

# 华中科技大学

## 课程实验报告

课程名称： C 语言程序设计

专业班级： CS1601

学 号： U201614532

姓 名： 吕鹏泽

指导教师： 甘早斌

报告日期： \_\_\_\_\_

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 .....	4
1.1	实验目的 .....	4
1.2	实验内容 .....	4
1.3	实验小结 .....	13
2	流程控制实验 .....	14
2.1	实验目的 .....	14
2.2	实验内容 .....	14
2.3	实验小结 .....	32
3	函数与程序结构实验 .....	33
3.1	实验目的 .....	33
3.2	实验内容 .....	33
3.3	实验小结: .....	47
4	编译预处理实验 .....	49
4.1	实验目的 .....	49
4.2	实验内容 .....	49
4.3	实验小结 .....	63
5	数组实验 .....	65
5.1	实验目的 .....	65
5.2	实验内容 .....	65
5.3	实验小结: .....	81

6	指针实验 .....	82
6.1	实验目的.....	82
6.2	实验题目及要求.....	82
6.3	实验小结: .....	105
7	结构与联合实验 .....	106
7.1	实验目的.....	106
7.2	实验题目及要求.....	106
7.3	实验总结.....	129
8	文件实验 .....	130
8.1	实验目的.....	130
8.2	实验题目及要求.....	130
8.3	实验小结.....	138
	参考文献: .....	139

# 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.  voidmain( void )
```

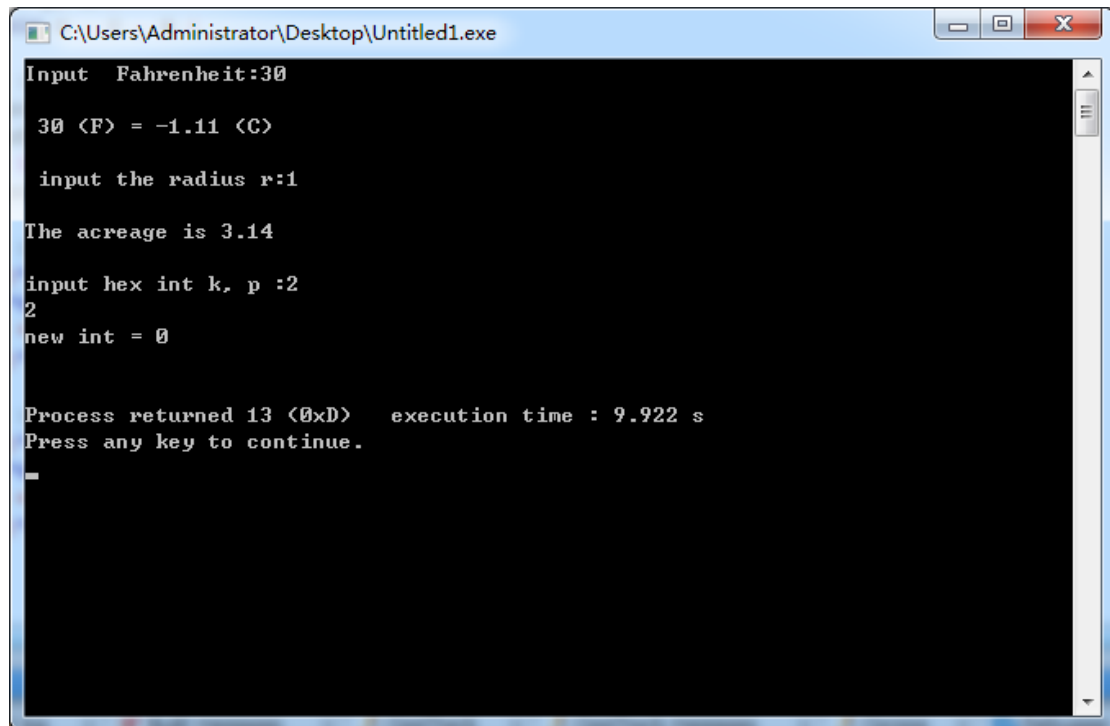
```
4.  {  
    a)  int f ;  
    b)  short p, k ;  
    c)  double c , r , s ;  
  
5.  /* for task 1 */  
    a)  printf( "Input  Fahrenheit:" ) ;  
    b)  scanf( "%d" , f ) ;  
    c)  c = 5/9*(f-32) ;//  
    d)  printf( "  \n %d (F) = %.2lf (C)\n\n " , f,  
c ) ;  
  
6.  /* for task 2 */  
7.  printf("input the radius r:");  
8.  scanf("%f", &r);  
9.  s = PI * r * r;  
10. printf("\nThe acreage is %.2f\n\n",&s);  
11. printf("\nThe acreage is %.2f\n\n",s);  
  
12. /* for task 3 */  
13. printf("input hex int k, p :");  
14. scanf("%x %x", &k, &p );
```

```
15. newint = (p&0xff00) | (k&0xff00)<<8; // printf("new
int = %x\n\n", newint);

16. }
```

**解答:**

1. 2. define 后无分号, 正确的形式为 #define PI 3.14159
2. 3. void 后面没空格, 正确的形式为 void main(void)
3. 5. a) 引号错误, 正确形式为 printf("Input Fahrenheit:");
4. 5. b) f 前缺少 &, 引号错误, 正确形式为 scanf("%d", &f );
5. 5. c) 5/9 为整形, 正确的形式为 5.0/9.0
6. 5. d) 引号错误, %f 转换符少了个 1, 正确的形式为 printf( " \n %d (F) = %.21f (C)\n\n ", f);
7. 8. %f 转换符少 1, 正确的形式为 scanf("%lf", &r);
8. 10. s 前多了 &, 正确的形式为 printf("\nThe acreage is %.2f\n\n", s);
9. 15. newint 未声明, <<, 正确的形式为 newint = (p&0xff00) | ((k&0xff00)>>8), 声明 int newint;



```
C:\Users\Administrator\Desktop\Untitled1.exe
Input Fahrenheit:30
30 (F) = -1.11 (C)
input the radius r:1
The acreage is 3.14
input hex int k, p :2
2
new int = 0

Process returned 13 (0xD) execution time : 9.922 s
Press any key to continue.
```

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);
}
```

替换后的程序如下所示：

```
#include<stdio.h>

void 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);

}
```

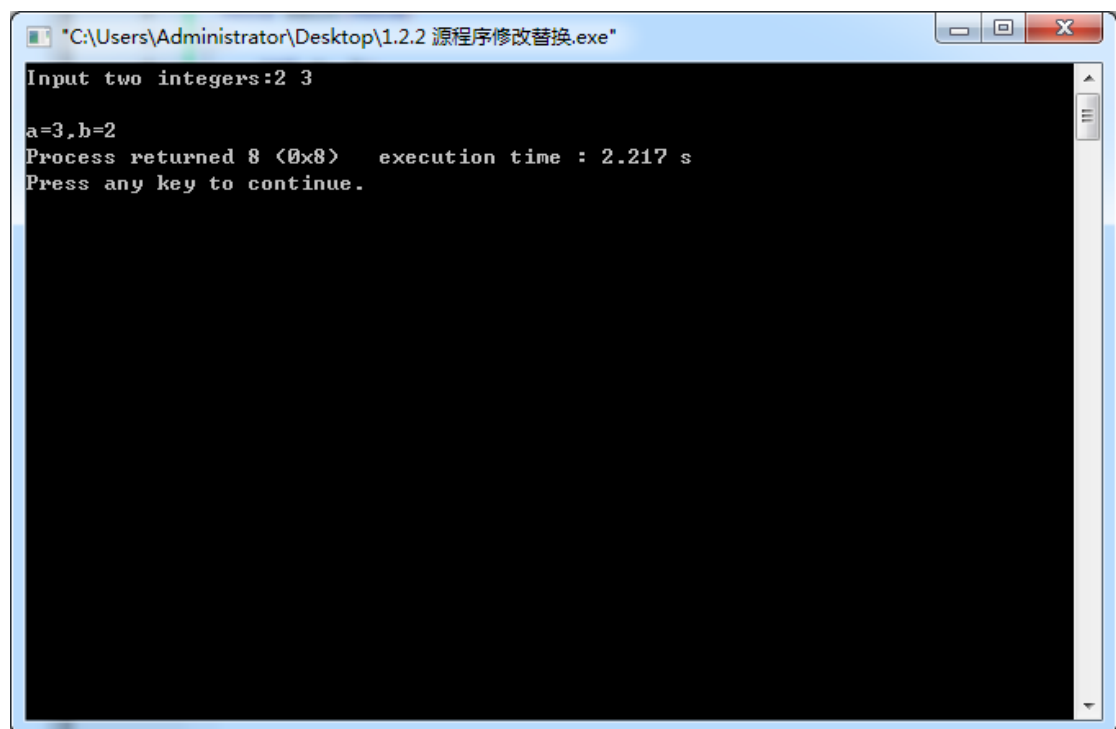


图 1- 2



### 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) 解题思路：

1. 使用 if 语句判断，A-Z 期间的字母  $c$  输出  $c+'a'-'A'$ 。
2. 否则输出原字符。

2) 程序清单

```
#include<stdio.h>

int main(void)
{
    char c;

    if((c=getchar())>='A' && c<='Z')//如果 c 是大写

        c=c+'a'-'A';//读取的大写转换为小写

    putchar(c);//输出

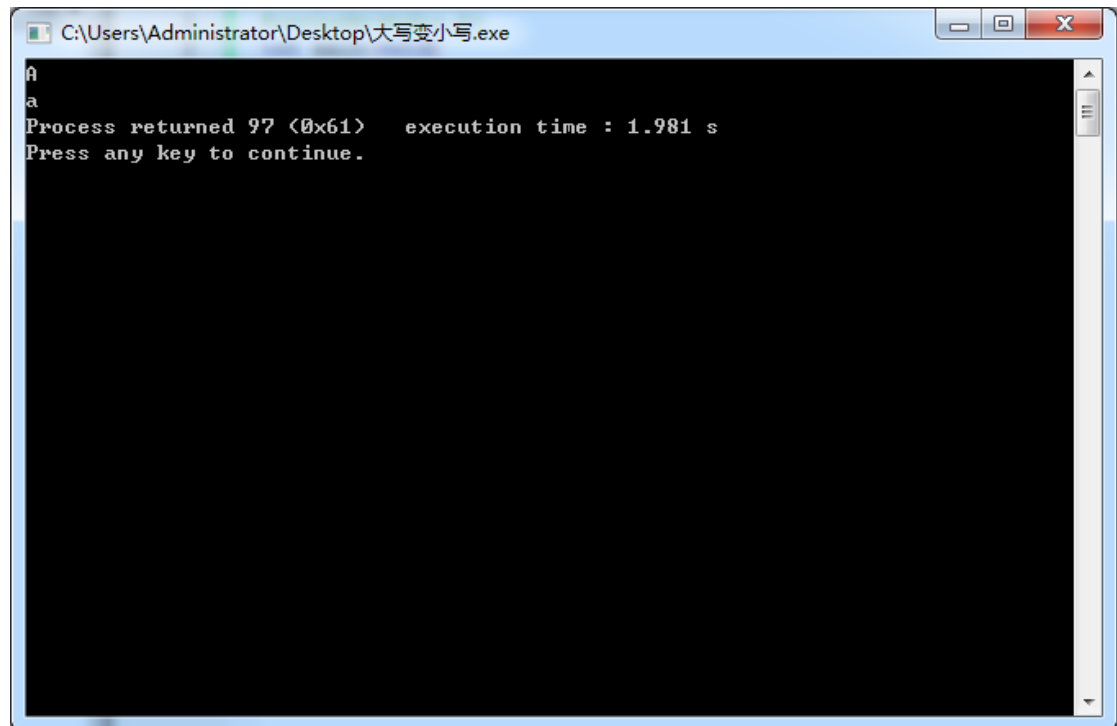
}
```

3) 测试

(a) 测试数据：

A

(b) 测试结果：

[illegible]

解答：

1) 解题思路:

1. 循环移位构造逻辑尺
2.  $x$  与逻辑运算获得目标数。

## 2) 程序清单

```
#include<stdio.h>

int main(void)
{
    unsigned short x,m,n,mask,mask1;

    int i;
```

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

for(i=m ,mask=0;i<=m+n-1;i++)

{

    mask1= 1<<i;

    mask+=mask1;//构造逻辑尺

}

x=x&mask;//取得 x

x=x<<(16-(m+n));//x 移位到左端

printf("%hu",x);

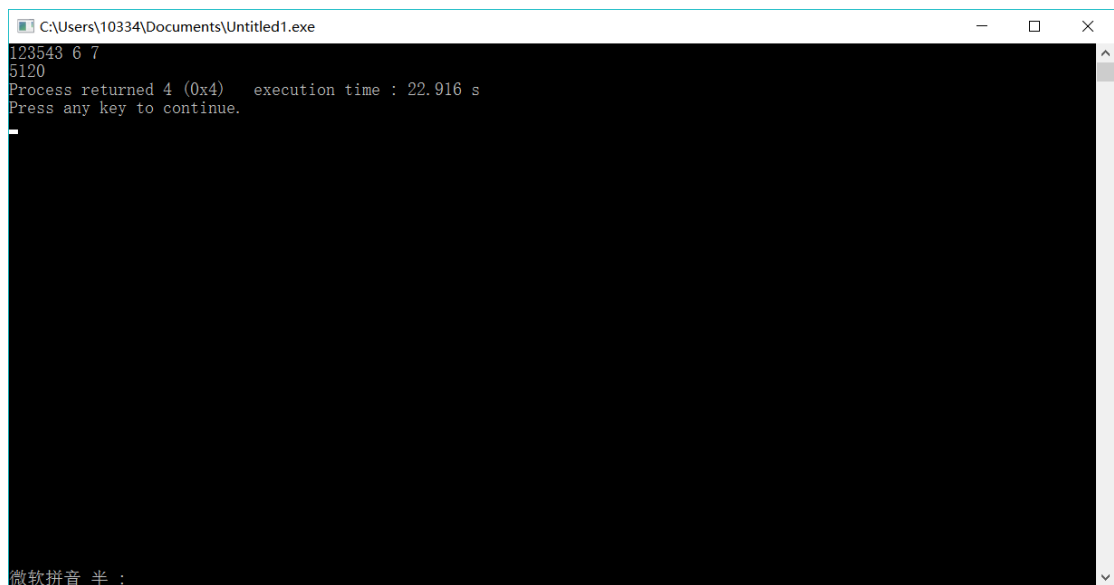
}
```

### 3) 测试

#### (a) 测试数据:

123543 6 7

#### (b)



```
C:\Users\10334\Documents\Untitled1.exe
123543 6 7
5120
Process returned 4 (0x4)   execution time : 22.916 s
Press any key to continue.
微软拼音 半:
```

(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) 解题思路：

1. 每 8 位取出数据
2. 将存储的数据存储在变量 ipi 中，最后输出。

2) 程序清单

```
#include<stdio.h>

int main(void)
{
    unsigned long ip, ip1, ip2, ip3, ip4;

    unsigned long mask=255;//0000...0000 1111 1111

    scanf("%u",&ip);//获取 ip

    ip1=ip&mask;//移位，每 8 位存储，8 位之后的置为 0

    ip2=(ip>>8)&mask;

    ip3=(ip>>16)&mask;

    ip4=(ip>>24)&mask;

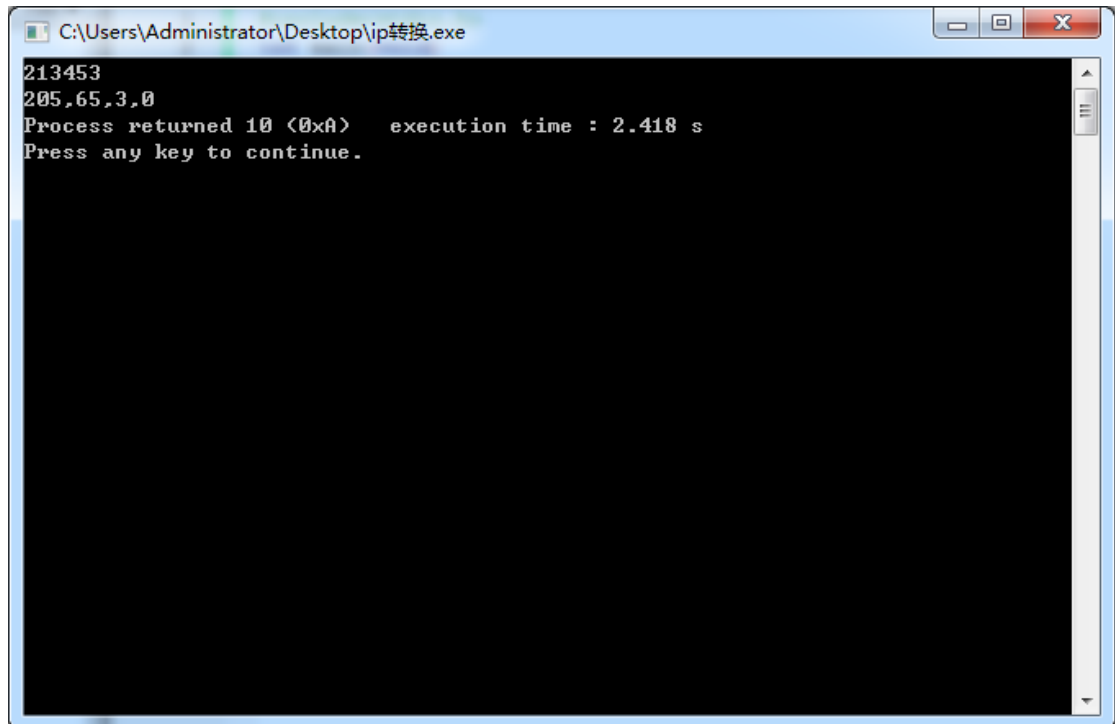
    printf("%u, %u, %u, %u", ip1, ip2, ip3, ip4);
}
```

3) 测试

(a) 测试数据：

213453

(b)



```
C:\Users\Administrator\Desktop\ip转换.exe
213453
205,65,3,0
Process returned 10 (0xA) execution time : 2.418 s
Press any key to continue.
```

### 1.3 实验小结

学习 C 语言要熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型以及运算过程中的类型转换。还要熟记运算符的优先级和结合性。稍有一点错误就会对程序的结果造成影响，导致结果的不准确。所以在这方面一定要严谨。虽然在编写程序时难免犯错，但我们要通过不断的练习来提高正确性！要熟练掌握 `getchar`, `putchar`, `scanf` 和 `printf` 这些基本函数的用法，是编写过程更为流畅。每个 C 程序都要包含顺序结构程序，所以掌握顺序结构程序的编写方法对于程序的编写尤为重要。编写程序时要仔细认真，尽量减少犯错，节省调试时间，做到高效准确！此外要注意每次调运 `scanf` 之后会在缓冲区留下一个换行符，因此有些情况下需要清空这个换行符（常使用 `getchar`），否则会得不到正确的结果。

## 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， $8! = 40320$ 。

程序清单：

