答：

(1)

- 1) 错误为：输出结果为空
- 2) 错误修改如下：
 - a) 第七行和第二十行函数的第一个参数改为：struct s_list **headp
 - b) 第 12 行调用函数时的第一个参数改为：&head

3) 修改后运行结果截图如图 7-1



```

1  2  3  4  5  6  7  8
Program ended with exit code: 0

```

图 7-1

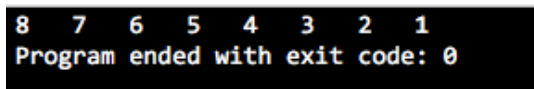
(2) 修改后的函数为

```

void create_list(struct s_list **headp, int *p){
    struct s_list *loc_head = NULL, *tail;
    if(p[0] == 0);
    else{
        loc_head = (struct s_list *)malloc(sizeof(struct s_list));
        loc_head->data = *p++;
        loc_head->next = NULL;
        while(*p){
            tail = loc_head;
            loc_head = (struct s_list *)malloc(sizeof(struct s_list));
            loc_head->data = *p++;
            loc_head->next = tail;
        }
    }
    *headp = loc_head;
}

```

修改后运行结果截图如图 7-2



```

8 7 6 5 4 3 2 1
Program ended with exit code: 0

```

图 7-2

7.2.3 编程设计题

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

```

void f0(struct bits b)
{

```

```
    Printf("the function %d is called!\n",b);  
}
```

解答：

1) 源代码清单

```
# include <stdio.h>  
struct bits{  
    unsigned char bit0:1;  
    unsigned char bit1:1;  
    unsigned char bit2:1;  
    unsigned char bit3:1;  
    unsigned char bit4:1;  
    unsigned char bit5:1;  
    unsigned char bit6:1;  
    unsigned char bit7:1;  
};  
void f0(struct bits b)  
{  
    printf("the function %d is called!\n",b);  
}  
void f1(struct bits b)  
{  
    printf("the function %d is called!\n",b);  
}  
void f2(struct bits b)  
{  
    printf("the function %d is called!\n",b);  
}  
void f3(struct bits b)  
{  
    printf("the function %d is called!\n",b);  
}  
void f4(struct bits b)  
{  
    printf("the function %d is called!\n",b);  
}
```

```

void f5(struct bits b)
{
    printf("the function %d is called!\n",b);
}
void f6(struct bits b)
{
    printf("the function %d is called!\n",b);
}
void f7(struct bits b)
{
    printf("the function %d is called!\n",b);
}
int main(void){
    struct bits b = {1, 0, 0, 0, 1, 0, 0, 0};
    int num = b.bit0 + b.bit1 + b.bit2 + b.bit3 + b.bit4 + b.bit5 + b.bit6 + b.bit7;
    void (*p_fun[8])(struct bits);
    p_fun[0] = f0;
    p_fun[1] = f1;
    p_fun[2] = f2;
    p_fun[3] = f3;
    p_fun[4] = f4;
    p_fun[5] = f5;
    p_fun[6] = f6;
    p_fun[7] = f7;
    if(b.bit0 == 1)
        p_fun[0](b);
    if(num >= 2){
        if(b.bit1 == 1)
            p_fun[1](b);
        if(b.bit2 == 1)
            p_fun[2](b);
        if(b.bit3 == 1)
            p_fun[3](b);
        if(b.bit4 == 1)
            p_fun[4](b);
    }
}

```

```

        if(b.bit5 == 1)
            p_fun[5](b);
        if(b.bit6 == 1)
            p_fun[6](b);
        if(b.bit7 == 1)
            p_fun[7](b);

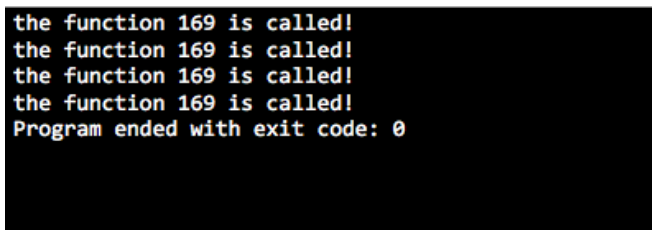
    }
    return 0;
}

