

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: 物联网工程 1601

学 号: U201614897

姓 名: 潘越

指导教师: 刘芳

报告日期: 2017.6.7

计算机科学与技术学院

目 录

实验 1 表达式输入与输出实验.....	1
1.1 实验目的.....	1
1.2 实验内容.....	1
1.2.1 源程序改错.....	1
1.2.2 源程序修改替换.....	4
1.2.3 程序设计.....	5
1.3 自设题.....	10
1.4 实验小结.....	12
实验 2 流程控制实验.....	13
2.1、实验目的.....	13
2.2、实验内容.....	13
2.2.1 源程序改错.....	13
2.2.2 源程序修改替换.....	14
2.2.3 程序设计.....	16
2.3 自设题.....	28
2.4 实验小结.....	30
实验 3 函数与程序结构实验.....	31
3.1 实验目的.....	31
3.2 实验内容.....	31
3.2.1 源程序改错.....	31
3.2.2 源程序修改替换.....	33
3.2.3 跟踪调试题.....	35
3.2.4 程序设计.....	37
3.3 自设题.....	44
3.4 实验小结.....	49
实验 4 编译预处理实验.....	50
4.1 实验目的.....	50
4.2 实验内容.....	50
4.2.1 源程序改错.....	50
4.2.2 源程序修改替换.....	51
4.2.3 跟踪调试题.....	55
4.2.4 程序设计.....	58
4.3 实验小结.....	61
实验 5 数组实验.....	62
5.1 实验目的.....	62
5.2 实验内容.....	62
5.2.1 源程序改错.....	62
5.2.2 源程序修改替换.....	63
5.2.3 跟踪调试题.....	68
5.2.4 程序设计.....	71

5.3 实验小结.....	82
实验 6 指针实验.....	83
6.1 实验目的.....	83
6.2 实验内容.....	83
6.2.1 源程序改错.....	83
6.2.2 源程序修改替换.....	84
6.2.3 跟踪调试题.....	87
6.2.4 程序设计.....	91
6.3 实验小结.....	108
实验 7 结构与联合实验.....	109
7.1 实验目的.....	109
7.2 实验内容.....	109
7.2.1 表达式求值与程序验证.....	109
7.2.2 源程序修改替换.....	110
7.2.3 程序设计.....	115
7.3 自设题.....	136
7.4 实验小结.....	149
实验 8 文件实验.....	150
8.1 实验目的.....	150
8.2 实验内容.....	150
8.2.1 文件类型的程序验证.....	150
8.2.2 源程序修改替换.....	152
8.2.3 程序设计.....	155
8.3 实验小结.....	156

实验 1 表达式输入与输出实验

1.1 实验目的

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

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

(3) 掌握简单 C 程序（顺序结构程序）的编写方法。

1.2 实验内容

1.2.1 源程序改错

下面给出了一个简单 C 语言程序例程，用来完成以下工作：

(1) 输入华氏温度 `f`，将它转换成摄氏温度 `c` 后输出；

(2) 输入圆的半径值 `r`，计算并输出圆的面积 `s`；

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

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

```
1  #include<stdio.h>
2  #define PI 3.14159;
3  voidmain( void )
4  {
5      int f;
6      short p, k;
7      double c , r , s;
8      /* for task 1 */
9      printf("Input  Fahrenheit:");
```

```

10    scanf("%d", f);
11    c = 5/9*(f-32);
12    printf( " \n %d (F) = %.2f (C)\n\n ", f, c );
13 /* for task 2 */
14    printf("input the radius r:");
15    scanf("%f", &r);
16    s = PI * r * r;
17    printf("\nThe acreage is %.2f\n\n",&s);
18 /* for task 3 */
19    printf("input hex int k, p :");
20    scanf("%x %x", &k, &p );
21    newint = (p&0xff00)|(k&0xff00)<<8;
22    printf("new int = %x\n\n",newint);
23 }

```

解答:

(1) 错误修改:

1)第 2 行的符号常量定义后不能有分号, 正确形式为:

```
define PI 3.14159
```

2)第 3 行的 voidmain 中间应该有空格, 正确形式为:

```
void main (void)
```

3)第 9 行 printf(“Input Fahrenheit.”);中间使用了全角双引号, 正确形式为:

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

4)第 10 行 scanf(“%d”, f);中间使用了全角双引号, 正确形式为:

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

5)第 10 行 scanf(“%d”, f);中间应该使用 f 的地址, 正确形式为:

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

6)第 11 行 c = 5/9*(f-32);进行的是整数计算, 正确形式为:

```
c = 5.0/9.0*(f-32);
```

7)第 12 行 `printf(“\n %d (F) = %.2f (C)\n\n”, f, c);`中间使用了全角双引号, 正确形式为:

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

8)第 15 行 `scanf("%f", &r);`占位符应该使用`%lf`(double 类型), 正确形式为:

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

9)第 17 行 `printf("\nThe acreage is %.2f\n\n",&s);`中打印的是 `s` 的地址而非值, 正确形式为:

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

10)第 21 行的 `newint` 为定义, 正确形式为:

在开始加上 `short newint`

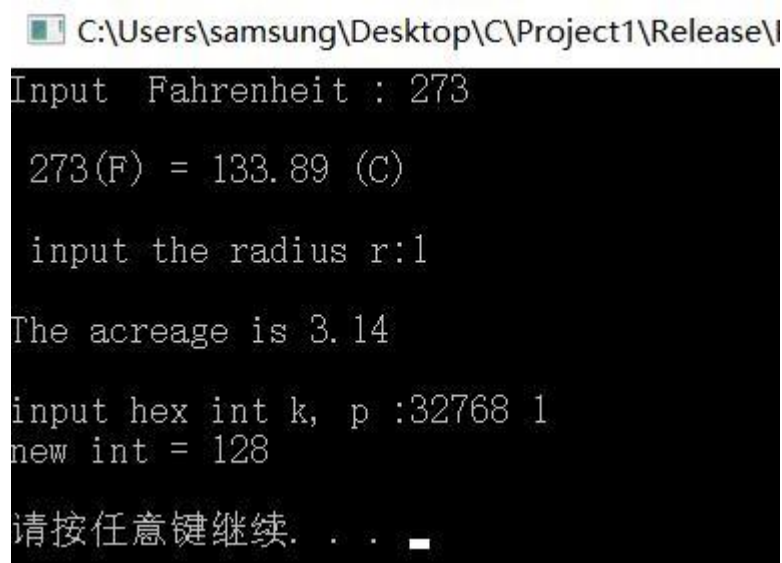
11)第 21 行 `newint = (p&0xff00)|(k&0xff00)<<8;`中后面应该是右移 8 位, 正确形式为:

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

12)第 22 行 `printf("new int = %x\n\n",newint);`中占位符应该使用`%hd`(整型), 正确形式为:

```
printf("new int = %hd\n\n",newint);
```

(2) 错误修改后运行结果:



```
C:\Users\samsung\Desktop\C\Project1\Release\  
Input Fahrenheit : 273  
273(F) = 133.89 (C)  
input the radius r:1  
The acreage is 3.14  
input hex int k, p :32768 1  
new int = 128  
请按任意键继续. . .
```