```
#include <stdio.h>

void main(void)
{
    int i, n, s = 1;

    printf("Please enter n:");

    scanf("%d", &n); // 键盘读取整型n应加上&
```

```
    for (i = 1; i <= n; i++)//for() 各参数应用分号分开，而不是逗号

        s = s*i;

    printf("%d! = %d", n, s);

}
```

**解答：**

程序清单：

```
#include <stdio.h>

void main(void)

{

    int i, n, s = 1;

    printf("Please enter n:");

    scanf("%d", &n); //键盘读取整型n应加上&

    for (i = 1; i <= n; i++)//for() 各参数应用分号分开，而不是逗号

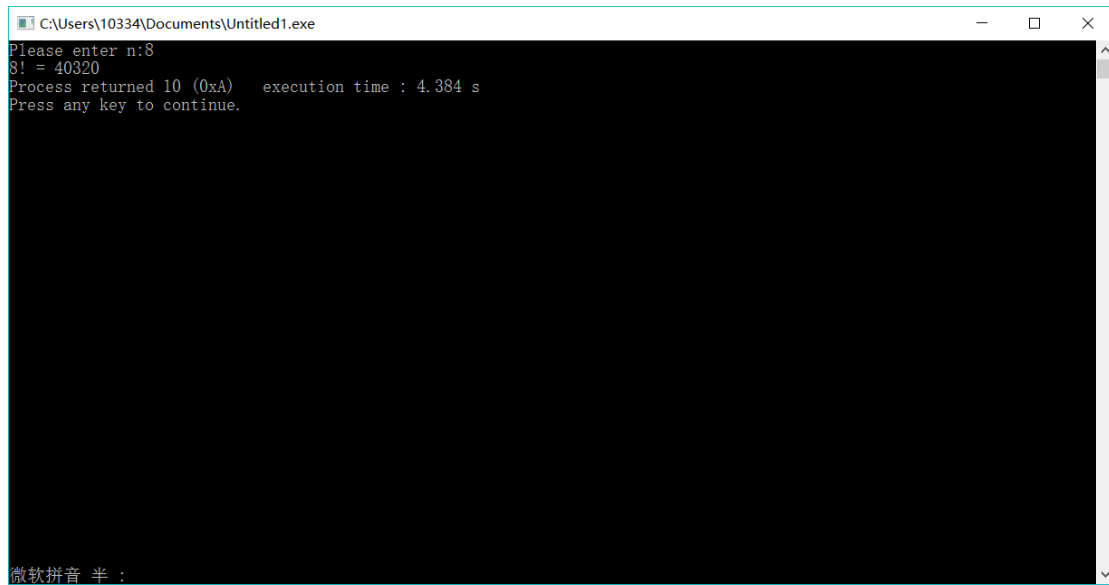
        s = s*i;

    printf("%d! = %d", n, s);

}
```

测试样例：8

测试结果：



```
C:\Users\10334\Documents\Untitled1.exe
Please enter n:8
8! = 40320
Process returned 10 (0xA)   execution time : 4.384 s
Press any key to continue.
```

### 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。

/\*while语句写法\*/

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

```
void main(void)
```

```
{
```

```
    int i=1, n, s = 1;
```

```
    printf("Please enter n:");
```

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

```
    while (i <= n)
```

```
    {
```

```
        s = s*i;
```

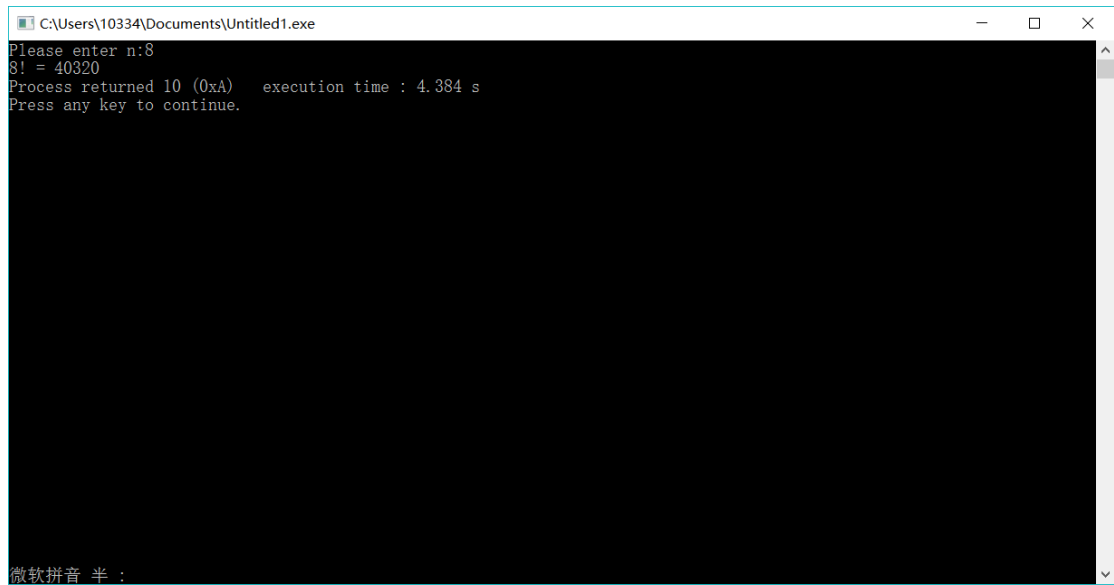
```
        i++;
```

```
    }
```

```
    printf("%d! = %d", n, s);
```



```
}
```



```
/*do-while语句写法*/
```

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

```
void main(void)
```

```
{
```

```
    int i=1, n, s = 1;
```

```
    printf("Please enter n:");
```

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

```
    do
```

```
    {
```

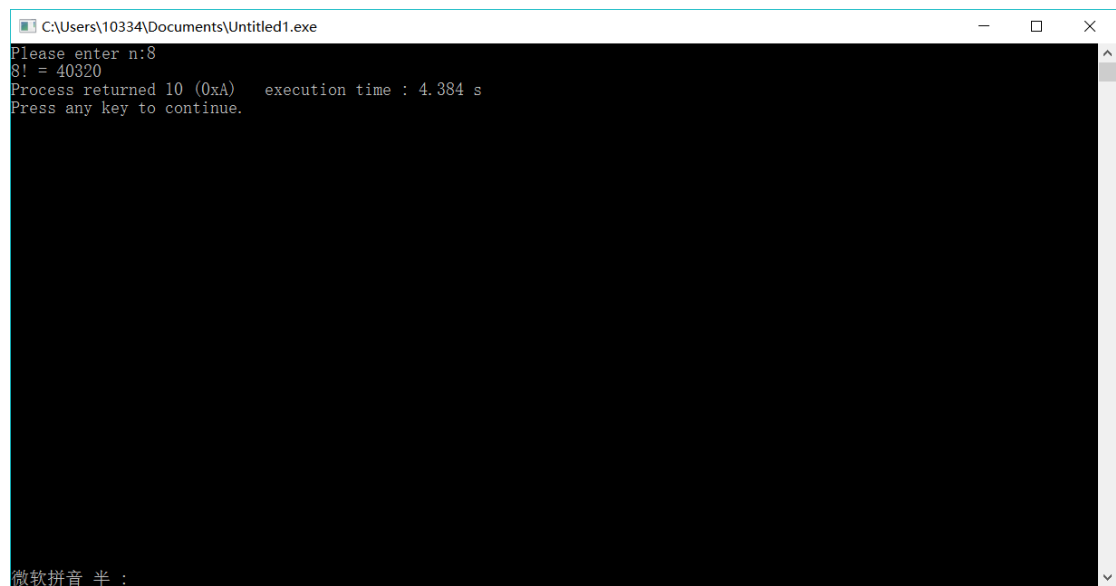
```
        s = s*i;
```

```
        i++;
```

```
    } while (i <= n);
```

```
    printf("%d! = %d", n, s);
```

```
}
```



```
C:\Users\10334\Documents\Untitled1.exe
Please enter n:8
8! = 40320
Process returned 10 (0xA)   execution time : 4.384 s
Press any key to continue.

微软拼音 半:
```

(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  $n=8$ 。

```
/*do-while语句写法*/

#include <stdio.h>

int divisible_max(int S);

void main(void)
{
    int S, result;

    printf("Please enter S:");

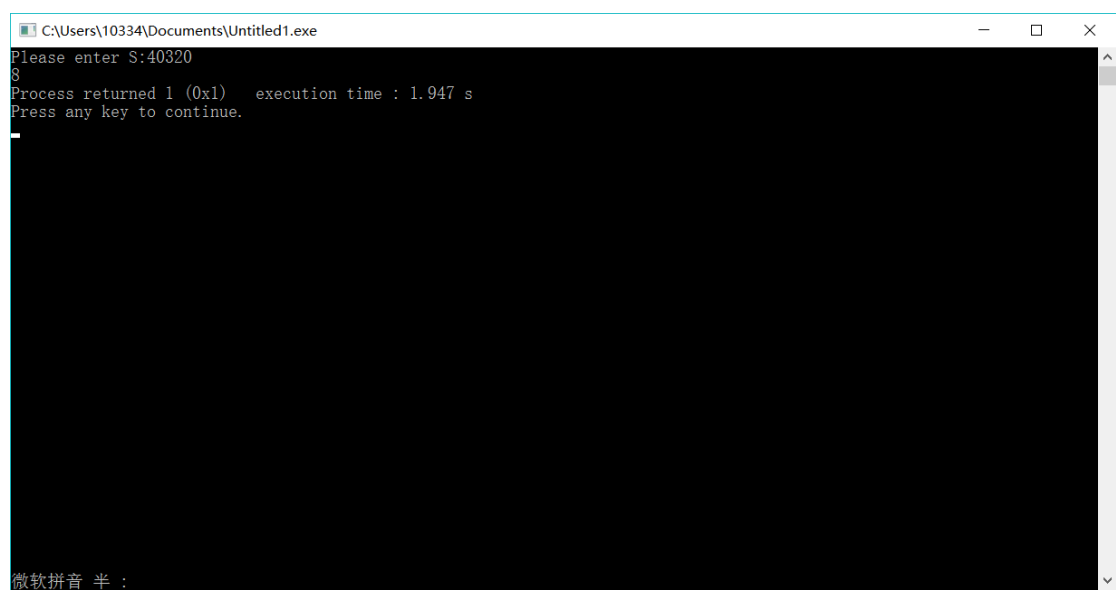
    scanf("%d", &S);

    result = divisible_max(S);

    printf("%d", result);
}

int divisible_max(int S)
```

```
{  
  
    int i = 1, s = 1;  
  
    do  
  
    {  
  
        s *= i;  
  
        i++;  
  
    } while (s < S);  
  
    return i - 1;  
  
}
```



### 3. 编程设计题

(1) 打印如下杨辉三角形。

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。

```

#include<stdio.h>

#define Pas_Triangle_mlenth 20

int main(void)
{
    int num[Pas_Triangle_mlenth][Pas_Triangle_mlenth];

    int i, j, k;

```

```
int n;

for (i = 0; i < Pas_Triangle_mlenth; i++)
{
    num[i][0] = 1;

    num[i][i] = 1;

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

        num[i][j] = num[i][j - 1] * (i - j + 1) / j;

} //初始化Ci, j。

scanf("%d", &n);

getchar();

while (n<=12&& n>=1)

{

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

    {

        for (k = 2 * (n - i-1) ; k > 0; k--)

            putchar(' ');

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

            printf("%-4d", num[i][j]);

        putchar('\n');

    }

}
```

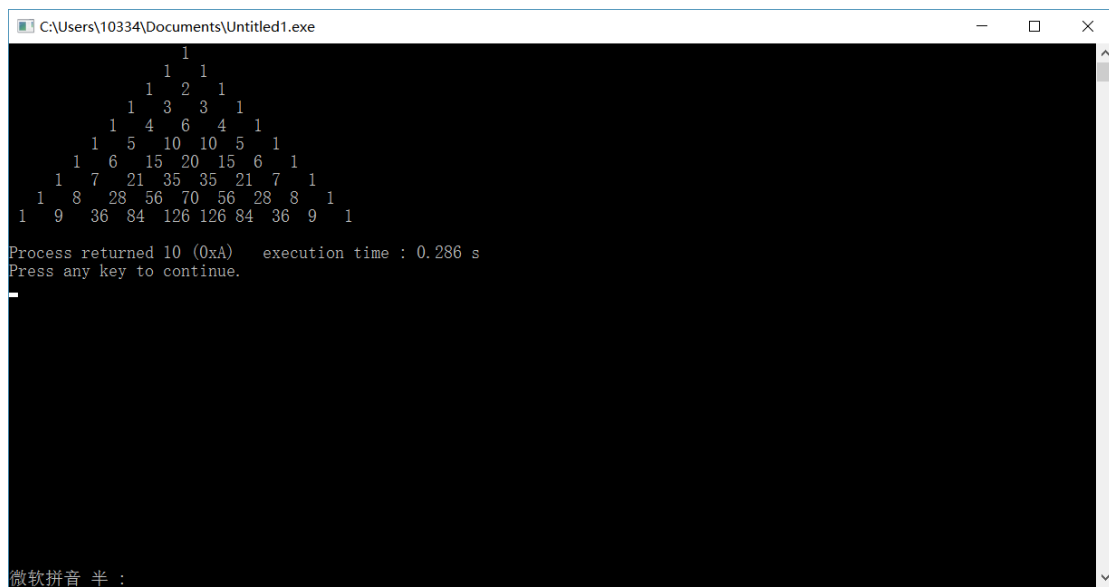
```
        putchar(' \n');

        scanf("%d", &n);

        getchar();

    }

}
```



The screenshot shows a Windows command prompt window titled "C:\Users\10334\Documents\Untitled1.exe". The window displays a Pascal's triangle pattern of numbers. The numbers are arranged in rows, with each row starting and ending with 1. The numbers in the interior of the triangle are the sum of the two numbers directly above them. The pattern is as follows:

```
      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
```

Below the pattern, the text "Process returned 10 (0xA) execution time : 0.286 s" and "Press any key to continue." is visible. At the bottom left of the window, the text "微软拼音 半：" is shown.

(2) 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。

```
#include <stdio.h>

int Digit(int x);

int main(void)
{

    int x, i, j;

    int num[20];

    int newNumber;
```

```
int lenth_of_x;

scanf("%d", &x);

while (x)
{
    lenth_of_x = Digit(x);

    for (i = 0; i <= lenth_of_x; i++)
    {
        num[i] = x % 10; //i=0, 1, 2, 3... 分别取x的个位, 十位, 百
        位, 千位数....

        x = x / 10;

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

    for (i = lenth_of_x, j = 1, newNumber = 0; i >= 0; i--, j
*= 10)
    {
        num[i] *= j;

        newNumber += num[i];
    }

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

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

```
        getchar();

    }

}

int Digit(int x)

{

    int i, count;

    count = 0;

    i = 10;

    while (x / i)

    {

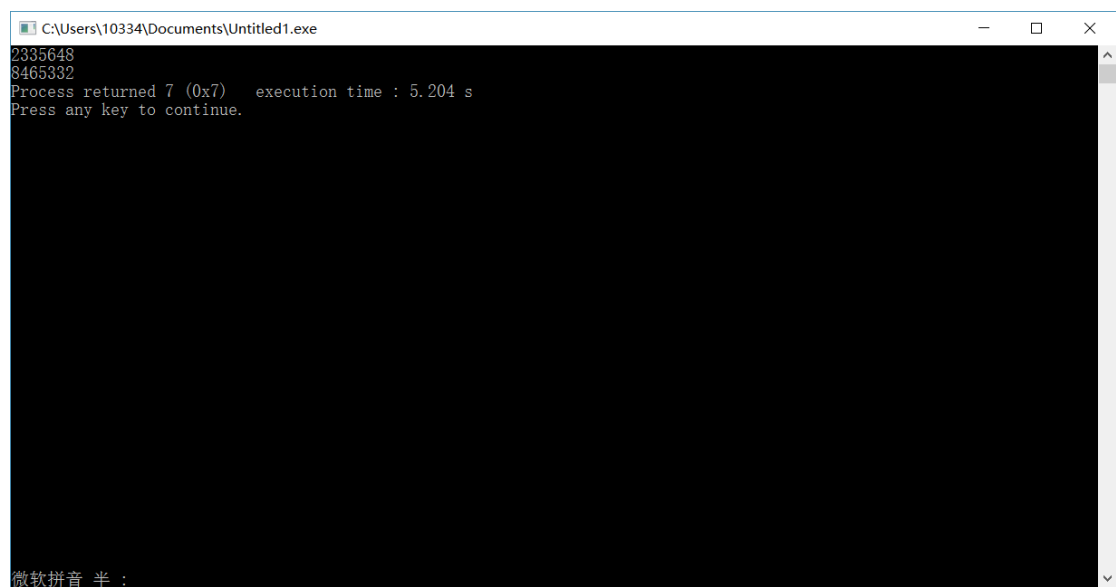
        count++;

        i *= 10;

    }//判断输入数字位数

    return count;

}
```



```
C:\Users\10334\Documents\Untitled1.exe
2335648
8465332
Process returned 7 (0x7)   execution time : 5.204 s
Press any key to continue.
```



(3). 假设工资税金按以下方法计算  $X \leq 1000$  元, 不收取税金。  
 $1000 < X < 2000$ , 收取 5% 的税金;  $2000 \leq X < 3000$ 。收取 10% 的税金;  
 $3000 \leq X < 4000$ 。收取 15% 的税金;  $4000 \leq X < 5000$ 。收取 20% 的税金;  
 $X \geq 5000$  收取 25% 的税金. 编写一个程序, 输入工资金额输出应收取税金额度。要求分别用 if 语句和 switch 语句来实现。

```
#include<stdio.h>

#define Tax_level1 0

#define Tax_level2 0.05

#define Tax_level3 0.10

#define Tax_level4 0.15

#define Tax_level5 0.20

#define Tax_level6 0.25

#define Tax_Boundary1 1000

#define Tax_Boundary2 2000

#define Tax_Boundary3 3000

#define Tax_Boundary4 4000

#define Tax_Boundary5 5000

#define TAX1 (Tax_Boundary2-Tax_Boundary1)*Tax_level2//1000-2000
的税

#define TAX2 (Tax_Boundary3-Tax_Boundary2)*Tax_level3+TAX1//1000-
3000的税

#define TAX3 (Tax_Boundary4-Tax_Boundary3)*Tax_level4+TAX2//1000-
4000的税

#define TAX4 (Tax_Boundary5-Tax_Boundary4)*Tax_level5+TAX3//1000-
```