```

2) 测试

a) 测试数据 $b = \{1, 0, 0, 1, 0, 1, 0, 1\}$

b) 对测试数据的运行结果截图如图 7-3



```

the function 169 is called!
the function 169 is called!
the function 169 is called!
the function 169 is called!
Program ended with exit code: 0

```

图 7-3

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

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

解答：

1) 源代码清单

```

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

```

```

struct student{
    unsigned int num_stu;           //学号

```



```

    char name[20];
    double english, math, physics, C;
    double avg;
    struct student *next;
};
void input(struct student *);
void output1(struct student *);
void change(struct student *, unsigned int, const int, int);
void avg(struct student *);
void output2(struct student *);
int main(void){
    struct student *head = (struct student *)malloc(sizeof(struct student));
    unsigned int num;
    int i;
    double score;
    input(head);
    output1(head);
    scanf("%u %d %lf", &num, &i, &score);
    change(head, num, i, score);
    avg(head);
    output2(head);
    return 0;
}
void input(struct student *head){
    while(scanf("%u %s %lf %lf %lf %lf", &head->num_stu, head->name,
&head->english, &head->math, &head->physics, &head->C)){
        head->next = (struct student *)malloc(sizeof(struct student));
        head = head->next;
    }
    getchar();
    head->next = NULL;
}
void change(struct student *head, unsigned int num, const int i, int score){
    //用学号 num 标识学生 用 i 表示需要修改的信息
    while(1){

```

```

        if(head->num_stu == num)
            break;
        head = head->next;
    }
    switch(i){
        case 1:head->english = score;
            break;
        case 2:head->math = score;
            break;
        case 3:head->physics = score;
            break;
        case 4:head->C = score;
            break;
    }
}
void output1(struct student *head){
    while(1){
        if(head->next == NULL)
            break;
        printf("%u\t%s\t%.0lf\t%.0lf\t%.0lf\t%.0lf\t\n", head->num_stu, head->name,
head->english, head->math, head->physics, head->C);
        head = head->next;
    }
}
void avg(struct student *head){
    while(1){
        if(head->next == NULL)
            break;
        head->avg = (head->english + head->math + head->physics + head->C) / 4;
        head = head->next;
    }
}
void output2(struct student *head){
    while(1){
        if(head->next == NULL)

```

```

        break;
        printf("%u\t%s\t%.0lf\t%.0lf\t%.0lf\t%.0lf\t%.2lf\n", head->num_stu,
head->name, head->english, head->math, head->physics, head->C, head->avg);
        head = head->next;
    }
}

```

2) 测试

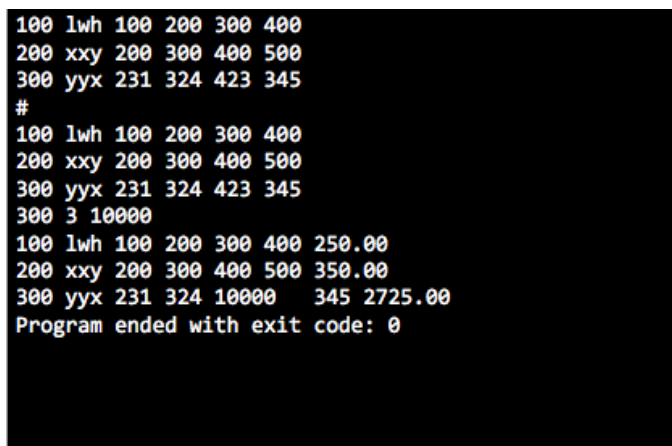
a) 测试数据

```

100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 324 423 345#

```

b) 对测试数据的运行结果截图如图 7-4



```

100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 324 423 345
#
100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 324 423 345
300 3 10000
100 lwh 100 200 300 400 250.00
200 xxy 200 300 400 500 350.00
300 yyx 231 324 10000 345 2725.00
Program ended with exit code: 0

```

图 7-4

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。

解答:

1) 源代码清单

```

void sort1(struct student *head){
    struct student *p = head;
    char t[20];
    while(1){
        if(head->next == NULL)

```

```

        break;
p = head;
while(1){
    if(p->next == NULL)
        break;
    if(head->avg > p->avg){
        SWAP(head->num_stu, p->num_stu);
        SWAP(head->english, p->english);
        SWAP(head->math, p->math);
        SWAP(head->physics, p->physics);
        SWAP(head->C, p->C);
        SWAP(head->avg, p->avg);
        strcpy(t, head->name);
        strcpy(head->name, p->name);
        strcpy(p->name, t);
    }
    p = p->next;
}
head = head->next;
}
}

```

2) 测试

a) 测试数据

```

100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 342 423 345#
200 2 900

```

b) 对测试数据的运行结果截图如图 7-5

```

100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 342 423 345#
100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 342 423 345
200 2 900
100 lwh 100 200 300 400 250.00
200 xxy 200 900 400 500 500.00
300 yyx 231 342 423 345 335.25
100 lwh 100 200 300 400 250.00
300 yyx 231 342 423 345 335.25
200 xxy 200 900 400 500 500.00
Program ended with exit code: 0

```

图 7-5

(2)对选做题第(1)题,进一步写出用交换结点指针域的方法升序排序的函数。

解答:

1) 源代码清单

```

void sort2(struct student *head){
    struct student *p1 = head, *Head;
    Head = (struct student *)malloc(sizeof(struct student));
    Head->next = head;           //链表头
    while(1){
        if(head->next == NULL)
            break;
        p1 = head;
        while(1){
            if(p1->next == NULL)
                break;
            if(head->avg > p1->avg){
                head->next = p1->next;
                Head->next = p1;
                p1->next = head;
            }
            p1 = p1->next;
        }
        Head = Head->next;
        head = head->next;
    }
}

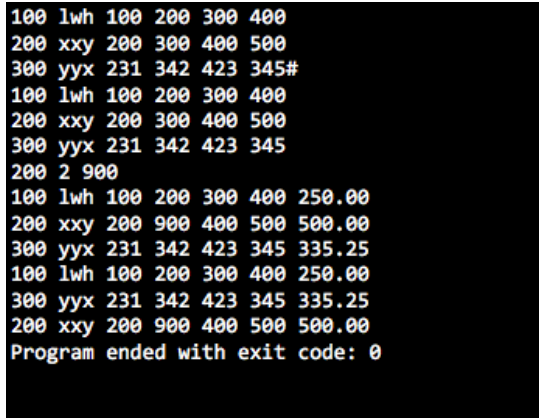
```

2) 测试

a) 测试数据

```
100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 342 423 345#
200 2 900
```

b) 对测试数据的运行结果截图如图 7-6



```
100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 342 423 345#
100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 342 423 345
200 2 900
100 lwh 100 200 300 400 250.00
200 xxy 200 900 400 500 500.00
300 yyx 231 342 423 345 335.25
100 lwh 100 200 300 400 250.00
300 yyx 231 342 423 345 335.25
200 xxy 200 900 400 500 500.00
Program ended with exit code: 0
```

图 7-6

(3) 采用双向链表重做编程设计题第 (2) 题。

解答:

1) 源代码清单

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

struct student{
    unsigned int num_stu;          //学号
    char name[20];
    double english, math, physics, C;
    double avg;
    struct student *before;
    struct student *next;
};

void input(struct student *);
void output1(struct student *);
void change(struct student *, unsigned int, const int, int);
void avg(struct student *);
```

```

void output2(struct student *);
void output3(struct student *);
int main(void){
    struct student *head = (struct student *)malloc(sizeof(struct student));
    unsigned int num;
    int i;
    double score;
    head->before = NULL;
    input(head);
    output1(head);
    scanf("%u %d %lf", &num, &i, &score);
    change(head, num, i, score);
    avg(head);
    output2(head);
    output3(head);
    return 0;
}

void input(struct student *head){
    while(scanf("%u %s %lf %lf %lf %lf", &head->num_stu, head->name,
&head->english, &head->math, &head->physics, &head->C)){
        head->next = (struct student *)malloc(sizeof(struct student));
        head->next->before = head;
        head = head->next;
    }
    getchar();
    head->next = NULL;
}

void change(struct student *head, unsigned int num, const int i, int score){
    //用学号 num 标识学生 用 i 表示需要修改的信息
    while(1){
        if(head->num_stu == num)
            break;
        head = head->next;
    }
    switch(i){

```