图 1-1 错误改正之后的测试图

1.2.2 源程序修改替换

下面的程序利用常用的中间变量法实现两数交换，请改用不使用第 3 个变量的方法实现。该程序中 `t` 是中间变量，要求将定义语句中的 `t` 删除，修改下划线处的语句，使之实现两数对调的操作。

```
#include<stdio.h>

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

解答：

`t=a, a=b, b=t;` 改为 `a = a * b, b = a / b, a = a / b;` 替换后的程序如下所示：

```
#include<stdio.h>

void main( )
{
    int a, b, t;
    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);
}
```

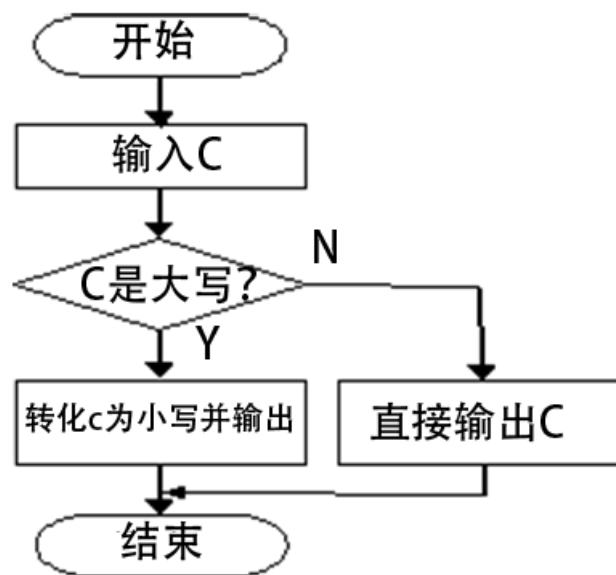



图 1-2 程序替换之后的测试图

1.2.3 程序设计

(1) 编写一个程序，输入字符 c ，如果 c 是大写字母，则将 c 转换成对应的小写，否则 c 的值不变，最后输出 c 。

解答：



1) 算法流程如图 1.1 所示

图 1-3 编程题 1 的程序流程图

2) 源程序清单

```

#include<stdio.h>

int main(void) {
    char c;

    printf("Please enter a character: ");
    scanf("%c", &c);

```

```

if (c >= 65 && c <= 90)
    printf("\n%c\n", c + 'a' - 'A');
else
    printf("\n%c\n", c);
system("pause");
return 0;
}

```

3) 测试

(a) 测试数据:

Z s g A

(b) 对应测试数据的运行结果截图



图 1-4 编程题 1 用 ‘Z’ 进行测试的测试图

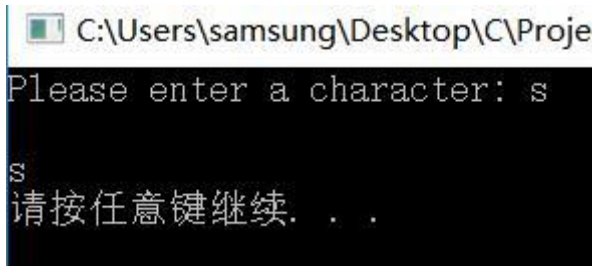


图 1-5 编程题 1 用 ‘s’ 进行测试的测试图

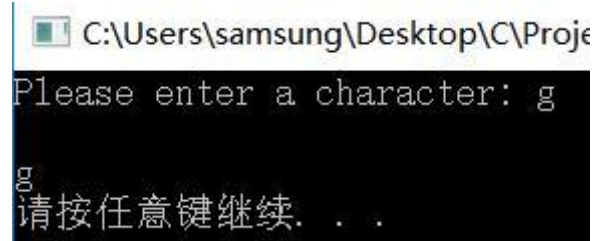


图 1-6 编程题 1 用 ‘g’ 进行测试的测试图

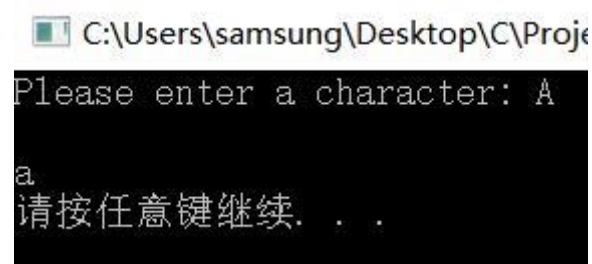


图 1-7 编程题 1 用 ‘A’ 进行测试的测试图

(2) 编写一个程序，输入无符号短整数 x , m , n ($0 \leq m \leq 15, 1 \leq n \leq 16-m$), 取出 x 从第 m 位开始向左的 n 位 (m 从右至左编号为 $0 \sim 15$), 并使其向左端 (第 15 位) 靠齐。

解答:

1) 解题思路:

1. 输入 x , m , n , 为了方便分析测试结果, x 的输入采用 16 进制

2. 如果 $0 \leq m \leq 15, 1 \leq n \leq 16-m$, 转 2.1, 否则转 3.

2.1 首先 $x \gg m$, 将要处理的 n 位移动到最右;

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

2.3 用 16 进制输出结果并转 4.

3. 显示输入错误信息;

4. 结束

2) 程序清单

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

```
void main(void)
```

```
{
```

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

```
    printf("输入 x (16 进制)、m (0~15) 和 n (1~16-m): \n");
```

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

```
    if (0 <= m && m <= 15 && 1 <= n && n <= 16 - m)    /*判断 m、n 的值是
    否在合理范围内*/
```

```
        printf("ans=%hx\n", (x >> m) << (16 - n));
```

```
    else    printf("输入错误!\n");
```

```
}
```

3) 测试

(a) 测试数据

表 1-1 编程题 2 的测试数据

测试用例	程序输入			理论结果	运行结果
	X	m	N		
用例 1	0100 0110 1000 0000 (4680)	7	4	计算结果 1101 0000 0000 0000 即 D000	D000

用例 2	1101 0101 1000 0011 (D583)	16	1	输入错误 (m 值超范围)	输入错误!
用例 3	1101 0101 1000 0011 (D583)	13	5	输入错误 (n 值超范围)	输入错误!

(b) 对应测试的运行结果

```

选择C:\Users\samsung\Desktop\C\Project1\Release\Project1.exe
输入x (16进制)、m (0~15) 和n (1~16-m) :
4680 7 4
ans=d000
请按任意键继续. . .

```

图 1-8 编程题 2 的测试用例一的运行结果

```

C:\Users\samsung\Desktop\C\Project1\Release\Project1.exe
输入x (16进制)、m (0~15) 和n (1~16-m) :
D583 16 1
输入错误!
请按任意键继续. . .

```

图 1-9 编程题 2 的测试用例二的运行结果

```

C:\Users\samsung\Desktop\C\Project1\Release\Project1.exe
输入x (16进制)、m (0~15) 和n (1~16-m) :
D583 13 5
输入错误!
请按任意键继续. . .

```

图 1-10 编程题 2 的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

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

解答：

1) 解题思路：

- 1.输入一个数字。
- 2.分别用四个变量储存该数的每 8 位。
- 3.将每 8 位转换为一个十进制数字。
- 4.以常见的句点分隔的 4 部分的形式输出。

2) 程序清单:

```
#include<stdio.h>

int main(void)
{
    unsigned int i, x, y, z, m;

    printf("请输入一个机器存储的 32 位的互联网 IP 地址:  \n");

    scanf("%d", &i);

    x = (i >> 24) & ~(~0 << 8);
    y = (i >> 16) & ~(~0 << 8);
    z = (i >> 8) & ~(~0 << 8);
    m = (i >> 0) & ~(~0 << 8);

    printf("%d.%d.%d.%d", x, y, z, m);

    return 0;
}
```

3) 测试

(a) 测试数据

表 1-2 编程题 3 的测试数据

测试用例	程 序 输 入	理 论 结 果	运 行 结 果
	i		
用例 1	00100000 00110111 00000001 01100110 (540475750)	对应 IP 地址 32.55.1.102	32.55.1.102
用例 2	01110010 01110010 01110010 01110010 (1920103026)	对应 IP 地址 114.114.114.114	114.114.114.114
用例 3	01100110 11010110 01110011 11010101 (1725330389)	对应 IP 地址 102.214.115.213	102.214.115.213

(b) 对应测试的运行结果



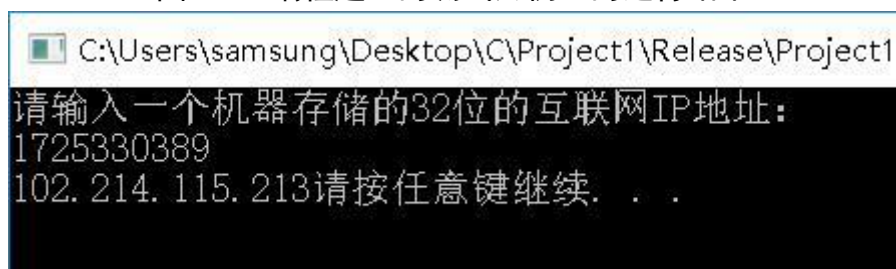
```
C:\Users\samsung\Desktop\C\Project1\Release\Project1.exe
请输入一个机器存储的32位的互联网IP地址:
540475750
32. 55. 1. 102请按任意键继续. . .
```

图 1-11 编程题 3 的测试用例一的运行结果



```
C:\Users\samsung\Desktop\C\Project1\Release\Project1.exe
请输入一个机器存储的32位的互联网IP地址:
1920103026
114. 114. 114. 114请按任意键继续. . .
```

图 1-12 编程题 3 的测试用例二的运行结果



```
C:\Users\samsung\Desktop\C\Project1\Release\Project1.exe
请输入一个机器存储的32位的互联网IP地址:
1725330389
102. 214. 115. 213请按任意键继续. . .
```

图 1-13 编程题 3 的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1.3 自设题

(1) 自设实验题目：用 C 语言以字符串的方法解：“有一个整数 abcde，其 4 倍是 ecdab，求该数”

(2) 实验目的：本题用数字来做非常简单，这里尝试使用字符串方法来重解，以达到锻炼 C 语言输入输出字符的能力。

(3) 题目分析：本题的数学方法可以取出每一位数字，再按照十进制的表示一次倒序乘以 10，100，1000，10000。这里将数字转换为字符串，再进行倒序，再转换为数字进行比较。又由于乘以 4 之后是五位数，故该数字小于 25000，且其倒过来是偶数（原数的四倍），故其首位数必为 2，于是只用循环 20000-25000 即可。

(3) 实验程序:

```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
    int i, j;
    char num[6], num4[6], reverse[6];
    for (i = 20000; i <= 25000; i++)
    {
        _itoa(i, num, 10);
        _itoa(i * 4, num4, 10);
        for (j = 0; j < 5; j++)
        {
            reverse[j] = num4[4 - j];
        }
        reverse[5] = num[5];
        if (i == atoi(reverse))
            printf("An answer is %d\n", i);
    }
    return 0;
}
```

(5) 实验测试:

输出的结果是: 21978 21978 * 4=87912, 验证成立

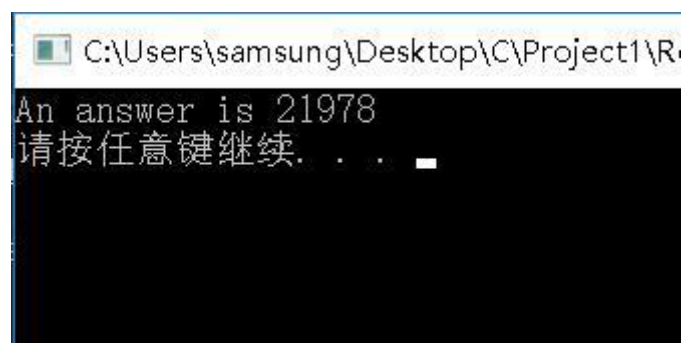


图 1-14 自测题的测试运行结果

附例：用数字方法解决该题：

```
#include<stdio.h>

int main(void)
{
    int i, j;
    int a, b, c, d, e;
    for (i = 20000; i < 25000; i++)
    {
        a = i % 10;
        b = (i / 10) % 10;
        c = (i / 100) % 10;
        d = (i / 1000) % 10;
        e = i / 10000;
        j = e + d * 10 + c * 100 + b * 1000 + a * 10000;
        if (i * 4 == j)
            printf("A answer is %d\n", i);
    }
    return 0;
}
```

(5) 实验结论： 用字符串方法解题，实际上就是代替了数学方法里面的取出每一位数字，并且按照相应的进制乘回去的方法，但显然在非十进制的时候，使用 `_itoa` 函数和 `atoi` 函数要比使用第二种方法更改程序要轻松许多。

1.4 实验小结

通过这次实验，我对 C 语言的输入和输出有了更近一步的认识，更加熟悉了各种占位符的应用，了解了一些简单问题的算法，C 语言能力有了进一步的提升。

实验 2 流程控制实验

2.1、实验目的

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

(2)练习循环结构 for、while、do-while 语句的使用。

(3)练习转移语句和标号语句的使用。

(4)使用 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

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:

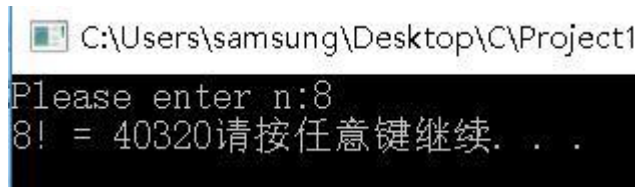
(1) 错误修改

1)第六行 `scanf("%d",n);`用错, 正确形式为:

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

2)第七行 `for(i=1,i<=n,i++)`用错, 正确形式为:

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



(2) 错误修改后运行结果:

图2-1 错误改正之后的测试图

2.2.2 源程序修改替换

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

(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) 修改后的程序如下所示

1)用 `while` 语句:

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

```
void main(void)
```

```
{
```

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

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

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

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

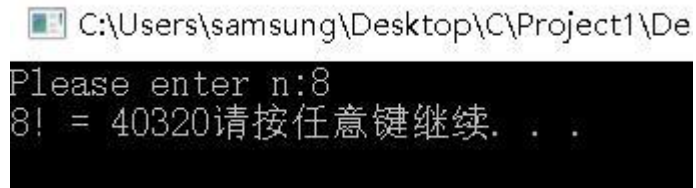
```
        s *= i;
```