## 5000的税

```
double If_Method(double wage);

double Switch_Method(double wage);

int main(void)
{
    double wage;

    scanf("%lf", &wage);

    while (wage != 0)
    {
        printf("if:%lf\n", If_Method(wage));

        printf("switch:%lf\n", Switch_Method(wage));

        scanf("%lf", &wage);
    }
}

double If_Method(double wage)
{
    double tax;

    if (wage <= Tax_Boundary1) tax = Tax_level1*wage;

    else if (wage <= Tax_Boundary2) tax = Tax_level2*(wage -
```

```
Tax_Boundary1);

    else if (wage <= Tax_Boundary3) tax = Tax_level3*(wage -
Tax_Boundary2) + TAX1;

    else if (wage <= Tax_Boundary4) tax = Tax_level4*(wage -
Tax_Boundary3) + TAX2;

    else if (wage <= Tax_Boundary5) tax = Tax_level5*(wage -
Tax_Boundary4) + TAX3;

    else tax = Tax_level6*(wage - 5000) + TAX4;

    return tax;
}

double Switch_Method(double wage)
{
    double tax;

    switch ((int)(wage - 1) / 1000)
    {

        case 0:tax = Tax_level1*wage; break;

        case 1:tax = Tax_level2*(wage - Tax_Boundary1); break;

        case 2:tax = Tax_level3*(wage - Tax_Boundary2) + TAX1; break;

        case 3:tax = Tax_level4*(wage - Tax_Boundary3) + TAX2; break;

        case 4:tax = Tax_level5*(wage - Tax_Boundary4) + TAX3; break;

        case 5:tax = Tax_level6*(wage - 5000) + TAX4; break;
```

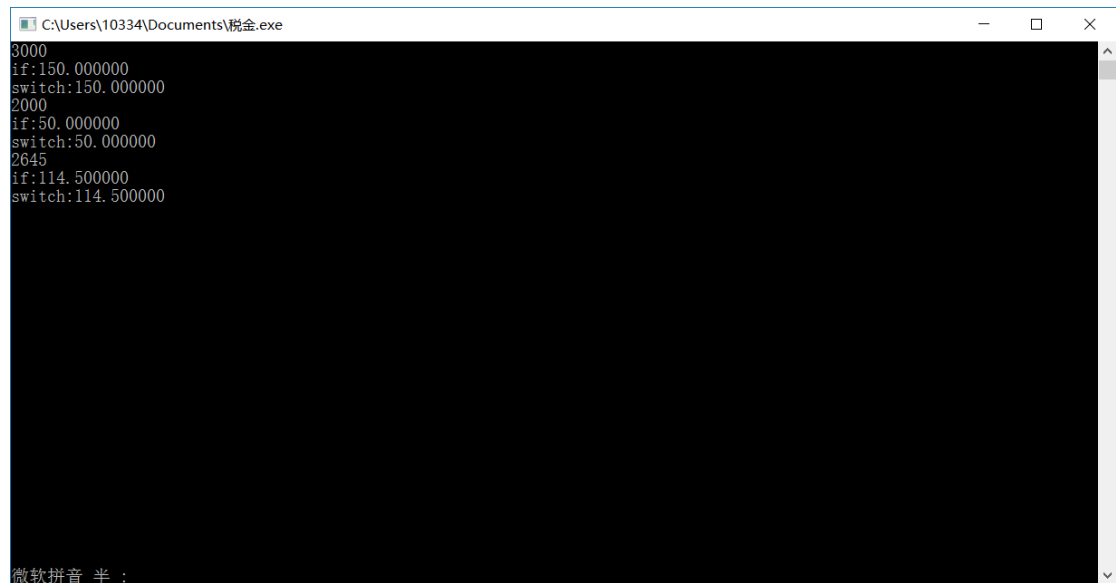
```
    default:

        break;

}

return tax;

}
```



```
C:\Users\10334\Documents\税金.exe
3000
if:150.000000
switch:150.000000
2000
if:50.000000
switch:50.000000
2645
if:114.500000
switch:114.500000
微软拼音 半:
```

(4) 编写一个程序，用牛顿迭代法求方程  $f(x)=3x^3-4x^2-5x+13=0$  满足精读 0.000001 的一个近似根，并在屏幕上输出所求近似根。

```
#include<stdio.h>

#include<math.h>

double FUNCTION_F(double x);

double Derivative_F(double x);

#define PRECISION 0.000001

int main(void)

{

    double x1, x2;
```

```
x1 = 1;

x2 = 1;

while (fabs(-FUNCTION_F(x1)/Derivative_F(x1))>PRECISION)

{

    x1 = x2;

    x2 = x1 - FUNCTION_F(x1) / Derivative_F(x1);

}

printf("%lf", x2);

system("pause");

}

double FUNCTION_F(double x)

{

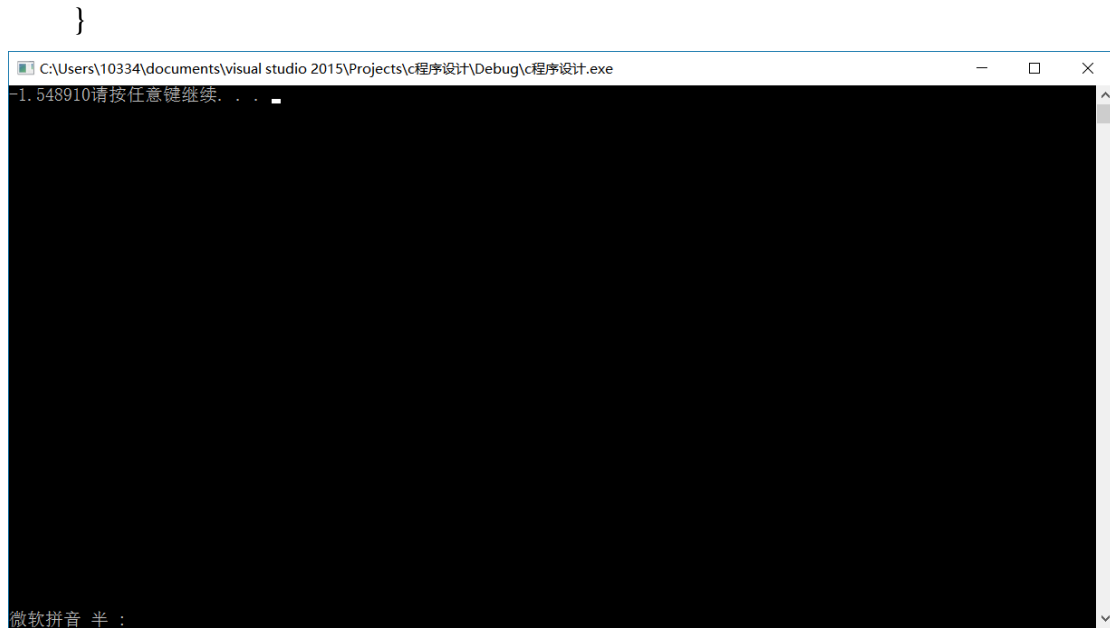
    return 3 * x*x*x - 4 * x*x - 5 * x + 13;

}

double Derivative_F(double x)

{

    return 9 * x*x - 8 * x - 5;
```



(5) 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。

```
/*空格处理*/

#include<stdio.h>

#include<string.h>

int main(void)

{

    int N;

    int i,j;

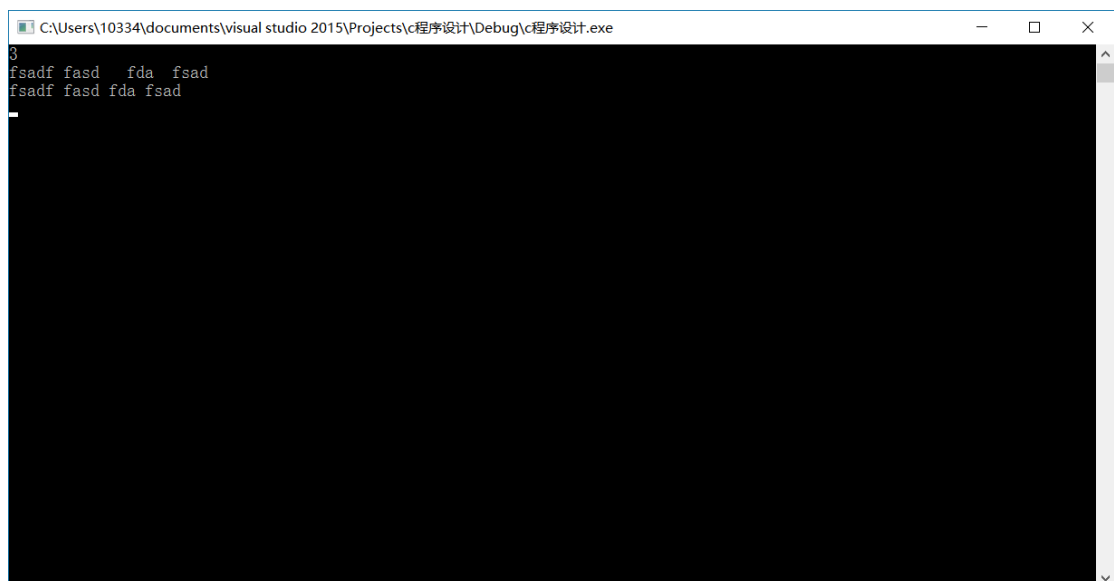
    char mystring[100];

    scanf("%d", &N);

    getchar();

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

```
{  
  
    fgets(mystring, 100, stdin);  
  
    for (j = 0; j < strlen(mystring); j++)  
    {  
  
        if (mystring[j] == ' ')  
        {  
  
            putchar(mystring[j]);  
  
            while (mystring[j] == ' ') j++;  
  
        }  
  
        putchar(mystring[j]);  
  
    }  
  
}
```



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe  
3  
fsadf fsad fda fsad  
fsadf fsad fda fsad
```

## 2.3 实验小结

编写 C 语言程序要熟练掌握复合语句，if 语句，switch 语句的使用，熟练掌握 for、while、do-while 基本的循环控制语句的使用，使程序更为流畅，掌握重复循环技术，了解转移语句与标号语句。编写条件语句时，要充分想到所有的情况，避免遗漏，是程序结果出现错误；编写循环语句时，要充分考虑循环结束时的条件，一面出错，使程序运行不了或出现死循环；在考虑循环结束的条件时，应尽量做一个最好的选择，让程序变得简洁，避免占用过长的篇幅，同时也可以减少错误的发生；要了解 for 语句、while 语句、do-while 语句之间以及 if 语句与 switch 语句之间的转换，使编写程序时有更大的选择空间，从而找到一个最优的方法。还要熟练掌握集成开发环境中的调试功能，例如单步执行，设置断点等等。以免因为修改程序中出现的错误而消耗过多的时间。



## 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  $s=1!+2!+3!+\cdots+n!$  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) 第1行, 没有声明函数sum\_fac(), 正确形式为:

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

2) 第15行, s+=fac应放入for循环内, 正确形式为:

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

{

    fac *= i;

    s += fac;

}
```

3) 第12行, fac未初始化, 正确的形式为:

```
Long fac=1;
```

(2) 错误修改后运行结果:

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

```
long sum_fac(int n);

void main(void)

{

    int k;

    for (k = 1; k<6; k++)

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

}

long sum_fac(int n)

{

    long s = 0;

    int i;

    long fac=1;

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

    {

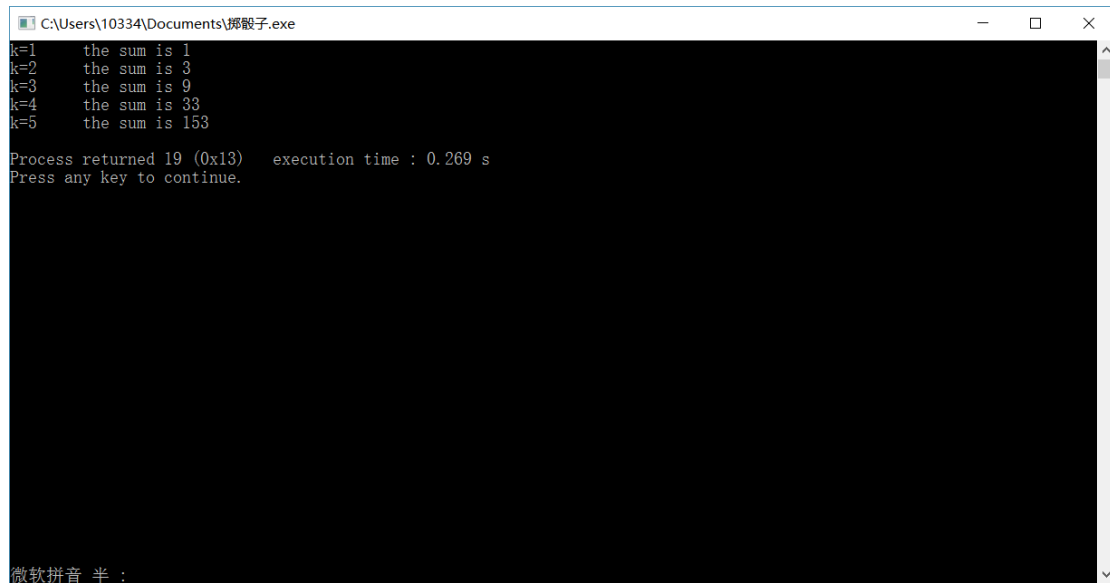
        fac *= i;

        s += fac;

    }

    return s;

}
```



```
C:\Users\10334\Documents\掷骰子.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

Process returned 19 (0x13)   execution time : 0.269 s
Press any key to continue.

微软拼音 半 :
```

### 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。

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

**解答：**

(1) 替换后的程序如下所示：

```
long sum_fac(int n)
{

    static long fac=1;

    fac *= n;//计算阶乘

    static long sum = 0;//存储阶乘和

    return sum+=fac;

}
```

(2) 替换后的程序如下所示：

```
#include<stdio.h>

double sum_fac(int n);

void main(void)

{

    int k;

    for (k = 1; k<6; k++)

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

    system("pause");

}

double sum_fac(int n)

{

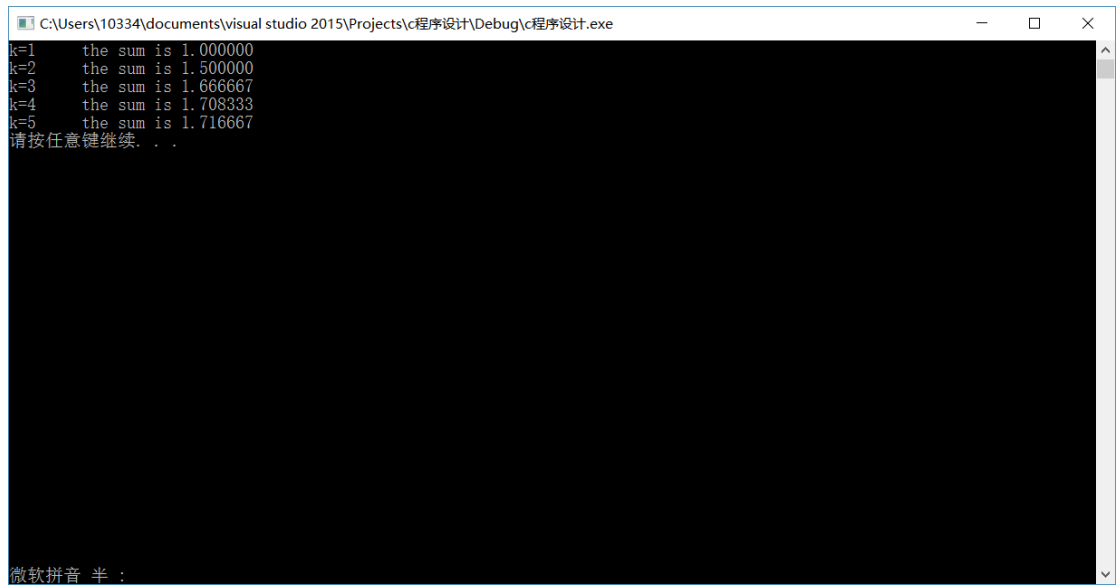
    static long fac = 1;

    fac *= n;//计算阶乘

    static double sum = 0;//存储阶乘和

    return sum += 1.0/fac;

}运行结果如下:
```



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
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
请按任意键继续. . .
微软拼音 半 :
```

### 3.2.3. 跟踪调试题

计算 fibonacci 数列前 n 项和的程序如下：

其中，`long sum=0,*p=&sum;`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）。

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

    long sum=0,*p=&sum;

    scanf("%d",&k);

    for(i=1;i<=k;i++){

        sum+=fibonacci(i);

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

    }

}

long fibonacci(int n)
```

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

单步执行程序，观察 p, i, sum, n 值。

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

解答：

P= 0x012ffc2c

I= -858993460

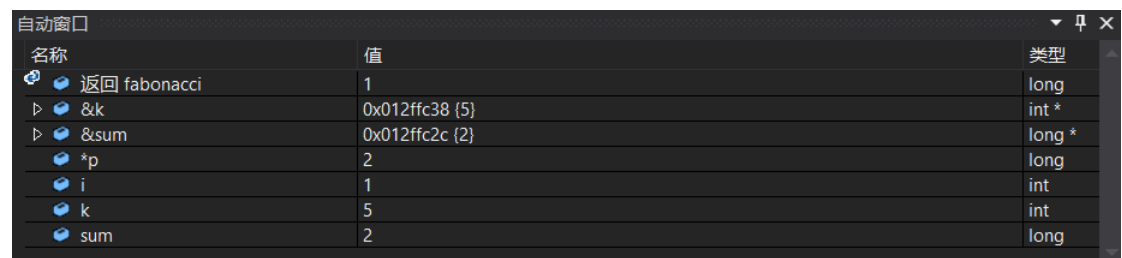
(2) 从 fibonacci 函数返回后光条停留在哪个语句上？

解答：

`sum+=fibonacci(i);`

(2) 进入 fibonacci 函数，watch 窗口显示的是什么？

解答：



名称	值	类型
返回 fibonacci	1	long
&k	0x012ffc38 {5}	int *
&sum	0x012ffc2c {2}	long *
*p	2	long
i	1	int
k	5	int
sum	2	long

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

解答：

3 变为 11604038

### 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) 解题思路：

运用辗转相除法，输入两个数  $x$ ,  $y$ ，通过计较大小保证  $y > x$ ，作为参数传递给 `max_commom_divisor()` 函数，函数模拟辗转相除法计算最大公约数。

2) 程序清单

```
#include<stdio.h>

int max_common_divisor(int y, int x);

int main(void)
{
    int x, y, temp;

    scanf("%d %d", &x, &y);

    while (x)
    {
        if (x > y)
        {
            temp = x;
            x = y;
            y = temp;
        }
    }
}
```



```
        } //保证y>=x

        printf("%d\n", max_common_divisor(y, x));

        scanf("%d %d", &x, &y);

    }

}

int max_common_divisor(int y, int x)

{

    int m;

    m = y%x;

    if (!m)

        return x;

    else return max_common_divisor(x, m);