```

        case 1:head->english = score;
            break;
        case 2:head->math = score;
            break;
        case 3:head->physics = score;
            break;
        case 4:head->C = score;
            break;
    }
}

void output1(struct student *head){
    while(1){
        if(head->next == NULL)
            break;

        printf("%u\t%s\t%.0lf\t%.0lf\t%.0lf\t%.0lf\t\n", head->num_stu, head->name,
head->english, head->math, head->physics, head->C);
        head = head->next;
    }
}

void avg(struct student *head){
    while(1){
        if(head->next == NULL)
            break;

        head->avg = (head->english + head->math + head->physics + head->C) / 4;
        head = head->next;
    }
}

void output2(struct student *head){
    while(1){
        if(head->next == NULL)
            break;

        printf("%u\t%s\t%.0lf\t%.0lf\t%.0lf\t%.0lf\t%.2lf\n", head->num_stu,
head->name, head->english, head->math, head->physics, head->C, head->avg);
        head = head->next;
    }
}

```



```

}
void output3(struct student *head){
    while(1){
        if(head->next->next == NULL)
            break;
        head = head->next;
    }
    while(1){
        if(head == NULL)
            break;
        printf("%u\t%s\t%.0lf\t%.0lf\t%.0lf\t%.0lf\t%.2lf\n", head->num_stu,
head->name, head->english, head->math, head->physics, head->C, head->avg);
        head = head->before;
    }
}
}

```

2) 测试

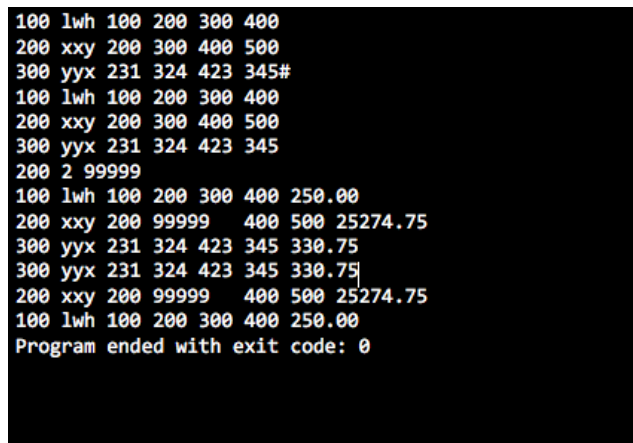
a) 测试数据

```

100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 324 423 345#
200 2 99999

```

对测试数据的运行结果截图如图 7-7



```

100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 324 423 345#
100 lwh 100 200 300 400
200 xxy 200 300 400 500
300 yyx 231 324 423 345
200 2 99999
100 lwh 100 200 300 400 250.00
200 xxy 200 99999 400 500 25274.75
300 yyx 231 324 423 345 330.75
300 yyx 231 324 423 345 330.75
200 xxy 200 99999 400 500 25274.75
100 lwh 100 200 300 400 250.00
Program ended with exit code: 0

```

图 7-7

7.3 实验小结

注意链表使用过程中指针的移动以及某些时候指针的值改变后可能造成的数据丢失和内存泄漏问题。

8 文件实验

8.1 实验目的

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

8.2 实验题目及要求

8.2.1 文件类型的程序验证题

设有程序:

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

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

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