```
        i++;
```

```
    }
```

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

```
return 0;
```



```
}
```

图2-2 程序替换之后的测试图

2)用 do-while 语句:

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

```
void main(void)
```

```
{
```

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

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

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

```
    do {
```

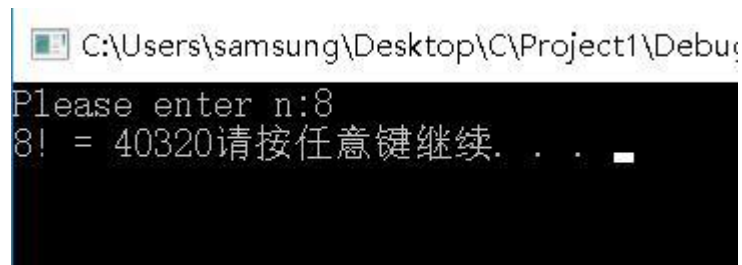
```
        s *= i;
```

```
        i++;
```

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

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

```
    return 0;
```



```
}
```

图2-3 程序替换之后的测试图

(2) 修改后的程序如下所示

```
#include <stdio.h>

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



图2-4 程序替换之后的测试图

2.2.3 程序设计

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

解答：

(1) 解题思路：根据对应税金区域列式计算即可。

(2) 程序清单:

a) if 语句实现

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

```
int main(void) {  
    double x, fax;  
    scanf("%lf", &x);  
    while (x != 0) {  
        if (x <= 1000)  
            fax = 0;  
        else if (x <= 2000)  
            fax = (x - 1000) * 0.05;  
        else if (x <= 3000)  
            fax = (x - 2000) * 0.1 + 1000 * 0.05;  
        else if (x <= 4000)  
            fax = (x - 3000) * 0.15 + 1000 * (0.1 + 0.05);  
        else if (x <= 5000)  
            fax = (x - 4000) * 0.2 + 1000 * (0.15 + 0.1 + 0.05);  
        else  
            fax = (x - 5000) * 0.25 + 1000 * (0.2 + 0.15 + 0.1 + 0.05);  
        printf("%lf\n", fax);  
        scanf("%lf", &x);  
    }  
    return 0;  
}
```

b) switch 语句实现

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

```
int main(void) {  
    double x, fax;
```

```

scanf("%lf", &x);
int num = (int)x / 1000;
while (x != 0) {
    switch (num) {
        case 0:
            fax = 0;
            break;
        case 1:
            fax = (x - 1000) * 0.05;
            break;
        case 2:
            fax = (x - 2000) * 0.1 + 1000 * 0.05;
            break;
        case 3:
            fax = (x - 3000) * 0.15 + 1000 * (0.1 + 0.05);
            break;
        case 4:
            fax = (x - 4000) * 0.2 + 1000 * (0.15 + 0.1 + 0.05);
            break;
        default:
            fax = (x - 5000) * 0.25 + 1000 * (0.2 + 0.15 + 0.1 + 0.05);
            break;
    }
    printf("%lf\n", fax);
    scanf("%lf", &x);
    num = (int)x / 1000;
}
return 0;
}

```

(3) 测试

a) 测试数据

500

1000

1500

2000

2500

3000

3500

4000

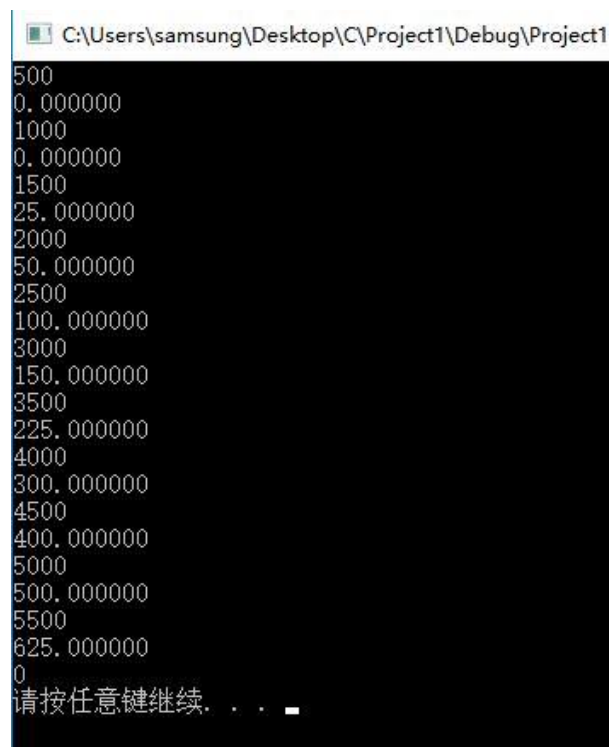
4500

5000

5500

0

b) 测试结果



```
C:\Users\samsung\Desktop\C\Project1\Debug\Project1
500
0.000000
1000
0.000000
1500
25.000000
2000
50.000000
2500
100.000000
3000
150.000000
3500
225.000000
4000
300.000000
4500
400.000000
5000
500.000000
5500
625.000000
0
请按任意键继续. . .
```

图 2-5 编程题 1 用 if 语句得到的测试图

```
C:\Users\samsung\Desktop\C\Project
500
0.000000
1000
0.000000
1500
25.000000
2000
50.000000
2500
100.000000
3000
150.000000
3500
225.000000
4000
300.000000
4500
400.000000
5000
500.000000
5500
625.000000
0
请按任意键继续. . .
```

图 2-6 编程题 1 用 switch 语句得到的测试图

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

解答：

(1) 解题思路

1. 定义两种状态，COPY 和 SPACE；

2. 开始时将状态设置为 COPY

3. 循环：读取字符 c

3.1 当状态为 COPY 时：

若 c 为空格，状态改为 SPACE，输出 c，进行下一次循环，否则只输出 c，进行下一次循环；

3.2 当状态为 SPACE 时：

若 c 为空格，状态不变，进行下一次循环，否则输出 c，并将状态改为 COPY，进行下一次循环。

(2) 程序清单

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



```

enum {COPY, SPACE};

int main(void) {
    int n, i;
    scanf("%d", &n);
    getchar();
    for (i = 1; i <= n; i++) {
        int state = COPY;
        char c;
        while ((c = getchar()) != '\n') {
            switch (state) {
                case COPY:
                    if (c == ' ') {
                        putchar(c);
                        state = SPACE;
                    }
                    else
                        putchar(c);
                    break;
                case SPACE:
                    if (c == ' ')
                        state = SPACE;
                    else {
                        putchar(c);
                        state = COPY;
                    }
                    break;
            }
        }
    }
}

```


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

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

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) 解题思路：

1. 输入正整数 n，杨辉三角的行数 n；
2. 若 n 不为 0 对每行进行循环：
 - 2.1 用一个循环打印第 i 行行首的 2 * (n - i - 1) 个空格；
 - 2.2 对每行进行循环：
 - 2.21 以题中所给公式计算 Cij；
 - 2.22 输出 Cij；
 - 2.3 打印换行符；
 - 2.4 再次输入 n；
3. 程序结束。

(2) 程序清单：

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

int cij(int i, int j);
void printA(int n);

int main(void) {
    int n;
    scanf("%d", &n);
    while (n != 0) {
        printA(n);
        scanf("%d", &n);
    }
}
```