}
```

### 3) 测试

#### a) 测试数据:

1 6

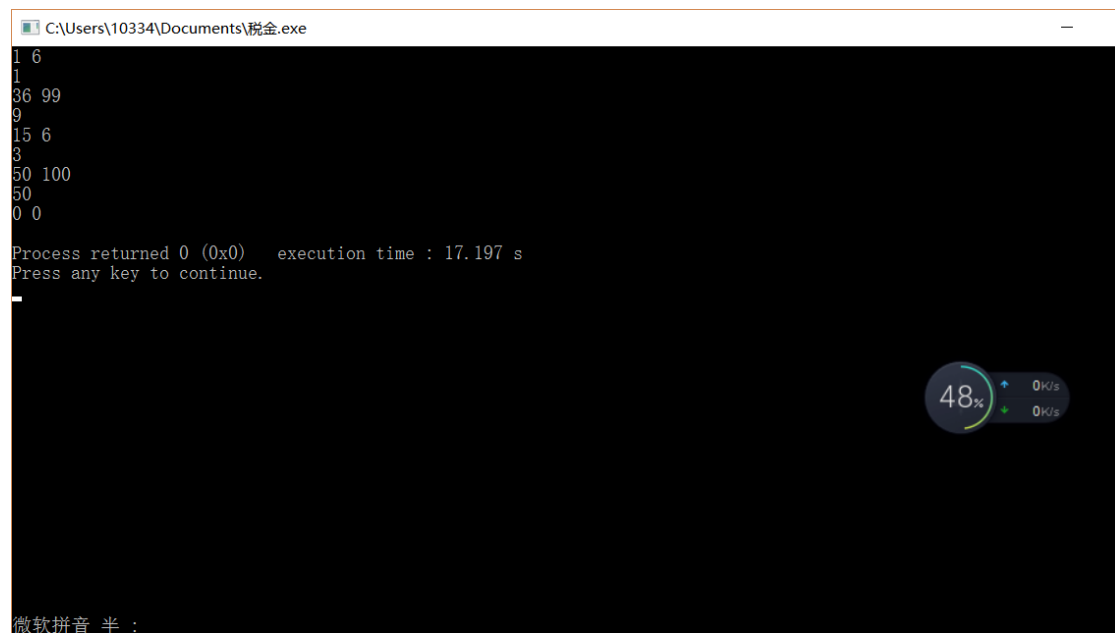
36 99

15 6

50 100

0 0 (退出)

## b) 测试结果:



```

C:\Users\10334\Documents>税金.exe
1 6
1
36 99
9
15 6
3
50 100
50
0 0
Process returned 0 (0x0)   execution time : 17.197 s
Press any key to continue.

```

(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。

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) 解题思路:

Isprime() 函数判断素数，Goldbach\_num() 验证哥德巴赫猜想，输入 begin 和 end，while 中判断 begin 是否为奇数，如果为奇数将其加 1，i 从 begin 到 end 依次加 2，且每一次判断 i 由两个素数之

和, Goldbach\_num() 函数从 2 遍历到参数 x 寻找素数, 当找到素数 i 时判断 x-i 是否为素数, 若为, 则输出, 否则继续寻找下一个。

## 2) 程序清单

```
#include<stdio.h>

#include<math.h>

int IsPrime(int n);

int Goldbach_num(int x);

int main(void)
{
    int begin, end, i;

    scanf("%d %d", &begin, &end);

    while (begin)
    {
        if (begin % 2)

            begin++;

        for (i = begin; i <= end; i += 2)
        {
            Goldbach_num(i);
        }

        putchar('\n');

        scanf("%d %d", &begin, &end);
    }
}
```

```
    }

}

int IsPrime(int n)

{

    int s, i;

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

    {

        s = n%i;

        if (s == 0)

            return 0;

    }

    return 1;

}

int Goldbach_num(int x)

{

    int j, k;

    for (j = 2; j <= x; j++)

    {

        if (IsPrime(j) && IsPrime(x - j))

        {

            printf("%d=%d+%d\n", x, j, x - j);

        }

    }

}
```

```
j = x + 1;  
  
}  
  
}  
  
}
```

### 3) 测试

#### a) 测试数据:

4 10

5 9

10 23

#### b) 测试结果:



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe  
4 10  
4=2*2  
6=3+3  
8=3+5  
10=3+7  
5 9  
6=3+3  
8=3+5  
10 23  
10=3+7  
12=5+7  
14=3+11  
16=3+13  
18=5+13  
20=3+17  
22=3+19
```

### 3.2.5. 选做 s 题

1、设 file1.c 如下:

```
#include <stdio.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 */

void main(void)

{

    x=10;

    y=20;

    ch=getchar();

    printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);

    func1();

}
```

file2.c 如下:

```
extern int x,y; /* 外部变量的引用性说明 */

extern char ch; /* 外部变量的引用性说明 */

void func1(void)

{

    x++;

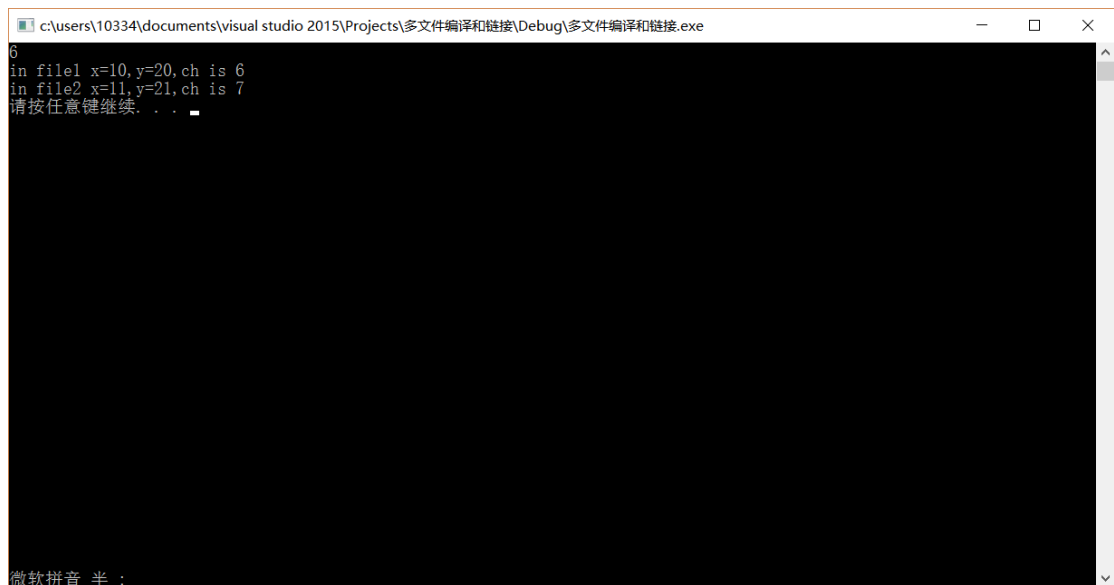
    y++;

    ch++;

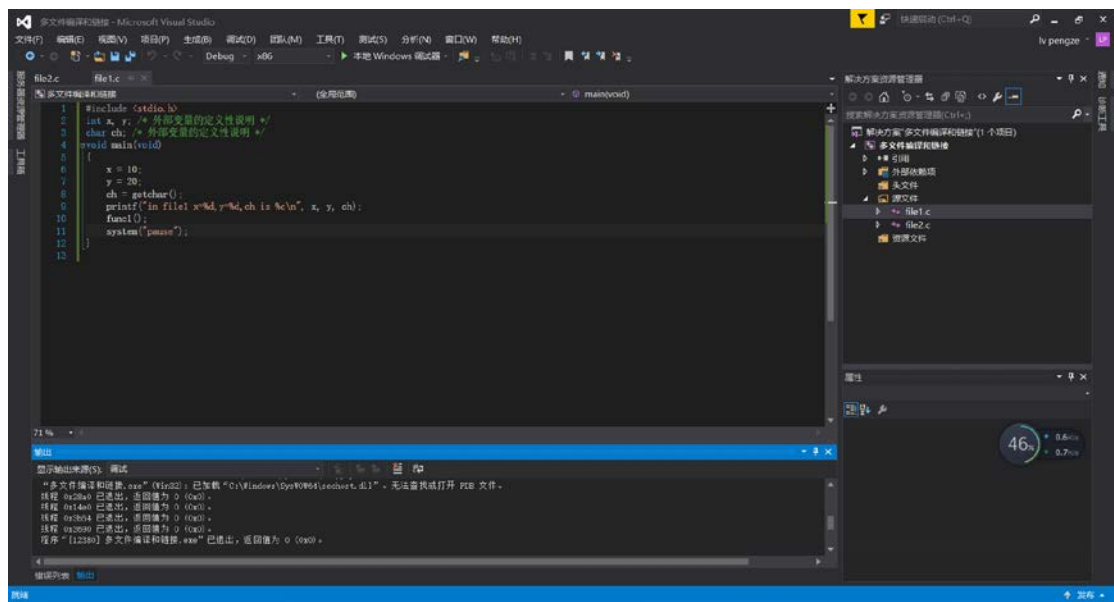
    printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}
```

试用 TCC 进行多文件编译和链接。然后在 DOS 环境下运行生成的可执行文件。



```
c:\users\10334\documents\visual studio 2015\Projects\多文件编译和链接\Debug\多文件编译和链接.exe
6
in file1 x=10,y=20,ch is 6
in file2 x=11,y=21,ch is 7
请按任意键继续. . .
```



### 3.3 实验小结：

通过本次实验，我掌握了分部调试，通过分步调试，我可以清楚的看到各变量的变化，有助于我充分理解程序以及修改程序错误。

其次，我掌握了静态变量的用法和多文件链接和编译。

验证哥德巴赫猜想这一题，可以看出计算机与数学问题的联系，可以通过计算机帮助人们解决许多难题。

此外，在编写程序时，一定要十分小心，尤其在循环语句的编写中，要注意循环继续和终止的条件。如果出现错误，要仔细的调试，一步一步的来，不能急

躁。在编写程序时，尽量使程序的计算量要小，这需要掌握好变量的种类运用。通过这次实验，我对程序调试的熟练度大大增加，了解了多文件程序的编译与链接。并且了解一些新的算法思想。这使我的思维角度大大拓宽。



## 4 编译预处理实验

### 4.1 实验目的

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

### 4.2 实验内容

#### 4.2.1. 源程序改错题

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);

}
```

**解答:**

## (1) 错误修改:

1) 第 2 行的宏定义未加括号, 正确形式为:

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

2) 第 3 行的宏定义未加括号, 正确形式为:

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

3) 第 5 行的 main 后未加括号, 正确的形式为:

```
void main()
```

4) 第 7 行未声明 a, 正确的形式为:

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

5) 第 9 行 %d 之间不能有逗号, 正确的形式为:

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

6) 第 10 行宏引用错误, 正确的形式为:

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

7) 第 4 行的 SWAP 出错, 不能交换两数, 正确的形式为:

```
#define SWAP(a,b) {t=b,b=a,a=t;}
```

(2) 错误修改后运行结果:



The screenshot shows a Windows command prompt window titled "C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe". The program prompts the user to "Input two integers a, b:" and the user enters "3 6". The program then outputs "SUM=9", "the difference between square of a and square of b is:-27", "Now a=6, b=3", and "请按任意键继续. . .". At the bottom left of the window, there is a small text "微软拼音 半;".

#### 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。

```
void main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

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

printf("\nthe maximum of them is %d\n",max(a,b,c));
```

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

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

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

**解答:**

1). 替换后的结果如下所示:

```
#include<stdio.h>

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

float sum(float x, float y);

void main(void)

{

    int a, b, c;

    float d, e;

    printf("Enter three integers:");

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

    printf("\nthe maximum of them is %d\n", max(a, b, c));


    printf("Enter two floating point numbers:");

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

    printf("\nthe sum of them is  %f\n", sum(d, e));

}


int max(int x, int y, int z)

{

    int t;

    if (x>y)
```

```
        t = x;

    else

        t = y;

    if (t<z)

        t = z;

    return t;

}
```

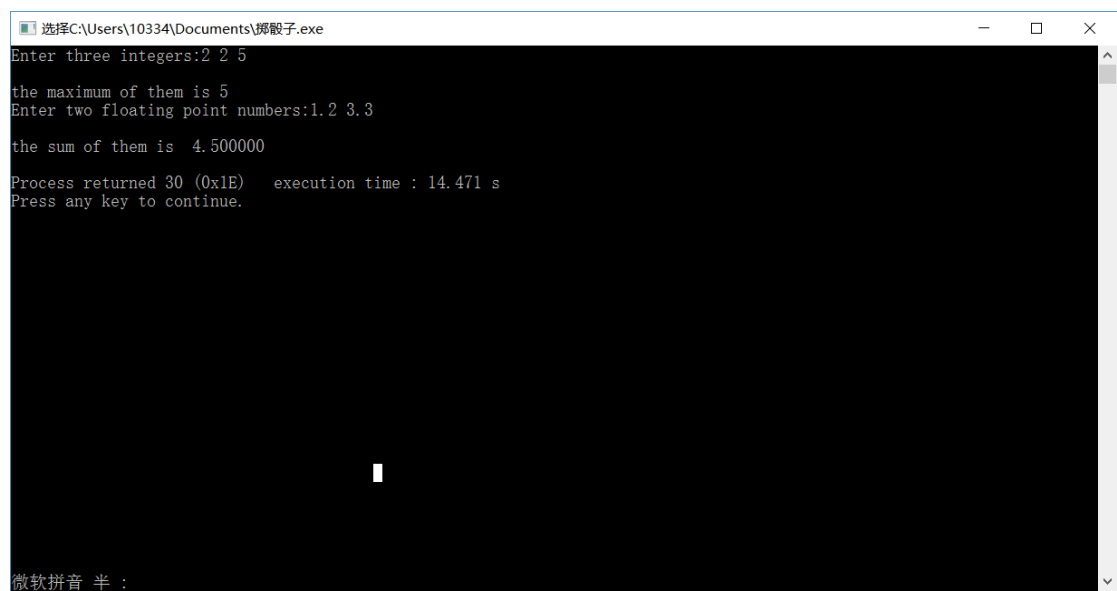
```
float sum(float x, float y)

{

    return x + y;

}
```

运行结果:



```
选择C:\Users\10334\Documents\掷骰子.exe
Enter three integers:2 2 5
the maximum of them is 5
Enter two floating point numbers:1.2 3.3
the sum of them is 4.500000
Process returned 30 (0x1E) execution time : 14.471 s
Press any key to continue.
```

2) . #define max(x, y, z) x>y?(x>z?x:z):(y>z?y:z)

### 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。

```
#define R

void main(void)

{

float r, s;

int s_integer=0;

    printf ("input a number: ");

    scanf ("%f",&r);

    #ifdef R

        s=3.14159*r*r;

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

        s_integer= integer_fraction(s);

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

        assert((s-s_integer)<1.0);

    #endif

}

int integer_fraction(float x)

{

    int i=x;

    return i;
```

```
}
```

1) 修改程序，使程序编译通过且能运行；

a) 修改后的结果为：

```
#include<stdio.h>

#include<assert.h>

#define R

int integer_fraction(float x);

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);

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

        assert((s - s_integer)<1.0);

    #endif

    system("pause");

}
```



```
int integer_fraction(float x)

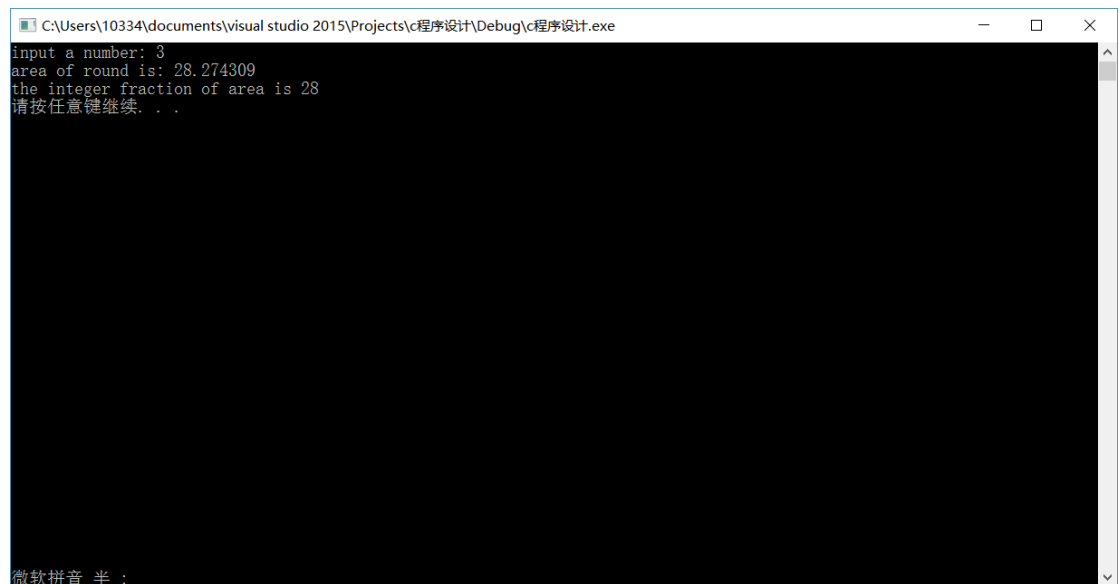
{

    int i = x;

    return i;

}
```

b) 运行结果:



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
input a number: 3
area of round is: 28.274309
the integer fraction of area is 28
请按任意键继续. . .
微软拼音 半:
```

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?