```

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

```

请思考程序的输出结果，然后通过上机运行来加以验证。

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

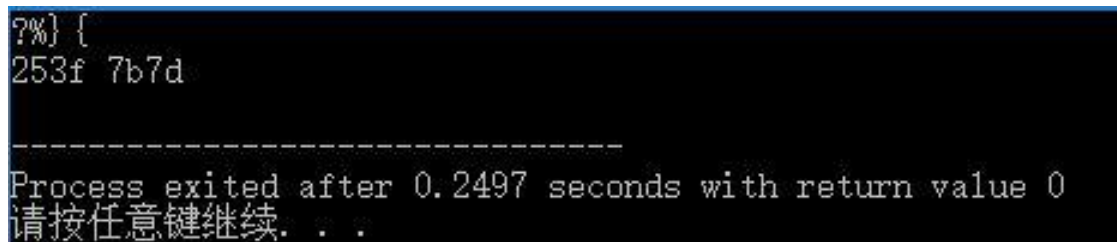
将 `fprintf(fp2, "%hx %hx", a, b)` 改为 `fprintf(fp2, "%d %d", a, b)` 结果有什么不同。

解答：

1) 输出结果为：?%}{

253f 7b7d

上机运行结果如图 8-1



```

?%}{
253f 7b7d
-----
Process exited after 0.2497 seconds with return value 0
请按任意键继续. . .

```

图 8-1

2) 输出结果为：?}

253f 7b7d

原因为：执行两行 `fwrite` 函数时仅读取 1 个字节的数据，因此写入 `fp1` 的数据为 `3f(?)` 和 `7d({)}`。

3) 输出结果为：?%}{

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
16     while(ch=fgetc(fp)!=EOF)                    /* 从 filename 中读字符 */
17         putchar(ch);                            /* 向显示器中写字符 */
18     fclose(fp);                                /* 关闭 filename */
19     return 0;
20 }

```

(2) 用输入输出重定向 `freopen` 改写 `main` 函数。

解答：

(1) 错误修改：将第 16 行改为： `while((ch = fgetc(fp)) != EOF)`

(2) 修改后的源代码如下：

```

#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);
    }
}

```

```

    freopen(argv[1], "r", stdin);
    while((ch=getchar())!=EOF)        /* 从 filename 中读字符 */
        putchar(ch);                /* 向显示器中写字符 */
    return 0;
}

```

8.2.3 编程设计题

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

解答：

1) 源代码清单：

```

#include <stdio.h>
int main(int argc, char *argv[]){
    char c;
    FILE *fp;
    fp = fopen("c:\\files\\test.txt", "w+");
    c = getchar();
    c = c - 'a' + 'A';
    fputc(c, fp);
    while(1){
        c = getchar();
        if(c == '\n')
            break;
        else if(c != ' ')
            fputc(c, fp);
        else{
            fputc(c, fp);
            c = getchar();
            c = c - 'a' + 'A';
            fputc(c, fp);
        }
    }
    fprintf(fp, "\n");
    fclose(fp);
}

```

```
return 0;  
}
```

2) 运行结果截图如图 8-2

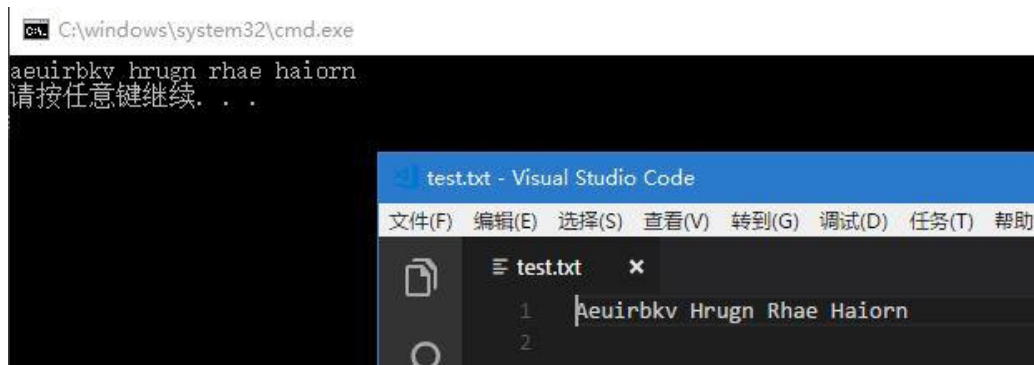


图 8-2

8.3 实验小结

注意在源代码中输入文件路径时转移字符的使用。

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