```

    }
    system("pause");
    return 0;
}

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

void printA(int n) {
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (k = 0; k < 2 * (n - i - 1); k++)
            printf(" ");
        printf("1");
        for (j = 1; j <= i; j++) {
            printf("%4d", cij(i, j));
        }
        printf("\n");
    }
    printf("\n");
}

```

(3) 测试:

a) 测试数据:

输入:

2

10

11

0

b) 测试结果:

```
C:\Users\samsung\Desktop\C\Project1\Debug\Project1.exe
2
 1
1 1
10
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
11
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
0
请按任意键继续...
```

图2-8 编程题3用测试数据得到的测试图

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

解答:

(1) 解题思路:

1. 输入要翻转的正整数 n ;

2. 若 n 不为 0，则开始循环:

2.1 若 n 不为个位数，则开始循环:

2.1.1 取出 n 的最后一位 a;

2.1.2 n/10 取整数;

2.1.3 输出 a;

2.2 打印 n 并换行;

3.结束。

(2) 程序清单

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

```
int main(void) {  
    unsigned int m;  
    scanf("%u", &m);  
    while (m != 0) {  
        unsigned int a;  
        while (m / 10 != 0) {  
            a = m % 10;  
            m /= 10;  
            printf("%d", a);  
        }  
        printf("%d\n", m);  
        scanf("%u", &m);  
    }  
    return 0;  
}
```

3) 测试

a) 测试数据:

123

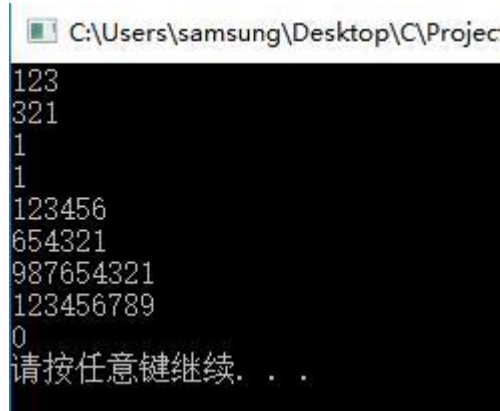
1

123456

987654321

0

b) 测试结果:



```
C:\Users\samsung\Desktop\C\Projec
123
321
1
1
123456
654321
987654321
123456789
0
请按任意键继续. . .
```

图 2-9 编程题 3 用测试数据得到的测试图

(5) 选做题: 用牛顿迭代法求方程的根: $3x^3-4x^2-5x+13=0$ 。

解答:

(1) 解题思路

使用牛顿迭代法进行循环, 直到 x 的精度满足要求

(2) 程序清单

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

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

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

```
    double x, y, dy, d;
```

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

```
    do {
```

```
        y = 3 * x * x * x - 4 * x * x - 5 * x + 13;
```

```
        dy = 9 * x * x - 8 * x - 5;
```

```
        d = y / dy;
```

```
        x -= d;
```

```
    } while (fabs(d) > 1e-6);
```

```
    printf("x=%lf\n", x);
```

```
    return 0;
```

```
}
```

(3) 测试

a) 测试数据

1

b) 测试结果

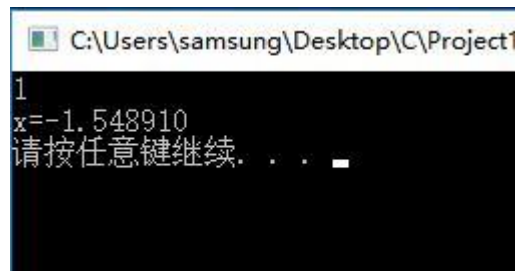


图 2-10 选做题用测试数据得到的测试图

2.3 自设题

(1) 自设实验题目：编写一个程序，提示用户输入大写字母，并按照下面金字塔型的格式打印字母：

A

ABA

ABCBA

ABCD CBA

ABCDEDCBA

(2) 实验目的：利用这种嵌套循环来练习流程控制的能力，达到提高水平的效果。

(3) 题目分析：由于是对字母操作，显然要用 ASCII 码来操作。另一方面，应该用嵌套循环来打印字母，其中内层循环有三个，分别是打印空格，以升序打印字母，以倒序打印字母。

(4) 实验程序：

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

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

```
    char ch;
```



```

printf("请输入一个大写字母: ");
while (scanf("%c", &ch) != EOF) {
    int n = ch - 'A' + 1;
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++)
            printf(" ");
        for (j = 0; j < i + 1; j++)
            printf("%c", 'A' + j);
        for (j = 0; j < i; j++)
            printf("%c", 'A' + i - 1 - j);
        printf("\n");
    }
}
return 0;
}

```

(5) 实验测试

a) 测试数据

E

G

D

K

b) 测试结果

```
C:\Users\samsung\Desktop\C\Project
请输入一个大写字母: E
  A
 ABA
ABCBA
ABDCBA
ABCDEDCBA
G
  A
  ABA
  ABCBA
  ABCDCBA
  ABCDEDCBA
  ABCDEFEDCBA
  ABCDEFGFEDCBA
D
  A
  ABA
  ABCBA
  ABCDCBA
K
      A
      ABA
      ABCBA
      ABCDCBA
      ABCDEDCBA
      ABCDEFEDCBA
      ABCDEFGFEDCBA
      ABCDEFGHGFEDCBA
      ABCDEFGHIHGFEDCBA
      ABCDEFGHIJHGFEDCBA
      ABCDEFGHIJKJHGFEDCBA
```

图 2-10 自设题用测试数据得到的测试图

2.4 实验小结

通过这次实验，很好的练习了 C 语言的循环和分支，对于流程控制方面有了更深入的理解。

实验 3 函数与程序结构实验

3.1 实验目的

(1) 熟悉和掌握函数的定义、声明；函数调用与参数传递方法；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

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)第五行 for(k=1;k<6;k++)中的 k<6 错误, 正确形式为:

scanf("%d", &n); for (k = 1; k <= n; k++);

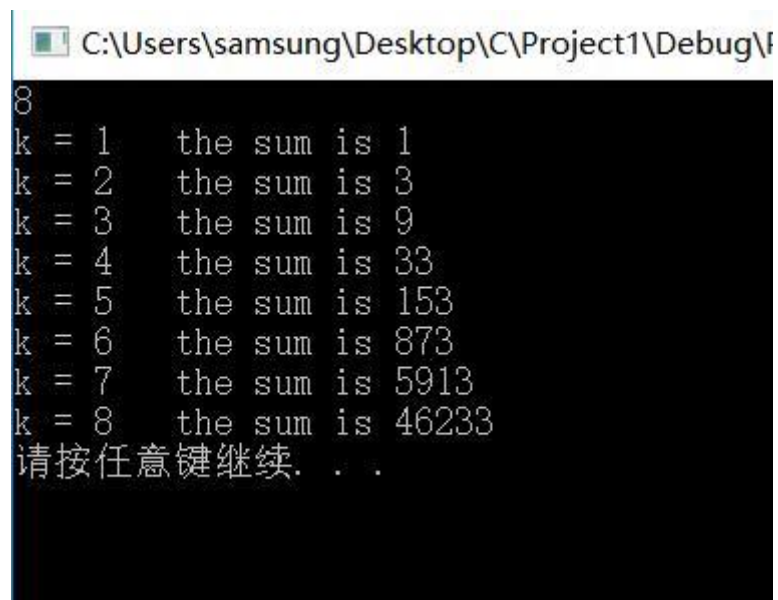
2)第十二行 fac 没有赋初始值, 正确形式为:

long fac = 1;

3)第十四行的 for 循环缺少大括号, 正确形式为:

```
for (i = 1; i <= n; i++) {
    fac *= i;
    s += fac;
}
```

错误修改后运行结果:



```
C:\Users\samsung\Desktop\C\Project1\Debug\I
8
k = 1    the sum is 1
k = 2    the sum is 3
k = 3    the sum is 9
k = 4    the sum is 33
k = 5    the sum is 153
k = 6    the sum is 873
k = 7    the sum is 5913
k = 8    the sum is 46233
请按任意键继续. . .
```

图3-1 错误改正之后的测试图

3.2.2 源程序修改替换

(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) 解答：修改后的程序如下所示：

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

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

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

```
    int k;
```

```
    int n;
```

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

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

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

```
    return 0;
```

```
}
```

```
long sum_fac(int n) {
```

```
    long s = 1;
```

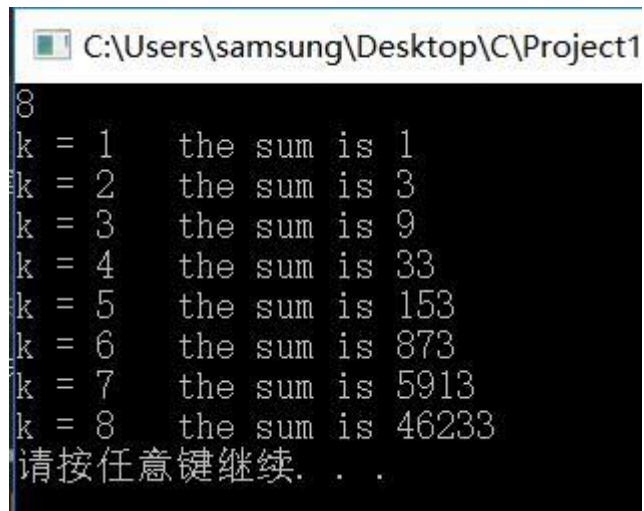
```
    int i;
```

```
    for (i = n; i >= 1; i--) {
```

```
        s = s * i + 1;
```

```
    }
```

```
    return s - 1;
```



```
C:\Users\samsung\Desktop\C\Project1
8
k = 1    the sum is 1
k = 2    the sum is 3
k = 3    the sum is 9
k = 4    the sum is 33
k = 5    the sum is 153
k = 6    the sum is 873
k = 7    the sum is 5913
k = 8    the sum is 46233
请按任意键继续. . .
```

```
}
```

图 3-2 程序替换之后的测试图

(2) 解答：修改后的程序如下所示：

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

```
double sum_fac(int n);
```

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

```
    int k;
```

```
    double n;
```

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

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

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

```
    return 0;
```

```
}
```

```
double sum_fac(int n) {
```

```
    double s = 1;
```

```
    double i;
```

```
    for (i = n; i >= 1; i--) {
```

```
        s = s / i + 1;
```

```

    }
    return s - 1;
}

```



```

8
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
k = 7    the sum is 1.718254
k = 8    the sum is 1.718279
请按任意键继续. . .

```

图 3-3 程序替换之后的测试图

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=∑`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？
- (2) 从 fibonacci 函数返回后光条停留在哪个语句上？
- (3) 进入 fibonacci 函数，watch 窗口显示的是什么？
- (4) 当 i=3，从调用 fibonacci 函数到返回，n 值如何变化？

解答：

(1)

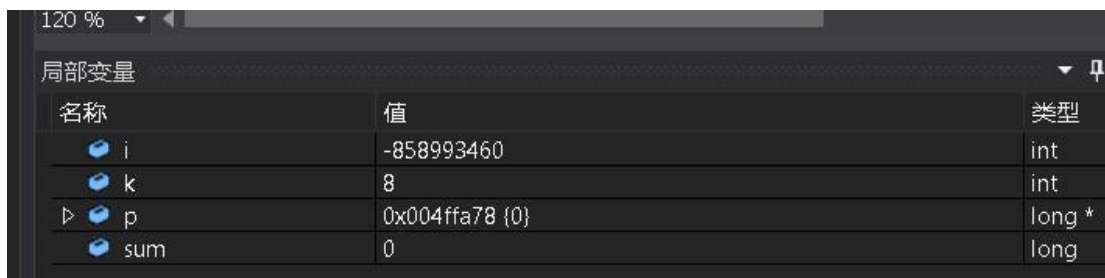


图 3-4 (1) 问对应的截图

(2)

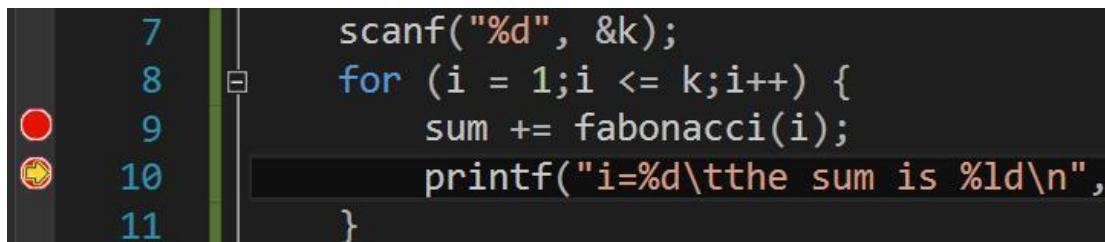


图 3-5 (2) 问对应的截图

(3)

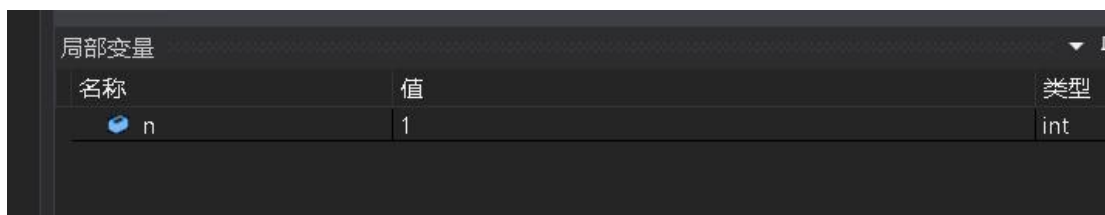


图 3-6 (3) 问对应的截图

(4)


局部变量		
名称	值	类型
 n	3	int

图 3-7 （4）问对应的截图 1


局部变量		
名称	值	类型
 n	2	int

图 3-8 （4）问对应的截图 2


局部变量		
名称	值	类型
 n	1	int

图 3-9 （4）问对应的截图 3

3.2.4 程序设计

（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：辗转相除法

1.输入 a, b

2.记 max 为 a, b 中较大的数，min 为 a, b 中较小的数。

3.1 若 min 能整除 max，则最大公约数为 min

3.2 否则，令 max 为 min，min 为 max 除以 min 的余数，进行步骤 3.

（2）程序清单：

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

```
int getMaxCommon(int a, int b);
```

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