`X= 28.2743092`

`I=28`

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`。

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

```
#include<assert.h>

#define R

int integer_fraction(float x);

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);

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

    assert((s - s_integer)<1.0);

#endif

    system("pause");
}

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

```
    return i;  
  
}
```

#### 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) 解题思路:

1. 宏定义 s 和 area。
2. Main() 键盘输入并读取 a, b, c。
3. 输出 area。

2) 程序清单:

```
1.  #include<stdio.h>  
2.  #include<math.h>  
3.  #define s(a,b,c) (a+b+c)/2  
4.  #define area(a,b,c,s) sqrt(s*(s-a)*(s-b)*(s-c))  
5.  int main(void)  
6.  {  
7.      int a, b, c;  
8.      while (scanf("%d %d %d", &a, &b, &c) != EOF)  
9.      {
```

```
10.          printf("%d %lf\n", s(a, b, c), area(a, b,  
c, s(a, b, c)));
```

```
11.          }
```

```
12. }
```

3) 测试:

1. 测试数据:

i. 3 4 5

ii. 6 6 6

iii. 7 12 13

2. 测试结果:



The screenshot shows a Windows command prompt window titled "C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe". The window displays the output of a C program. The first three lines of output are "3 4 5", "6 6.000000", and "6 6 6", which correspond to the test data provided in the text. The next three lines are "9 15.588457", "7 12 13", and "16 41.569219", which are the results of the program's calculations for the given inputs. The window has a standard Windows title bar with minimize, maximize, and close buttons.

(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 则输出变换后的文字,若 #define CHANGE 0 则原文输出。

**解答:**

1) 解题思路:

1. Define 定义 CHANGE, 使用 `#ifdef CHANGE 1` 和 `#endif` 来判断是否进行变换
2. 使用 `fgets()` 读取电报文字, 如果 `#define CHANGE 1` 则进行处理
  - i. 遍历字符串, 使用 `ctype.h` 中的 `isupper()` 和 `tolower()`, `toupper()` 更换字符串中的大小写。
3. 使用 `fputs()` 输出字符串。

2) 程序清单:

```
1.  #include<stdio.h>
2.  #include<string.h>
3.  #include<ctype.h>
4.  #define CHANGE 1//或者换为 0
5.  int main(void)
6.  {
7.      char c;
8.      char str[1000];
9.      while (fgets(str, 1000, stdin))
10.     {
11.         #ifdef CHANGE 1
12.             {
13.                 int i;
14.                 for (i = 0; i < strlen(str); i++)
```

```
15.      {
16.          if (isupper(str[i]))
17.              str[i] = tolower(str[i]);
18.          else str[i] = toupper(str[i]);
19.      }
20.
21.  }
22.      fputs(str, stdout);
23. #endif
24.  }
25.      return 0;
26. }
```

### 3) 测试:

#### 1. 测试数据;

##### i. #define CHANGE 1

```
1. 0j$MmKmKgFee:rOiN2vCuSHB94?Ir' XK)b#1$>"B5]?[
@ZN:
2. uQ;v$2?, $dt>. u$f]*5co?wVee>w\4) fJa7SEbMI) 9oD
!iYNm0=?3g>Xmo) 664
```

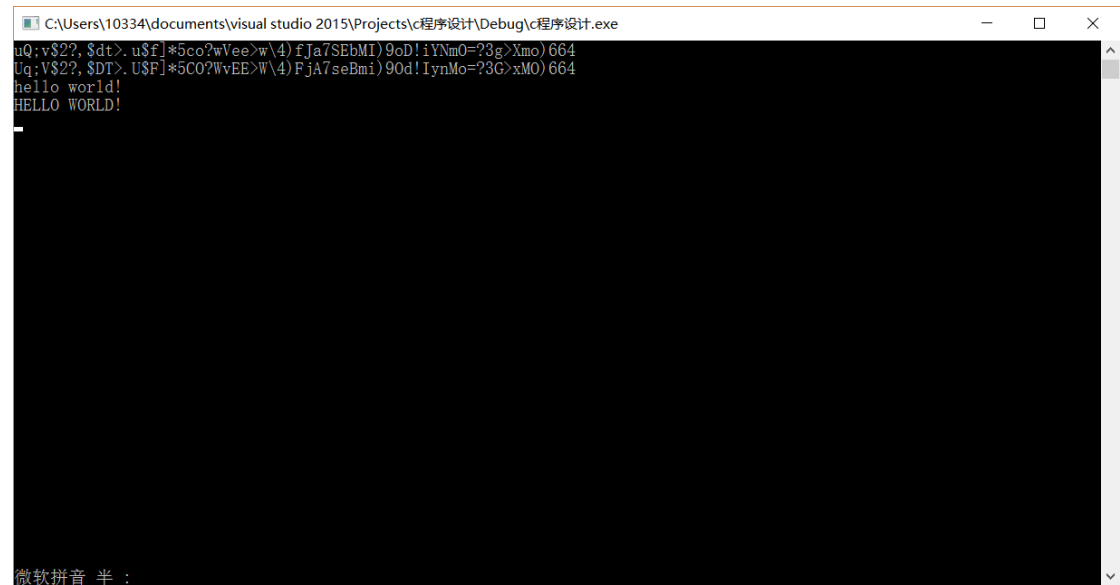
##### ii. #define CHANGE 0

```
1. 0j$MmKmKgFee:rOiN2vCuSHB94?Ir' XK)b#1$>"B5]?[
@ZN:
2. uQ;v$2?, $dt>. u$f]*5co?wVee>w\4) fJa7SEbMI) 9oD
```

```
!iYNm0=?3g>Xmo)664
```

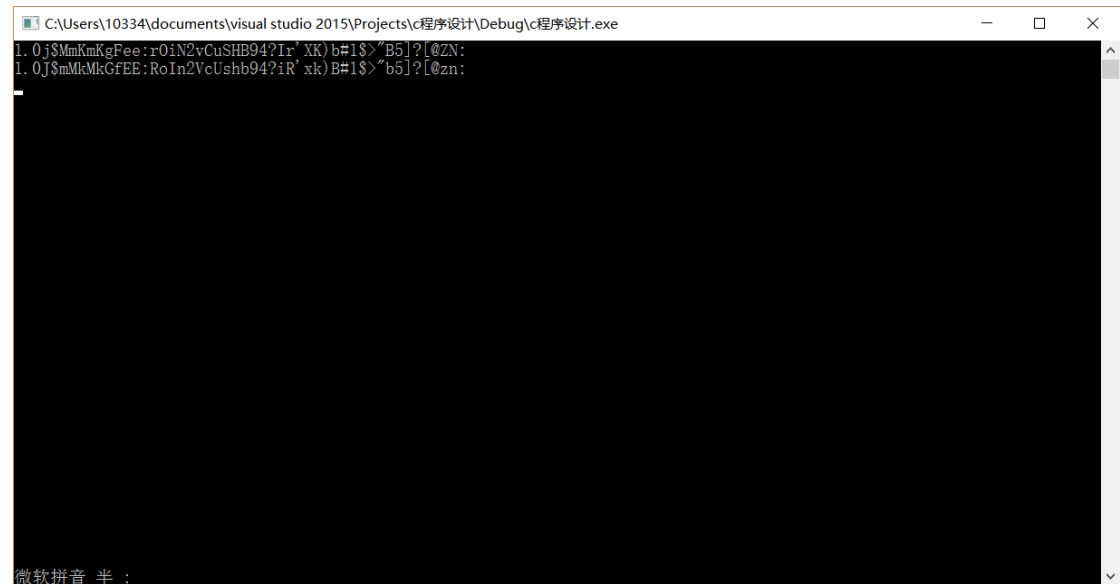
## 2. 测试结果:

### i. #define CHANGE 1



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
uQ;v$2?, $dt>. u$f]*bco?wVee>w\4) fJa/SEbMI)9oD!iYNm0=?3g>Xmo)664
Uq;V$2?, $DT>. U$F]*5C0?WvEE>W\4) Fja7seBmi)9oD!IynMo=?3G>xMO)664
hello world!
HELLO WORLD!
```

### ii. #define CHANGE 0



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
1. 0j$MmKmgFee:r0iN2vCuSHB94?Ir' xk) b#1$>'B5]?[@ZN:
1. 0j$MmKmgFEE:RoIn2VcUshb94?iR' xk) B#1$>'b5]?[@zn:
```

## 4.3 实验小结

通过这次试验,我掌握了条件编译的用法以及带参数的宏的用法,通过使用条件编译,尤其是在程序较大时可以节省许多内存空间,十分有效。而宏定义可以简化程序书写以及程序的维护。

通过实验我也发现了自己不少的问题,这都是只看书上的程序而没有自己亲身上机编写程序而无法得知的:

1. 使用`#define` 代替数学公式时要加上括号, 否则会出现错误
2. 条件编译除了`#ifdef` 外还有`#if` 指令, `#ifndef` 指令, 配套使用的有`#elif`, `#else`, 并且要以`#endif` 结束
3. 与`#ifdef` 类似的一个运算符是 `defined`(标识符)或 `defined` 标识符, 如果被定义, 值 1, 否则为 0
4. 巧妙运用 c 库函数可以节省许多时间, 比如电报那一题使用 `ctype.h` 中提供的 `isupper()`, `islower()` 来判断大小写, 使用 `toupper()`, `tolower()` 来转换大小写。



## 5 数组实验

### 5.1 实验目的

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

### 5.2 实验内容

#### 5.2.1 源程序改错

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

源程序

```
1 #include<stdio.h>

2 int main(void)

3 {

4     int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};

5     void sort(int [],int);

6     int i;

7     sort(a[0],10);

8     for(i = 0; i < 10; i++)

9         printf("%6d",a[i]);
```

```
10    printf("\n");

11    return 0;

12 }

13 void sort(int b[], int n)

14 {

15     int i, j, t;

16     for (i = 0; i < n - 1; i++)

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

18             if(b[j] < b[j+1])

19                 t = b[j], b[j] = b[j+1], b[j+1] = t;

20 }
```

**解答：**

1) 错误修改：

- 1) 第 7 行 sort()函数第第一个参数错误，参数应为地址，a[0]是一个数，正确的形式为：sort(a, 10);
- 2) 第 18 行，语义错误，题目中的意思是降序排序，正确顶形式为：if(b[j] > b[j+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? (b[M - i] = a[j - 1]) : (b[M-i]=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. `#include<stdio.h>`
2. `#define M 10`
3. `#define N 3`
4. `int main(void)`
5. `{`
6. `int a[M], b[M];`
7. `int i, j, k;`
8. `for (i = 0; i < M; i++)`
9. `a[i] = i + 1;`
10. `for (i = M, j = 0; i > 1; i--) {`
11. `for (k = 1; k <= N; k++)`

```
12. {  
  
13. while (!a[j])  
  
    i.    ++j;  
  
14. if (j++ >= M - 1) j = 0;  
  
15. }  
  
16. if (j)  
  
17. {  
  
18. b[M - i] = a[j - 1];  
  
19. a[j - 1] = 0;  
  
20. }  
  
21. else  
  
22. {  
  
23. int t;  
  
24. for (t = M - 1; a[t] == 0; t--);  
  
25. b[M - i] = a[t];  
  
26. a[t] = 0;  
  
27. }  
  
28. }  
  
29. for (i = 0; i < M - 1; i++)  
  
30. printf("%6d", b[i]);  
  
31. for (i = 0; i < 10; i++)  
  
32. if (a[i] != 0)
```

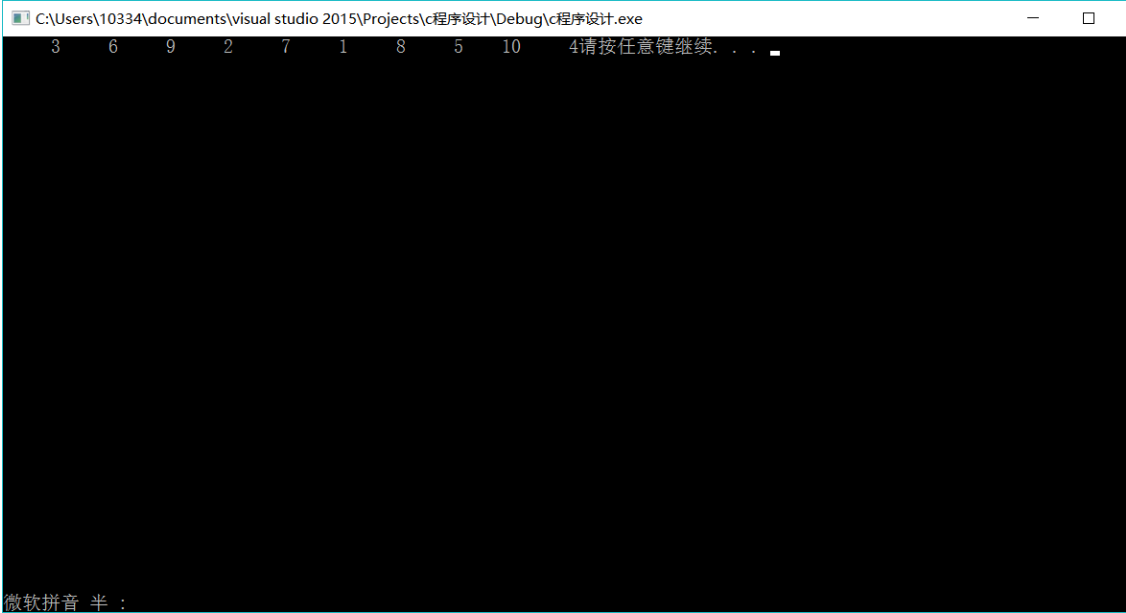
```
33. printf("%6d", a[i]);
```

```
34. system("pause");
```

```
35. return 0;
```

```
36. }
```

ii. 运行结果：



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
3 6 9 2 7 1 8 5 10 4请按任意键继续...
```

1. 微软拼音 半：

### 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) {

    i.   char a[50]="The adopted symbol is
        ",b[27]="abcdefghijklmnopqrstuvwxyz";

    ii.  strncat(a, b, 4);

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

    i.   return 0;

6) }

7) void strncat(char s[],char t[], int n)

8) {

    i.   int i = 0, j;

9) while(s[i++]) ;

10) for(j = 0; j < n && t[j];)

11) s[i++] = t[j++];

    i.   s[i] = '\0';

12) }
```

**解答:**

- 1) i=23; s=0x00bcf940 ,t=0x00bcf91c
- 2) 第 10 行改为 for(j = 0,i--; j < n && t[j];), 此时 a[i]的值为'\0',之后拼接是将空字符覆盖。

### 5.2.4 程序设计

编写并上机调试运行能实现以下功能的程序。

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

- 1) 解体思路:利用数组存储矩阵,  $a[i][j]$ , 那么他的反置矩阵就是  $a[j][i]$ , 利用嵌套 for 循环读取、交换和输出数组。
- 2) 程序清单:

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

2. int main(void)

3. {

4.     int matrix[3][4];

5.     int anti_matrix[4][3];

6.     int i, j, k;

7.     for (i = 0; i < 3; i++)
```



```
8.  for (j = 0; j < 4; j++)

9.  scanf(" %d", &matrix[i][j]);

10. for (i = 0; i < 3; i++)

11. for (j = 0; j < 4; j++)

12. anti_matrix[j][i] = matrix[i][j];

13. for (i = 0; i < 3; i++)

14. {

15. for (j = 0; j < 4; j++)

16. printf("%5d", matrix[i][j]);

17. putchar('\n');

18. }

19. putchar('\n');

20. for (i = 0; i < 4; i++)

21. {

22. for (j = 0; j < 3; j++)

23. printf("%5d", anti_matrix[i][j]);

24. putchar('\n');

25. }

26. system("pause");

27. }
```

3) 运行结果:

```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\D
1 2 3 4
5 6 7 8
9 10 11 12
    1    2    3    4
    5    6    7    8
    9   10   11   12

    1    5    9
    2    6   10
    3    7   11
    4    8   12
请按任意键继续. . .
```

(2).

1) 解题思路: 使用字符数组存储字符, 利用 for 循环, 并且每移位一次读取并输出输入数的末位。

2) 程序清单:

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

2. #include<string.h>

3. int main(void)

4. {

5.     int i,N,j,num,mask=1<<31;

6.     char bit[30][33];

7.     scanf("%d", &N);
```

```

8.   for (i = 0; i < N; i++)
9.   {
10.  bit[i][32] = '\0';
11.  scanf("%d", &num);
12.  for (j = 0; j < 32; j++)
13.  {
14.  bit[i][j] = (num < 0 ? '1' : '0');
15.  num = num << 1;
16.  }
17.  }
18.  for (i = 0; i < N; i++)
19.  puts(bit[i]);
20.  system("pause");
21.  }

```

3) 运 行 结 果 :

[illegible]

## (3)

- 1) 解题思路: name 数组存储姓名, grade 数组存储成绩, 采用选择交换法进行排序, for 循环输出结果, 使用二分查找寻找成绩, 将查找结果存储在 search\_result 数组中, 找到 BinarySearch()函数为二分查找函数, 找到返回数组的下标, 未找到返回-1.使用 for 循环和 if 判断输出结果。

- 2) 程序清单: #include<stdio.h>

```
#include<string.h>

#define FD 100//第一维度

#define SD 20//第二维度

int Binary_Search(int key, int a[], int n);//二分法查找, key
为查询目标, a[]查找的范围, n 为 a[]的数组大小, 找到返回 a[]的
角标, 找不到返回-1.

int main(void)
{
    char name[FD][SD];

    int grade[FD];

    int i, j, n;//i, j 控制循环

    scanf("%d", &n);//接下来有 n 组数据

    getchar();

    /*读取姓名和成绩*/

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

```
{

    scanf(" %s", name[i]);

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

    getchar();

}

/*选择交换法排序*/

{

    int i, j, k;

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

    {

        k = i;

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

        {

            if (grade[k] < grade[j])

                k = j;

        }

        if (i != k)

        {

            int temp;

            temp = grade[i];

            grade[i] = grade[k];
```

```
        grade[k] = temp;//成绩排序

        char c[SD];

        strcpy(c, name[i]);

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

        strcpy(name[k], c);//姓名排序

    }

}

/*输出排序后的结果*/

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

    //printf("%d\n", grade[j]);

    printf("%-20s %d\n", name[j], grade[j]);

putchar('\n');

/*查询*/

int N, search_result[FD];//进行 N 组查询，将查询的结果存在
search_result[]中。

{

    scanf("%d", &N);

    getchar();

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

    {

        int key;//查询的目标数
```

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

search_result[j]=Binary_Search(key, grade, n); //储存
查询结果。

//printf("*****%d\n", search_result[j]);

}

}

/*输出查询结果*/

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

{

    if (search_result[j] >= 0)

        printf("%-20s    %d\n",    name[search_result[j]],
grade[search_result[j]]);

    else printf("Not found!\n");

}


//system("pause");

}

int Binary_Search(int key, int a[], int n)

{

    int front = 0, back = n - 1, mid;

    while (back >= front)

    {

        mid = (front + back) / 2;
```

```
        if (a[mid] > key)

            front = mid + 1;

        else if (a[mid] < key)

            back = mid - 1;

        else return mid;

    }

    return -1;

}
```

### 3) 测试:

#### i. 测试数据:

5

ZhangChuanChao 88

XiaoHong 95

XiaoMing 90

LiSi 100

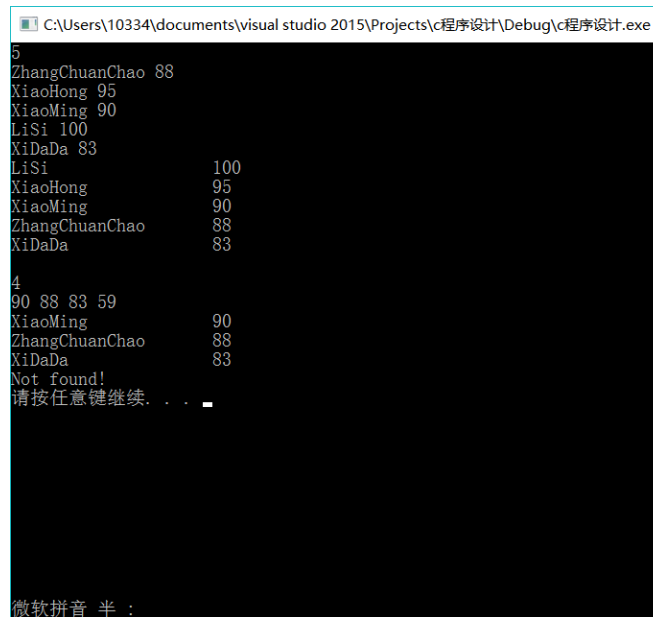
XiDaDa 83

4

90 88 83 59

#### ii. 测试结果:





```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
5
ZhangChuanChao 88
XiaoHong 95
XiaoMing 90
LiSi 100
XiDaDa 83
LiSi 100
XiaoHong 95
XiaoMing 90
ZhangChuanChao 88
XiDaDa 83
4
90 88 83 59
XiaoMing 90
ZhangChuanChao 88
XiDaDa 83
Not found!
请按任意键继续. . .
```

### 5.3 实验小结：

本次实验使我初步掌握了掌握数组的说明、初始化和使用以及数组与指针的关系，可以通过一维数组作为函数参数进行函数之间的数据交换，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，还学会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。

在实验过程中我也遇到了一些问题，对二维数组的理解不够透彻，比如在成绩处理的实验中，将二维数组作为参数传递给另一个函数的用法就没有很好的掌握，最后，通过百度查询和自行查阅资料（c 语言与程序设计，C primer plus）掌握了相关用法，在传递二维函数时，要将第二个维度明确下来，比如：

要处理 char 型数组 c[100][20]，对函数的声明可以是 L function(char a[][20],int n) 或者 L function(char (\* a)[20],int n)，引用时：function(c,100)，可见，数组与指针也有密切的联系。

## 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：

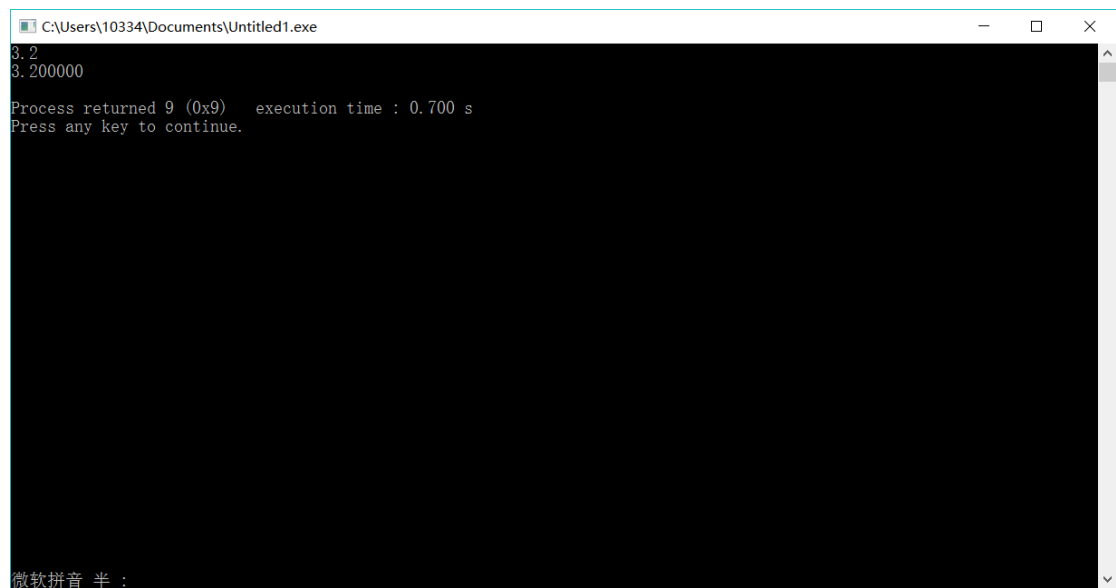
- 1) 第 4 行使用了未初始化的局部变量“p”，正确形式为：

```
float *p;
```

```
float q;
```

```
p = &q;
```

(2) 错误修改后运行结果：



```
C:\Users\10334\Documents\Untitled1.exe
3.2
3.200000
Process returned 9 (0x9) execution time : 0.700 s
Press any key to continue.
微软拼音 半:
```

### 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;

gets(a);_____

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

i=0;

gets(b);_____

result=___p___(a,b);

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

down:

```
;  
  
}
```

运行结果:

```
C:\Users\10334\Documents\Untitled1.exe  
1 copy string.  
2 connect string.  
3 exit.  
input a number (1-3) please!  
2  
input the first string please!  
adfaad hdj  
input the second string please!  
uiuto hj  
the result is adfaad hdjuiuto hj  
Process returned 33 (0x21) execution time : 13.948 s  
Press any key to continue.
```

(2) 为了使程序不受 scanf、getchar、gets 等函数输入后回车符的影响，请修改第 (1) 题程序，按要求输出下面结果：((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): 原程序清单:

```
#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;

    gets(a);

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

    i = 0;

    gets(b);

    result = p(a, b);

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

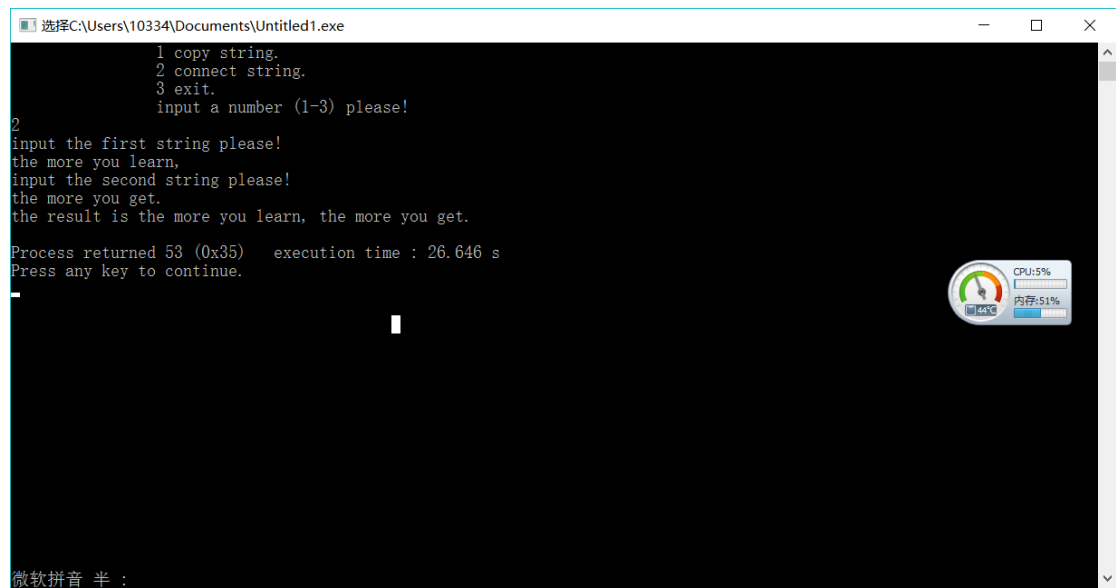
down:

    ;

}
```

(2): 运行结果:





```
1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!
2
input the first string please!
the more you learn,
input the second string please!
the more you get.
the result is the more you learn, the more you get.
Process returned 53 (0x35)   execution time : 26.646 s
Press any key to continue.
```

### 6.2.3. 跟踪调试题

```
#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);
}
```

(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.

(3) 选做：由于 watch 窗口中只显示 s 所指串的值，不显示 s 中存储的地址值，怎样才能观察到 s 值的变化呢？

**解答：**

(1)：进入时，s=0x004ff7a4；返回 main 时，s=0x004ff7c1。

(2)：

1)：程序清单：

```
#include "stdio.h"

char *strcpy(char *, char *);

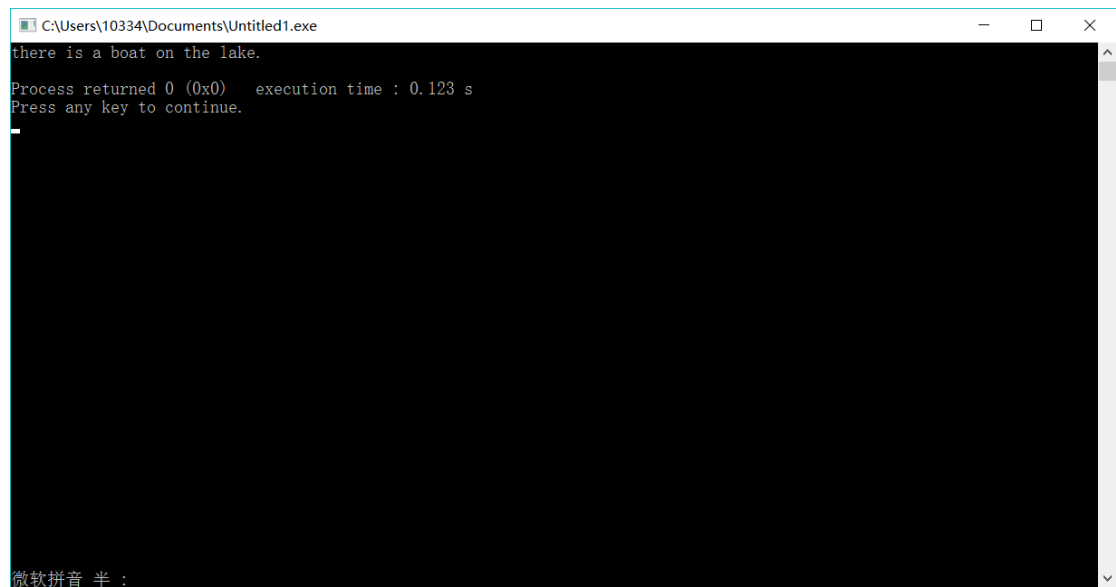
void main(void)
{
    char a[20], b[60] = "there is a boat on the lake.";
    printf("%s\n", strcpy(a, b));
}

char *strcpy(char *s, char *t)
{
    char * p = s;

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

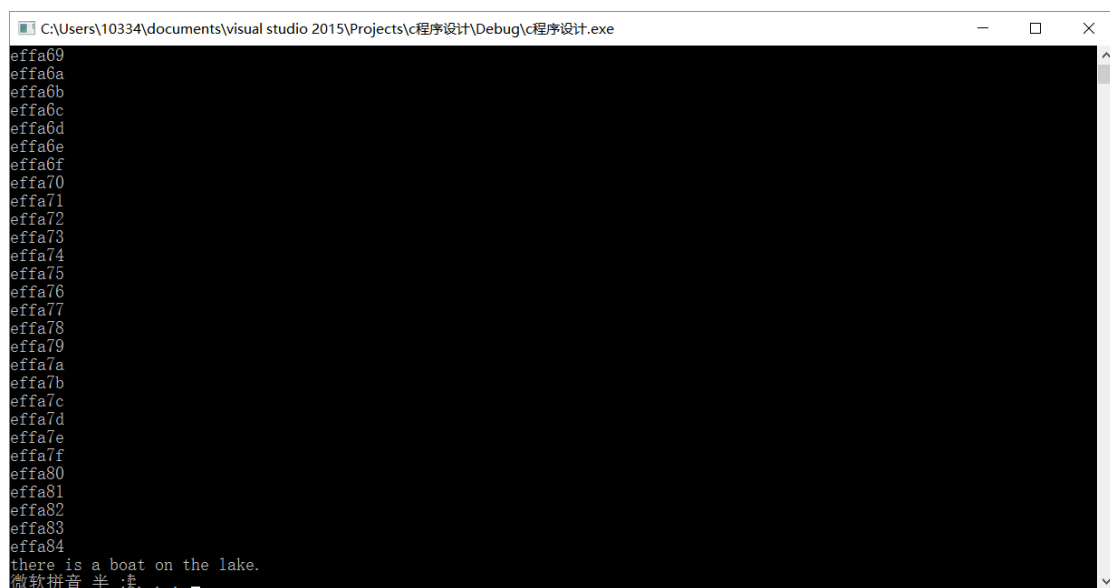
```
    return (p);  
  
}
```

2): 运行结果:



```
C:\Users\10334\Documents\Untitled1.exe  
there is a boat on the lake.  
Process returned 0 (0x0) execution time : 0.123 s  
Press any key to continue.  
微软拼音 半 :
```

(3) 使用 printf 将 s 的地址打印出来。Printf(“%x”,s), 运行结果:



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe  
effa69  
effa6a  
effa6b  
effa6c  
effa6d  
effa6e  
effa6f  
effa70  
effa71  
effa72  
effa73  
effa74  
effa75  
effa76  
effa77  
effa78  
effa79  
effa7a  
effa7b  
effa7c  
effa7d  
effa7e  
effa7f  
effa80  
effa81  
effa82  
effa83  
effa84  
there is a boat on the lake.  
微软拼音 半 : . . .
```

#### 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) 解题思路：

- a) scanf 读取输入的数，记为 num
- b) 构造最高字节的高 4 位为 1 的长整型逻辑尺(0xf000 0000);
- c) Num 和 mask 使用&运算符运算可以取出高字节的高 4 位，存储
- d) 将取出来高 4 位右移 28 位得到目标数，进行输出。
- e) 将 num 左移 4 位重新赋值给 num，重复 b) 步骤。
- f) 以上循环 8 次，将 32 位的 long 型按每 4 位依次取出。

2) 程序清单：

```
#include<stdio.h>

int main(void)
{
    long num;

    scanf("%d", &num);

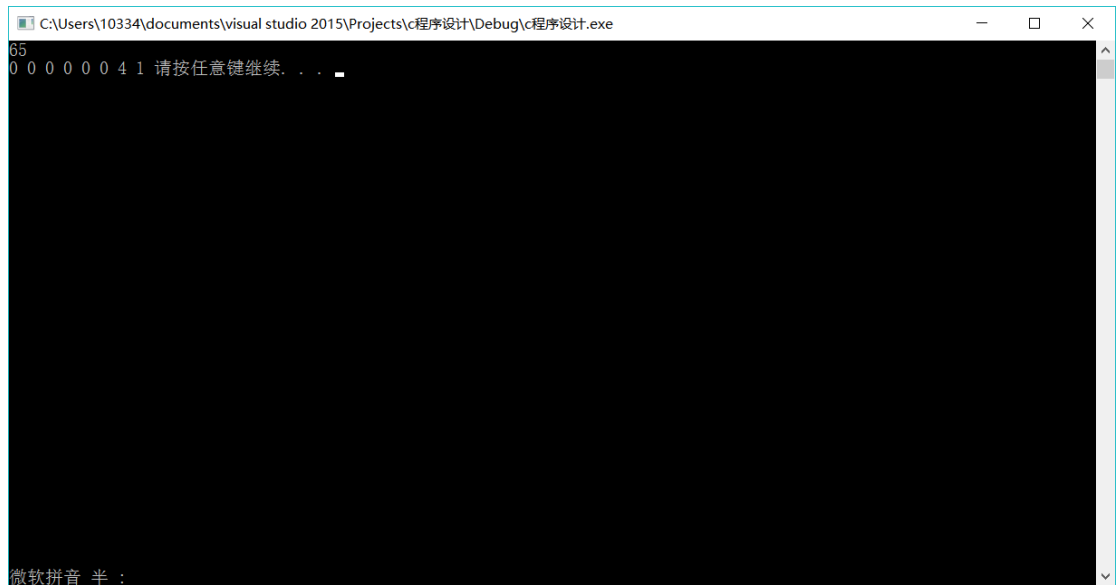
    long mask = 0xf0000000;

    for (int i = 1; i <=8; i++)
    {
        printf("%d ", (mask&num) >> 28);

        num = num << 4;
    }
}
```

```
system("pause");  
  
}
```

### 3) 运行结果:



(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) 解题思路:

- a) 创建指针数组 `*p[100]`, 用来保存输入的数据。
- b)  $N$  代表有  $n$  行输出，当  $n$  为 0 时退出循环。
- c) 对于一给定  $n$ ，使用 `for` 循环处理  $n$  行输入，每循环一次为 `p[i]` 分配 80 个字节的空间（每行输入不超过 80 个）
- d) 将 `fgets` 读取的内容存入 `p[i]` 指向的缓冲区中，将字符串的首地址赋值给 `p[i]`

e) for 循环 n 次处理 n 行输入

f) 再次 for 循环 n 次输出处理后的数据，处理数据使用 `space_trim()` 函数。

g) 函数接受存储字符串的地址，函数中定义两个指针 `pt1`, `pt2`，均指向接收的字符串，使用 `while(*pt1)` 遍历字符串，`*pt2++=*pt1++` 将 `pt1` 指向的值赋值给 `*pt2`，并且使用 `if` 语句判断 `*pt1` 是否为空格，如果是的话使用 `while(*pt1==' ')` 跳过剩下的空格。

h) 输出之后使用 `free()` 释放已经分配的动态存储空间。

2) 程序清单：

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX_STR 80

char * space_trim(char * c);

int main(void)

{

    char * p[100]; //创建指针数组，共 100 个

    int n, i;

    scanf("%d", &n);

    getchar();

    while (n) //当 n 为 0 时结束程序

    {

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

```
{

    p[i] = (char *)malloc(MAX_STR * sizeof(char)); //为指针指派 MAX_STR 个字节的缓冲区

    fgets(p[i], MAX_STR, stdin); //将 fgets 读取的内容存入 p[i] 指向的缓冲区中，将字符串的首地址赋值给 p[i]

}

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

{

    printf("%s", space_trim(p[i])); //输出处理后的字符串

    free(p[i]);

}

putchar('\n');

scanf("%d", &n);

getchar();

}

system("pause");

}

char * space_trim(char * c)

{

    char * pt = c;

    char * first = c;

    int i, out=1;
```

```
for (i = 0; i < strlen(c)-1&&out; i++)

    if (*(pt + i) != ' ')

        out=0;//如果字符串 c 从头到尾不全是空格或者换行的话就
处理，c 置为空串；
```

```
    if (out) *c = '\0';

    //printf("****%d\n",out);

    /*空格处理*/

    while (*pt!='\0')

    {

        if (*pt == ' ')

        {

            *c++ = *pt++;

            while (*pt == ' ')

                pt++;

        }

        *c++ = *pt++;

    }

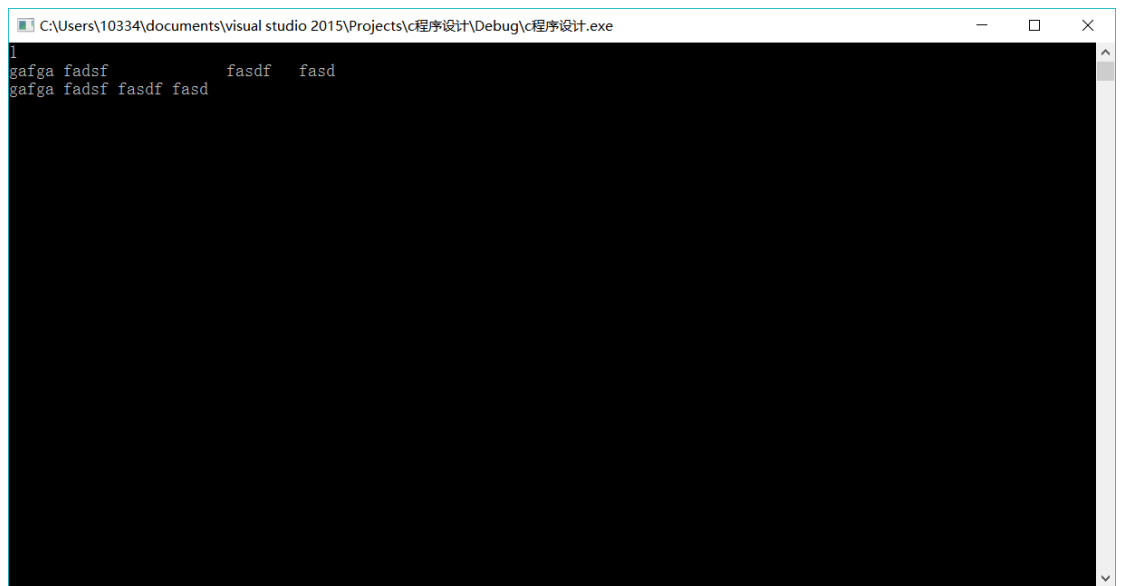
    *c = '\0';

    return first;

}
```



## 3) 运行结果:



```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
1
gafga fadsf fadsf fasd
gafga fadsf fadsf fasd
```

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

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

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

**解答：**

## 1) 解题思路：

- 使用 `define` 定义  $M$ （科目个数） $N$ （学生个数），方便修改函数
- `Score` 二维数组，行存储同一个学生的各科成绩，列存储同一个科目的各学生成绩。

c) Subject 数组记录各个科目名称。

d) Name 数组记录学生姓名。

e) 第一次 for 循环读取 M 个科目的名称。

f) 第二个 for 循环读取学生信息，每一次循环读取该学生姓名以及他的 M 个科目的成绩。

g) `double average_everyone(int * score, int n);` 函数计算每个人的平均成绩。

i. score 为分数，n 为科目数。

ii. 调用该函数时，将该学生的第一个成绩的地址作为参数传递，由于同一个学生的成绩时连续存储的，因此循环相加 n 次可以得到该学生的总成绩，除 n 就是该学生的平均成绩。

h) `double average_subject(int * x, int n);` 计算每个科目的平均成绩

i. 调用该函数时，将第一个学生的某一科目地址作为第一个参数传递，学生个数作为第二个参数，由于同一个科目成绩位于同一列中，因此该科目的第一个成绩（即第一个学生的该科目成绩）向后移动 `n*sizeof(double)` 个字节（假设成绩为 double 类型）就能访问该科目的第二个成绩

ii. 依次访问同一个科目的各个学生的成绩，相加再除就能得到该科目的平均成绩，将此成绩在返回 main 函数后保存在创建的 average 数组中

i) `int less_than_average(int * x, double average, int n);` 统计低于平均成绩的人数

i. 类似 `average_everyone()` 函数，依次访问每个学生的各科成绩，再同 average 比较

ii. 小于 average 计数器 count++。

iii. 返回 count。

j) int perfect\_student(int \* x, int n);统计优秀学生的人数

i. 类似于 less\_than\_average() 函数，同 90 比较，使用 count 计数

ii. 返回 count

2) 程序清单：

```
#include<stdio.h>

#define M 5//M 个科目

#define N 5//N 个学生

double average_everyone(int * score, int n);

double average_subject(int * x, int n);

int less_than_average(int * x, double average, int n);

int fail_student(int * x, int n);

int perfect_student(int * x, int n);

int main(void)

{

    int score[N][M]; //分数

    char subject[M]; //科目

    char name[N][100]; //姓名

    int i, j;

    double average[M];
```

```
/*读取科目*/

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

    scanf(" %c", &subject[i]);


/*读取每个人的姓名和其分数*/

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

{

    scanf(" %s", name[i]);

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

    {

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

    }

}


/*每个人的平均成绩*/

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

{

    printf("Average score of %s is %.2lf\n", name[i],
average_everyone(score[i], M));

}


/*每个科目的平均成绩*/

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

{
```

```
        average[i] = average_subject(&score[0][i], M);

        printf("Average score of %c is %.2lf\n", subject[i],
average[i]);

    }

    /*统计每个科目低于其平均成绩的人数*/

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

    {

        printf("Number of students lower than avg of %c is %d\n",
subject[i], less_than_average(&score[0][i], average[i], N));

    }

    /*统计每个科目不及格的人数*/

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

    {

        printf("Number of students %c fail is %d\n", subject[i],
fail_student(&score[0][i], N));

    }

    /*统计每个科目的优秀人数*/

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

    {

        printf("Number of students %c perfect is %d\n",
subject[i], perfect_student(&score[0][i], N));

    }

    //system("pause");
```

```
}

double average_everyone(int * score, int n)

{

    int i;

    double average = 0;

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

    {

        average += *(score + i);

        //printf("%.2lf***\n", average);

    }

    return average / n;

}

double average_subject(int * x, int n)

{

    int i;

    double average = 0;

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

    {

        average += *(x + N*i);

    }

}
```

```
        return average / n;
    }

    int less_than_average(int * x, double average, int n)
    {
        int count = 0, i;

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

            if (*(x + N*i) <= average)

                count++;

        return count;
    }

    int fail_student(int * x, int n)
    {
        int count = 0, i;

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

            if (*(x + N*i) <60)

                count++;

        return count;
    }

    int perfect_student(int * x, int n)
    {
        int count = 0, i;

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





程序。

提示：p 中元素可为 strcmp、strstr 等函数名。

### 6.3 实验小结：

编写 C 语言程序应熟练掌握指针的说明，赋值，使用；熟悉指向数组指针的使用；同时还要掌握指针函数、函数指针以及指针数组的用法。掌握带有参数的 main 函数的用法。带参数的 main 函数提供了一种向程序传输命令行参数的途径。在所传递的两个参数中，第一个整型形参表示命令行中字符串的个数，另一个则是指向命令行中各字符的字符指针数组。带参数 main 函数的运用提供了命令行可选参数，是程序编写、问题的解决更加多样化。在本次实验过程中，自己在指针数组应用这方面还不是很熟练，在用指针表示一维和多维数组时总是不能一次性得到正确的结果，今后在这方面还有待加强，还要多加训练。在函数指针这方面，我有时还是不能准确用指针定义函数的形参，经常出错，在修改程序上花费了好多时间。如果能准确高效而且熟练的运用指针，那将成为编写程序的有力工具！我们现在还都是初学者，还要努力，多加练习。要做到能够熟练运用指针！

## 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	$*p++ \rightarrow t, *p \rightarrow t$	U, x	U, x
4	$* ( ++p ) \rightarrow t$	x	x
5	$* ++p \rightarrow t$	x	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，先进先出链表的指头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

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

源程序如下：

```
#include "stdio.h"

#include "stdlib.h"

struct s_list{

    int data; /* 数据域 */

    struct s_list *next; /* 指针域 */

} ;

void create_list (struct s_list *headp, int *p);
```

```
void main(void)

{

    struct s_list *head=NULL,*p;

    int s[]={1,2,3,4,5,6,7,8,0}; /* 0 为结束标记 */

    create_list(head,s); /* 创建新链表 */

    p=head; /*遍历指针 p 指向链头 */

    while(p) {

        printf("%d\t",p->data); /* 输出数据域的值 */

        p=p->next; /*遍历指针 p 指向下一结点 */

    }

    printf("\n");

}

void create_list(struct s_list *headp,int *p)

{

    struct s_list * loc_head=NULL,*tail;

    if(p[0]==0) /* 相当于*p==0 */

        ;

    else { /* loc_head 指向动态分配的第一个结点 */

        loc_head=(struct s_list *)malloc(sizeof(struct s_list));

        loc_head->data=*p++; /* 对数据域赋值 */

        tail=loc_head; /* tail 指向第一个结点 */

        while(*p){ /* tail 所指结点的指针域指向动态创建的结点 */
```

```
        tail->next=(struct s_list *)malloc(sizeof(struct
s_list));

        tail=tail->next; /* tail 指向新创建的结点 */

        tail->data=*p++; /* 向新创建的结点的数据域赋值 */

    }

    tail->next=NULL; /* 对指针域赋 NULL 值 */

}

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

}
```

**解答:**

1. 运行结果为空白
2. 修改后:

```
#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 */  
  
}
```

### 3. 运行结果:

```
C:\Users\10334\Documents\7.源程序修改替换题.exe  
1 2 3 4 5 6 7 8  
Process returned 10 (0xA) execution time : 0.169 s  
Press any key to continue.
```

### 4. 微软拼音 半 :

(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.

```
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 */

        loc_head->last = NULL; //loc_head 指向空

        tail = loc_head; /* tail 指向第一个结点 */

        while (*p) {

            loc_head = (struct s_list *)malloc(sizeof(struct s_list)); //新建一个节点

            loc_head->last=tail; /* 新节点的 last 指向上一个节点 */

            tail = loc_head; //tail 指向新节点，为下一次赋值 last 做准备

            tail->data = *p++; /* 向新建的结点的数据域赋值 */

        }

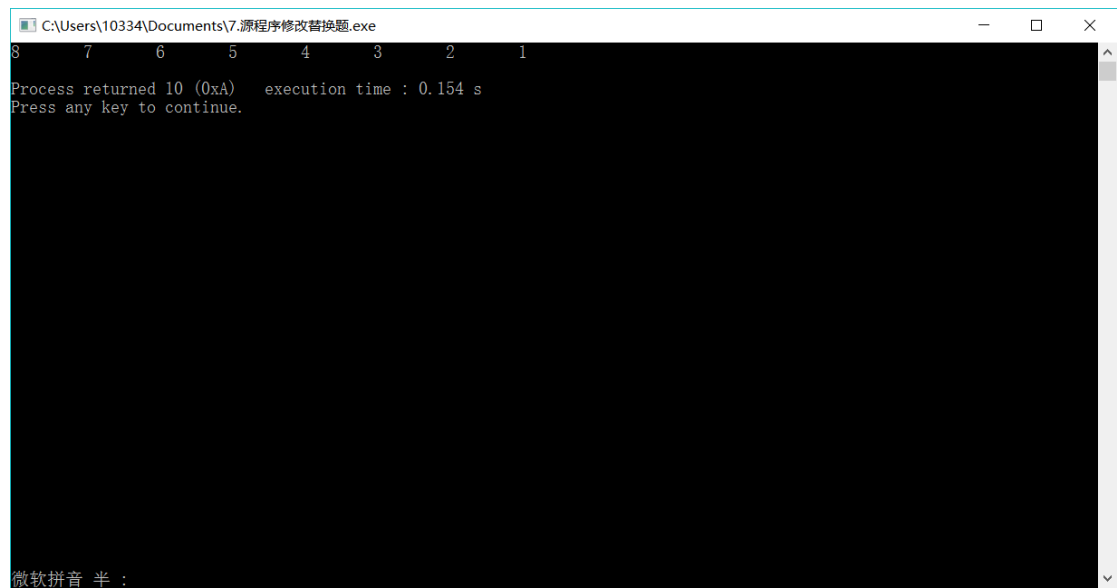
        //tail->last = NULL; /* 对指针域赋 NULL 值 */

    }

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

} 2. 运行结果:
```





### 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) 解题思路：

- a) 声明 8 个中断函数，记为 `isri`。
- b) 声明一个联合为字段赋值，联合占 2 个字节，中断字段也占

两个字节，而且存储是从低位往高位存储的，低 8 位分别对应 8 个 bit 结构，高 8 位为 rsv，因此为 all 赋值后，all 的低 8 位的值就是 bit 的值，all 的高 8 位的值就是 rsv 的值

c) 令结构指针指向相应的中断函数。

d) 依次访问每个字段，使用 if 语句判断是否满足要求，若满足则调用相应函数。

## 2) 程序清单

```
/*字段和联合的配合使用*/
```

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

```
/*定义 8 个中断提示函数*/
```

```
void isr0(void)
```

```
{
```

```
    printf("The Interrupt Service Routine isr0 is called!");
```

```
}
```

```
void isr1(void)
```

```
{
```

```
    printf("The Interrupt Service Routine isr1 is called!");
```

```
}
```

```
void isr2(void)
```

```
{
```

```
    printf("The Interrupt Service Routine isr2 is called!");
```

```
}
```

```
void isr3(void)
```

```
{

    printf("The Interrupt Service Routine isr3 is called!");

}

void isr4(void)

{

    printf("The Interrupt Service Routine isr4 is called!");

}

void isr5(void)

{

    printf("The Interrupt Service Routine isr5 is called!");

}

void isr6(void)

{

    printf("The Interrupt Service Routine isr6 is called!");

}

void isr7(void)

{

    printf("The Interrupt Service Routine isr7 is called!");

}

/*中断字段, bit_i 对应 isr_i, 如果 bit_i 为 1, 则引用 isr_i, rsv 凑够 1
个字节*/

struct ISR_BITS {

    unsigned int bit0 : 1;
```

```
    unsigned int bit1 : 1;

    unsigned int bit2 : 1;

    unsigned int bit3 : 1;

    unsigned int bit4 : 1;

    unsigned int bit5 : 1;

    unsigned int bit6 : 1;

    unsigned int bit7 : 1;

    unsigned int rsv : 8;

};
```

/\*使用联合可以为字段结构赋值,联合占 2 个字节,中断字段也占两个字节,  
而且存储是从低位往高位存储的,低 8 位分别对应 8 个 bit 结构,高 8 位为  
rsv, 因此为 all 赋值后, all 的低 8 位的值就是 bit 的值,

all 的高 8 位的值就是 rsv 的值\*/

```
union ISR_REG {

    unsigned short all;

    struct ISR_BITS bit;

};
```

```
void main(void)
```

```
{
```

```
    /*声明函数指针, 分别 p_isr[i]指向 isri*/
```

```
    void(*p_isr[8])(void);
```

```
    p_isr[0] = isr0;
```

```
p_isr[1] = isr1;

p_isr[2] = isr2;

p_isr[3] = isr3;

p_isr[4] = isr4;

p_isr[5] = isr5;

p_isr[6] = isr6;

p_isr[7] = isr7;

union ISR_REG isr_reg;

int N,i;//N 组输入

scanf("%d", &N);

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

{

    scanf("%hd", &isr_reg.all);

    printf("%d:\n", isr_reg.all);

    /*分别判断各 bit 位的数，如果为 1 则调用对应的中断函数*/

    if (isr_reg.bit.bit0) { p_isr[0](); putchar('\n'); }

    if (isr_reg.bit.bit1) { p_isr[1](); putchar('\n'); }

    if (isr_reg.bit.bit2) { p_isr[2](); putchar('\n'); }

    if (isr_reg.bit.bit3) { p_isr[3](); putchar('\n'); }

    if (isr_reg.bit.bit4) { p_isr[4](); putchar('\n'); }

    if (isr_reg.bit.bit5) { p_isr[5](); putchar('\n'); }

    if (isr_reg.bit.bit6) { p_isr[6](); putchar('\n'); }
```

```
if (isr_reg.bit.bit7) { p_isr[7](); putchar('\n'); }
```

/\*错误：引用函数指针是格式为 p (), 括号里面填参数, 但不能没有括号!!! \*/

```
putchar('\n');
```

```
}
```

```
//system("pause");
```

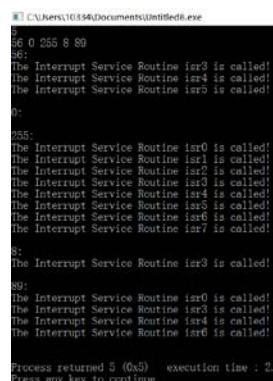
```
} 3) 测试
```

(a) 测试数据:

5

56 0 255 8 89

(b) 运行结果:



```
C:\Users\10334\Documents\Untitled8.exe
56 0 255 8 89
56:
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr4 is called!
The Interrupt Service Routine isr5 is called!
56:
255:
The Interrupt Service Routine isr0 is called!
The Interrupt Service Routine isr1 is called!
The Interrupt Service Routine isr2 is called!
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr4 is called!
The Interrupt Service Routine isr5 is called!
The Interrupt Service Routine isr6 is called!
The Interrupt Service Routine isr7 is called!
56:
The Interrupt Service Routine isr3 is called!
89:
The Interrupt Service Routine isr0 is called!
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr4 is called!
The Interrupt Service Routine isr4 is called!
Process returned 5 (0x5)   execution time : 2.070 s
Press any key to continue.
```

(c) 截图内容:

(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) 解题思路：

- a) 定义学生信息结构 STUDENT\_INFO
- b) 主函数负责调用信息输入、信息处理、信息输出函数
- c) 子函数处理信息：

i. struct STUDENT\_INFO \* input(int N); 信息输入函数，返回链表的头部地址, N 代表 N 组输入, 使用先进先出链表存储数据。并且采用了 if 语句判断 N, 只有在至少有一组输入时才创建链表。

ii. void output(const struct STUDENT\_INFO \*p); 信息输出函数，接受链表的头部地址，遍历到 next 指向空字符为止，依次输出学生信息

iii. void modify(struct STUDENT\_INFO \*p); 信息修改函数，修改指定学生（使用学号定位）的指定成绩，接收链表的头部地址，通过 C 库中的 strcmp() 函数比较学号，定位要修改信息的学生，使用结构指针指向存储该学生信息的结构，接下来同样用 strcmp() 定位要修改的科目，找到后就可进行更改，修改数据数字可以直接赋值，修改字符串则使用 strcpy()。

iv. void SumAndAvg(const struct STUDENT\_INFO \*p, float \*ptof, int subject\_num); 平均成绩和总成绩的计算，参数包括指向头链表的指针 p，存储平均成绩的地址 avg，学生人数 num\_of\_student，使用结构指针遍历学生，计算学生成绩之和。

v. output() 将信息输出。

d) 程序结束运行。

## 2) 程序清单

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/*定义结构*/

struct STUDENT_INFO

{

    char ID_card[15];

    char name[20];

    float ENG, MATH, PHY, C;

    struct STUDENT_INFO *next;

};

/*信息输入函数，返回链表的头部地址,N 代表 N 组输入*/

struct STUDENT_INFO * input(int N);

/*信息输出函数，接受链表的头部地址，遍历到 next 指向空字符为止，依次输出学生信息*/

void output(const struct STUDENT_INFO *p);

/*信息修改函数，修改指定学生（使用学号定位）的指定成绩，接收链表的头部地址*/

void modify(struct STUDENT_INFO *p);

/*平均和总成绩计算*/
```



```
void SumAndAvg(const struct STUDENT_INFO *p, float * ptof, int
subject_num);

/*以平均成绩排序*/

void Sort(struct STUDENT_INFO *p, float * avg, int num_of_student);

/*排序后输出*/

void sort_output(const struct STUDENT_INFO *p, float *avg);

int main(void)
{
    int N;

    float avg[20]; //存储平均成绩

    scanf("%d", &N);

    getchar();

    struct STUDENT_INFO *pt;

    pt = input(N);

    output(pt);

    modify(pt);

    printf("Alter:\n");

    output(pt);

    printf("SumAndAvg:\n");

    SumAndAvg(pt, avg, 4);

    putchar('\n');

    printf("Sort:\n");

    Sort(pt, avg, N);
```

```
    sort_output(pt, avg);

    //system("pause");

return 0

}

struct STUDENT_INFO * input(int N)

{

    int i;

    struct STUDENT_INFO *head, *tail, *p;

    if (N)

    {

        head = (struct STUDENT_INFO *)malloc(sizeof(struct
STUDENT_INFO));

        tail = head;

        scanf(" %s %s %f %f %f %f", tail->ID_card, tail->name,
&tail->ENG, &tail->MATH, &tail->PHY, &tail->C);

        getchar();

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

        {

            tail->next = (struct STUDENT_INFO *)malloc(sizeof(struct
STUDENT_INFO));

            tail = tail->next;

            scanf(" %s %s %f %f %f %f", tail->ID_card, tail->name,
```

```
&tail->ENG, &tail->MATH, &tail->PHY, &tail->C);

    }

    tail->next = NULL;

}

else return 0;

return head;

}

void output(const struct STUDENT_INFO *p)

{

    printf("%-15s%-20s%-10s%-10s%-10s%-10s\n",    "ID",    "Name",
"English", "Math", "Physics", "C");

    while (p)

    {

        printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",
p->ID_card, p->name, p->ENG, p->MATH, p->PHY, p->C);

        p = p->next;

    }

    putchar('\n');

}

void modify(struct STUDENT_INFO *p)

{
```

```
int N, i;

char ID[15];

char subject[20];

float new_score;

struct STUDENT_INFO *origin;

origin = p;

scanf("%d", &N);

for (i = 1; i <= N; i++, p=origin)
{
    scanf("%s %s %f", ID, subject, &new_score);

    //printf("%s %s %f", ID, subject, new_score);

    /*寻找学生*/

    while (strcmp(ID, p->ID_card))

    {
        p = p->next;
    }

    /*寻找科目*/

    if (!strcmp("English", subject)) p->ENG=new_score;

    else if (!strcmp("Math", subject)) p->MATH = new_score;

    else if (!strcmp("Physics", subject)) p->PHY = new_score;

    else if (!strcmp("C", subject)) p->C = new_score;
}
```

```
    }

    void SumAndAvg(const struct STUDENT_INFO *p, float *ptof, int
subject_num)

    {

        float sum;

        int i=0;

        printf("%-15s%-20s%-10s%-10s\n", "ID", "Name", "SUM", "AVG");

        while (p)

        {

            sum = ptof[i] = 0;

            sum = p->C + p->MATH + p->PHY + p->ENG;

            *ptof = sum / subject_num;

            printf("%-15s%-20s%-10.2f%-10.2f\n", p->ID_card, p->name,
sum, *ptof);

            ptof++;

            p = p->next;

        }

    }

    void Sort(struct STUDENT_INFO *p, float * avg, int num_of_student)

    {

        int i, j, k;

        int count;
```

```
struct STUDENT_INFO *pt = p;

for (i = 0; i < num_of_student; i++, p=p->next, pt = p)
{
    k = i;

    for (j = i; j < num_of_student; j++)
    {
        if (avg[j] < avg[k])
            k = j;
    }

    if (k != i)
    {
        float temp;

        temp = avg[i];
        avg[i] = avg[k];
        avg[k] = temp;

        for (count = 0; count < k - i; count++)
            pt = pt->next; //此时 pt 指向 avg 最小的结构

        struct STUDENT_INFO *p1, *p2, ptemp;

        p1 = p->next;
        p2 = pt->next;

        ptemp = *p;
        *p = *pt;
```

```
        *pt = ptemp;

        pt->next = p2;

        p->next = p1;

    }

}

}

void sort_output(const struct STUDENT_INFO *p, float * avg)

{

    printf("%-15s%-20s%-10s\n", "ID", "Name", "AVG");

    while (p)

    {

        printf("%-15s%-20s%-10.2f\n", p->ID_card, p->name, *avg);

        p = p->next;

        avg++;

    }

    //putchar('\n');

}
```

## 2) 测试

### (a) 测试数据:

5//输入输出

U20140101 ZhangChuanChao 85 86 87 88

U20140126 MaiDouDou 99 99 99 99

U20140158 XiaoDouDou 56 85 89 59

U20140312 DaoDaoDog 84 89 65 100

U20140359 XiDaDa 88.8 88.8 88.8 88.8

3//修改数据

U20140101 Math 95.6

U20140359 C 100

U20140359 English 100

(b) 测试结果:

## 7.2.4. 选做题

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



(2) 对选做题第(1)题,进一步写出用交换结点指针域的方法升序排序的函数。

(3) 采用双向链表重做编程设计题第(2)题。

### 7.3 实验总结

本次实验相对较难,编写的程序代码也较长。通过这次实验,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。其中,成绩处理这一题充分体现了函数式编程的优点,将一个大问题化为多个小问题,分部调试,哪里出了问题就修改相应的函数即可,而不必审阅整个程序,大大提高了效率,节省了时间。另外,程序修改和替换这道题让我掌握的先进先出链表和先进后出链表的声明,尤其要注意的是在使用函数声明先进先出链表时函数的形参要声明为指向指针的指针,该指针指向已声明的结构,再引用该函数时要将头指针的地址作为实参传递,而不能传递它的值。如程序修改替换这一题,传递了 head 存储的值(NULL),因此不能正确声明链表,同样也无法对 head 进行操作。

## 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;

}
```

(1) 请思考程序的输出结果，然后通过上机运行来加以验证。

a) ?%}{

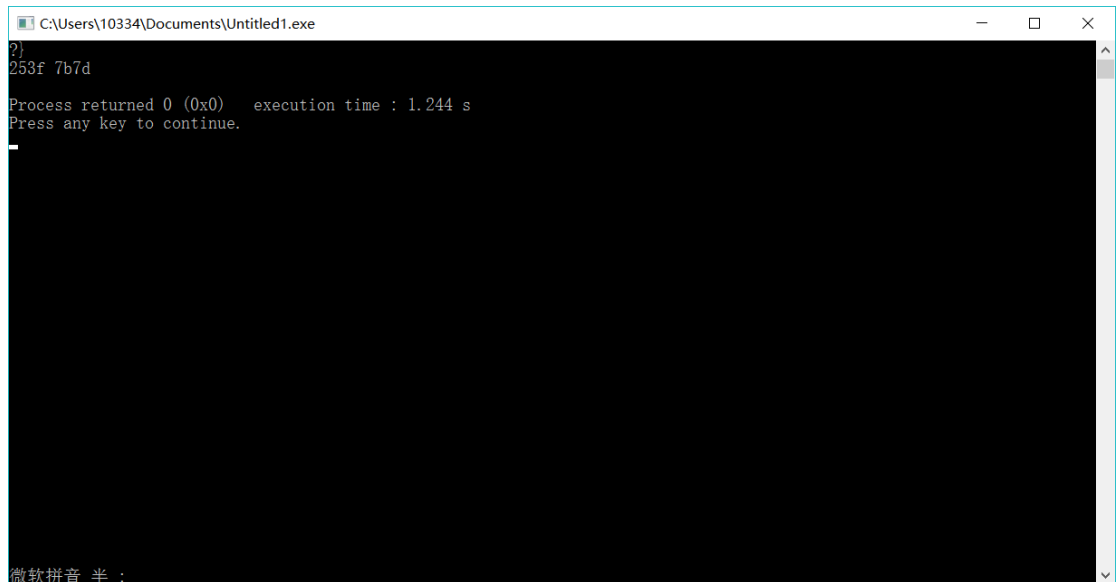
b) 253f 7b7d

c) 微软拼音 半 :

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

a) `?}`

b) `253f 7b7d`



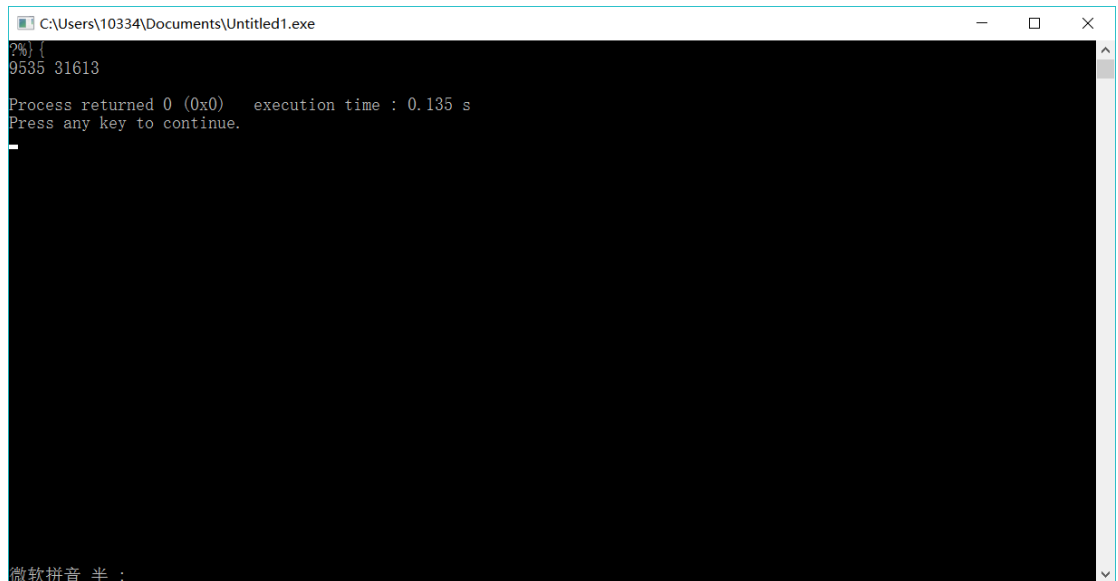
c) 微软拼音 半 :

d) 原因: `char` 类型为一个字节, `short` 类型为两个字节, 那么将 `short` 改为 `char` 后, 运行第一个 `fwrite()` 时, 只将 `a` (`0x253f`) 的第一个字节的信息, 即 `3f` 存储到了 `d:\abc1.bin` 文件中, 在运行第二个 `fwrite()` 时, 只将 `b = 0x7b7d` 的第一个字节的信息 (即 `7d`) 存储在了 `abc1.bin` 中, 因此在执行 `while((ch = fgetc(fp1)) != EOF) putchar(ch);` 时只输出了 `3f` 对应的 acii 码 `'?'` 和 `7d` 对应的 acii 码 `'}'`。

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

a) `?%}{`

b) `9535 31613`



c) 微软拼音 半 :

d) 第一次输出 253f 7b7d 是以 16 进制数存储的, 将 %hx 改为 %d 后是以 10 进制数存储的。

### 8.2.2. 源程序修改替换题

将指定的文本文件内容在屏幕上显示出来, 命令行的格式为:

```
type filename
```

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

```
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 }
```

**解答:**

1. 第 16 行 `while(ch=fgetc(fp)!=EOF)` 中 `ch=fgetc(fp)` 要加括号, 正确形式为 `while((ch=fgetc(fp))!=EOF)`

(2) 用输入输出重定向 `freopen` 改写上述源程序中的 `main` 函数。

a) 程序清单:

```
#include<stdio.h>

#include<stdlib.h>

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

{

    char ch;
```

```
FILE *fp;

if(argc!=2) {

    printf("Arguments error!\n");

    exit(-1);

}

if((fp=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
*/

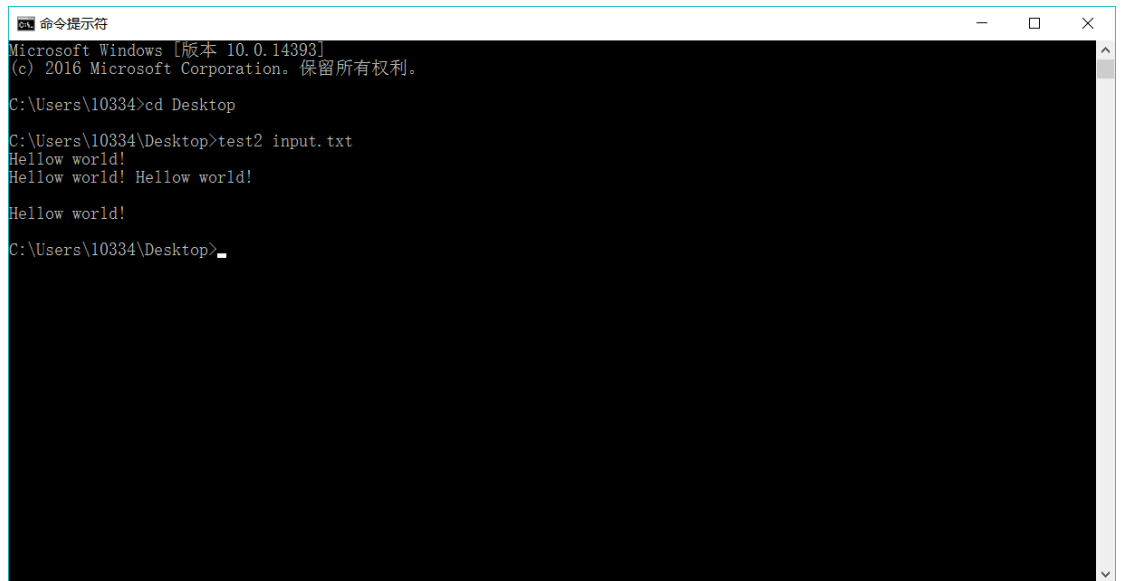
    putchar(ch);                                         /* 向显示器中写字符 */

fclose(fp);                                             /* 关闭 filename */

return 0;

}
```

b) 运行结果:



```
命令提示符
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\10334>cd Desktop
C:\Users\10334\Desktop>test2 input.txt
Hellow world!
Hellow world! Hellow world!

Hellow world!
C:\Users\10334\Desktop>
```

### 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. 解题思路：

- 当一个字母的前一个字母是空格或者换行符时，我们就认为该字母是首字母，定义一个 `char last` 存储上一个字母。
- 通过 `c=getchar()` 读取输入，如果上一个字符是空格的话，就输出 `toupper(c)`，否则输出 `c`，输出完毕后 `last=c`，进行下一次读取，知道 `c=EOF` 时退出循环。
- 调试成功后在程序开头加上 `fp=fopen("test.txt", "w", stdout)`，通过 `freopen()` 函数将 `test.txt` 设置为标准输出，我们就可以将数据存放到 `test.txt` 文件中了。

#### 2. 程序清单：

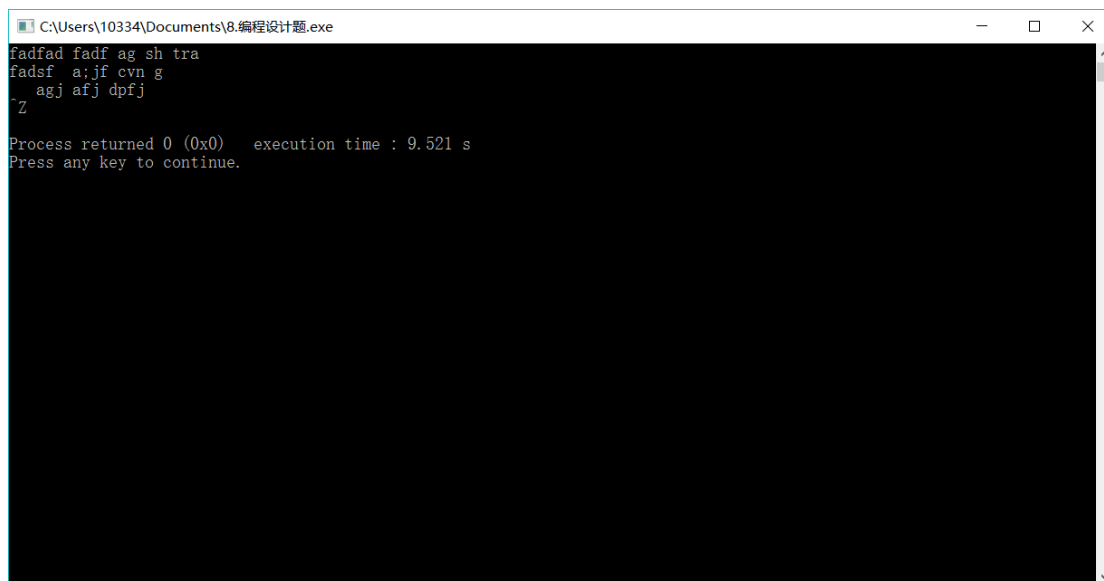
```
1  #include<stdio.h>

2  #include<ctype.h>
```

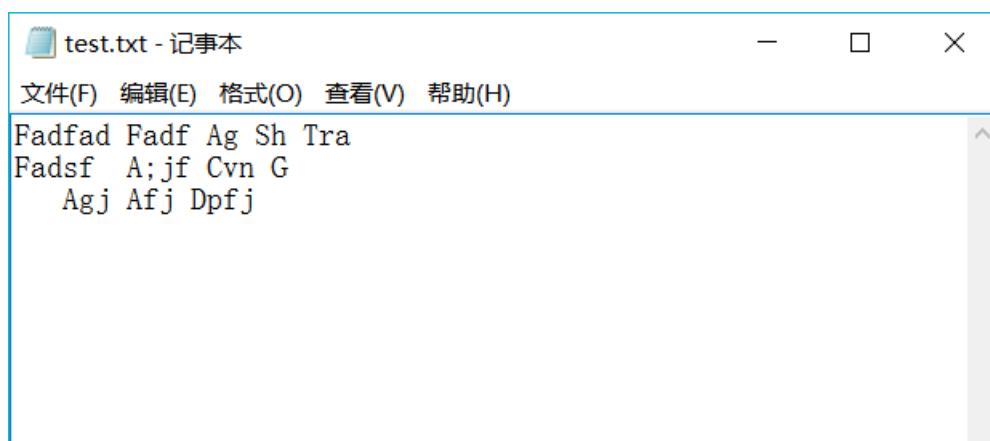


```
3  int main()
4  {
5      FILE * fp;
6      fp=fopen("test.txt","w",stdout);
7      char c,last;
8      last=' ';
9      while((c=getchar())!=EOF)
10     {
11         if((last==' '||last=='\n')&&c!=' ')
12             putchar(toupper(c));
13         else putchar(c);
14         last=c;
15     }
16     fclose(fp);
17     return 0;
18 }
19
```

### 3. 运行结果:



```
C:\Users\10334\Documents\8.编程设计题.exe
fadfad fadf ag sh tra
fadsf a;jf cvn g
agj afj dpfj
~Z
Process returned 0 (0x0) execution time : 9.521 s
Press any key to continue.
```



```
test.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Fadfad Fadf Ag Sh Tra
Fadsf A;jf Cvn G
Agj Afj Dpfj
```

### 8.3 实验小结

文件的读写操作让 c 程序的数据得以保存，让数据的存取变得简单易行，让我感觉自己写的程序进入了更高的台阶，程序的生命周期得以延长，要加强对文件读写操作的练习，争取熟练掌握。

在实际应用中，经过这次实验我掌握了从文件中读取数据的方法，对于大量数据，通过键盘输入的方法不现实也不可取，我们可以将数据存储到文本文件中，然后通过带参数的 `main()` 和文件处理函数从指定文件中读取数据，十分方便。另外，有时我们需要将处理的数据进行存储，文件读写操作为我们提供了一种途径将处理后的数据存放到指定文件之中。

## 参考文献：

C 语言程序与设计-北京电子工业出版社，2013.1

C Primer Plus 第六版/（美），普拉塔著；云巅工作室  
译