```
    int a, b;
```

```

scanf("%d %d", &a, &b);
while (a != 0) {
    int min, max;
    max = (a >= b)? a : b;
    min = (a >= b)? b : a;
    printf("%d\n", getMaxCommon(max, min));
    scanf("%d %d", &a, &b);
}
return 0;
}

```

```

int getMaxCommon(int a, int b) {
    int r;
    r = a % b;
    if (r == 0)
        return b;
    else
        return getMaxCommon(b, r);
}

```

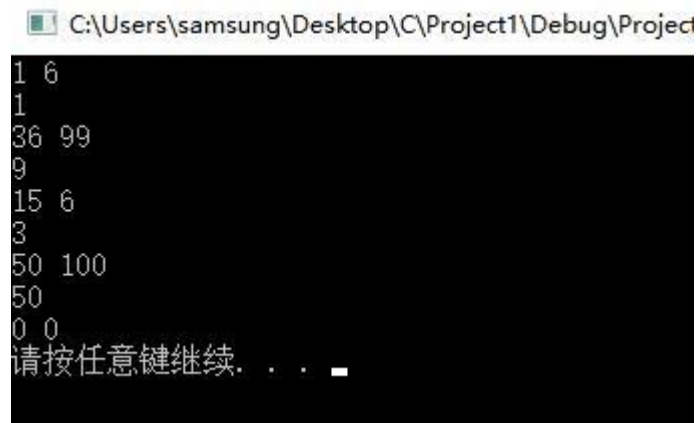
(3) 测试:

(a) 测试数据:

表 3-1 编程题 1 的测试数据

测试组数	测试数据	理论结果	实际结果
1	1, 6	1	1
2	36, 99	9	9
3	15, 6	3	3
4	50, 100	50	50

(b) 测试结果:



```
C:\Users\samsung\Desktop\C\Project1\Debug\Project1.exe
1 6
1
36 99
9
15 6
3
50 100
50
0 0
请按任意键继续. . . _
```

图 3-10 编程题 1 用四组测试数据得到的测试图

(2) 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

1. 用一个函数输出素数表，并用一个数组储存；
2. 用一个函数遍历素数表，找出是否有符合条件的等式，若有则输出。

(2) 程序清单

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

```
void getPrimes(int n, int *primes);
```

```
void printAns(int a, int b, int *primes);
```

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

```
    int begin, end;
```

```
    int primes[25];
```

```

    int i;
    getPrimes(100, primes);
    scanf("%d %d", &begin, &end);
    while (begin != 0) {
        printAns(begin, end, primes);
        scanf("%d %d", &begin, &end);
    }
    return 0;
}

```

```

void getPrimes(int n, int *primes) {
    int i, j, k = 0;
    int flag;
    for (i = 2; i <= n; i++) {
        flag = 1;
        for (j = 2; j <= (i / 2); j++) {
            if (i % j == 0)
                flag = 0;
        }
        if (flag == 1)
            primes[k] = i, k++;
    }
}

```

```

void printAns(int a, int b, int *primes) {
    int i = 0;
    int m, n;
    while (i < a)
        i += 2;
    for (; i <= b; i += 2) {

```

```

int m, n;
int flag = 0;
for (m = 0; m < 25; m++) {
    if (flag == 1)
        continue;
    for (n = m; n < 25; n++) {
        if (flag == 1)
            continue;
        if (primes[m] + primes[n] == i) {
            printf("%d=%d+%d\n", i, primes[m], primes[n]);
            flag = 1;
        }
    }
}
printf("\n");
}

```

(3) 测试

a) 测试数据

6 10

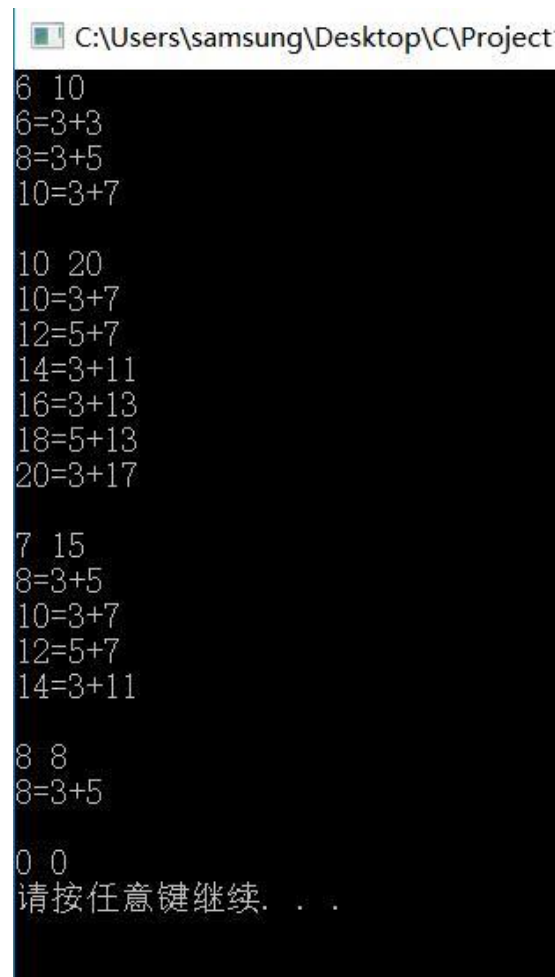
10 20

7 15

8 8

0 0

b) 测试结果



```
C:\Users\samsung\Desktop\C\Project>
6 10
6=3+3
8=3+5
10=3+7

10 20
10=3+7
12=5+7
14=3+11
16=3+13
18=5+13
20=3+17

7 15
8=3+5
10=3+7
12=5+7
14=3+11

8 8
8=3+5

0 0
请按任意键继续. . .
```

图 3-11 编程题 2 用测试数据得到的测试图

(3) 选做题:

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。

解答:

(1) 源码:

1) file.h

```
#include <stdio.h>
extern int x, y; /* 外部变量的引用性说明 */
extern char ch; /*
```

2) file1.c

```
#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 */
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

#include "file.h"

void func1(void)
{
    x++;
    y++;
    ch++;

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

```

(2) 测试:

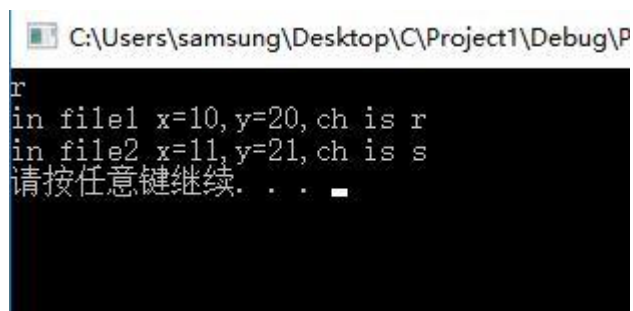


图 3-选做题生成程序运行的测试图

3.3 自设题

(1) 自设实验题目: 编写一个函数, 用逆序数定义来计算六阶以内行列式的值。

(2) 实验目的: 通过对算法的分析和函数参数的传递, 来进一步加强对函数编程的处理。

(3) 题目分析: 我们需要两个函数, 其中一个是计算逆序数, 另一个用来计算行列式的值, 另外在求值函数里面应该有一个能导出行列式每一行选一个数的全排列的部分。

(4) 实验程序:


```

#include <stdio.h>

#include <windows.h>

#include <math.h>

/*定义全局变量 sum,用以储存行列式的值*/
int sum = 0;

/*移动光标*/
void goto_xy(int x, int y);

/*计算行列式求和中每一项的符号*/
int getp(int *p, int n);

/*计算行列式的值*/
void getA(int *matrix, int * p, int counter, int n);//计算行列式的值

int main(void) {
    /*定义 n 表示矩阵的阶数*/
    int n;
    int i, j;
    /*定义一个 6*6 的二维数组储存矩阵*/
    int matrix[6][6];
    /*定义一个长度为 6 的数组，分别储存从矩阵每一行取出的一个数*/
    int p[6];
    /*输入矩阵的阶数*/
    printf("请选择您要计算的矩阵阶数（2~6）：");
    scanf_s("%d", &n);
    /*输入矩阵*/
    printf("\n 请输入您要计算的矩阵 A");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {

```

```

        /*控制光标的位置*/
        goto_xy(j * 4 + 1, 4 + i * 2);
        scanf_s("%d", &matrix[i][j]);
    }
}

printf("\n");
/*调用函数 getA，得出行列式的值*/
getA(matrix, p, 0, n);
/*输出*/
printf("以矩阵 A 中的元素构成的行列式的值是:  ");
printf("%d\n", sum);
system("pause");
return 0;
}

```

```

void goto_xy(int x, int y) {
    HANDLE hOut;
    hOut = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD pos = { x,y };
    SetConsoleCursorPosition(hOut, pos);
}

```

```

int getp(int *p, int n) {
    int i, j, k = 0;
    /*计算逆序数*/
    for (i = 0; i < n; i++) {
        for (j = 0; j < i; j++) {
            /*逆序数的定义：当某两个元素的先后次序与标准次序(1~n)不同
            时，就说有 1 个逆序。*/
            if (p[j] > p[i])

```

```

        k++;
    }
}
/*计算(-1)^k, 其中 k 为逆序数*/
if ((k % 2) == 0)
    return 1;
else
    return -1;
}

```

```

void getA(int *matrix, int *p, int counter, int n) {
    int i, j;
    /*当*/
    if (counter == n) {
        int ans = 1;
        /*根据序号的排列, 计算出矩阵中对应项的乘积*/
        for (i = 0; i < n; i++)
            ans *= *(matrix + i * 6 + p[i] - 1);
        /*判断每组乘积的符号, 并相加*/
        sum += ans * getp(p, n);
    }
    else {
        /*递归法导出 1~n 的全排列*/
        for (i = 1; i <= n; i++) {
            int ok = 1;
            for (j = 0; j < counter && ok; j++) {
                /*检测数 i 是否出现*/
                if (p[j] == i)
                    ok = 0;
            }

```

```

if(ok) {
    /*对数的排列赋值*/
    p[counter] = i;
    /*递归调用*/
    getA(matrix, p, counter + 1, n, 6);
}
}
}
}
}

```

(5) 实验测试:

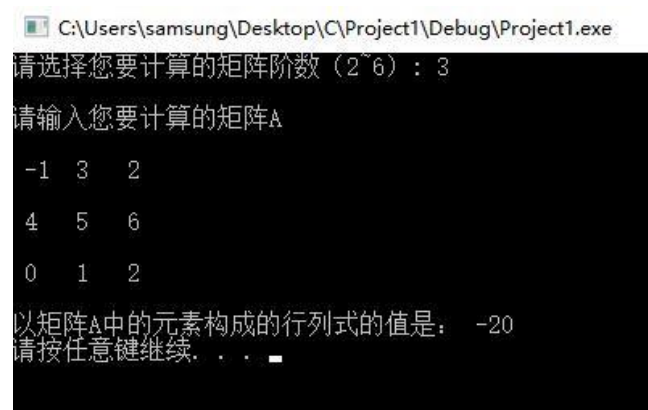


图 3-自设题用测试数据得到的测试图

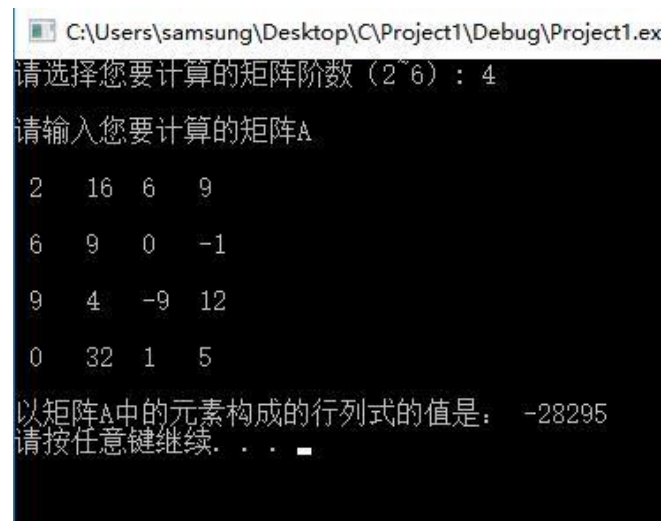


图 3-自设题用测试数据得到的测试图



图 3-自设题用测试数据得到的测试图

3.4 实验小结

本次实验是函数应用，其中对我帮助最大的是选做题关于多文件的处理，之前还没有遇到过这种几个文件的程序，这次好好的熟悉了这种程序的编写方法，加深了印象。

实验 4 编译预处理实验

4.1 实验目的

- (1) 掌握文件包含、宏定义、条件编译、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

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) 第一处：第四行应添加一个变量用于暂时储存：

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

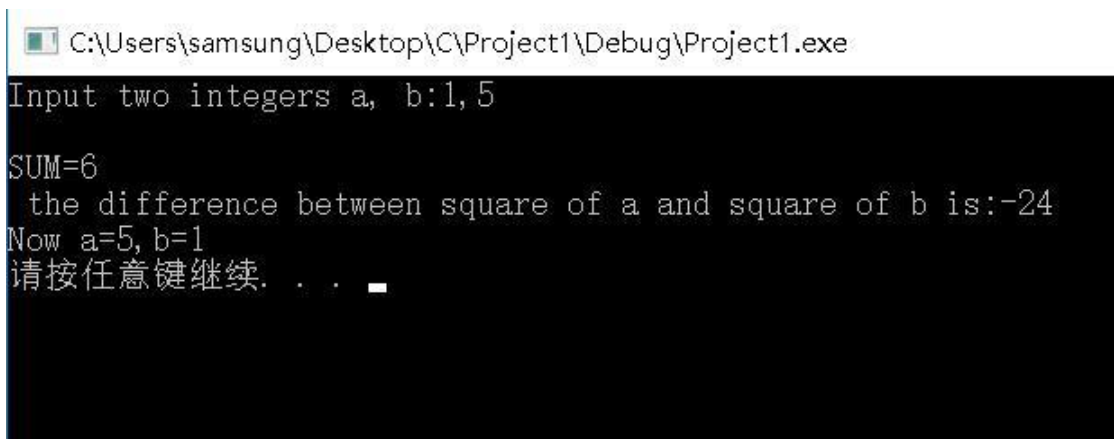
2) 第二处：第十二行 Printf 首字母应小写。正确形式为：

```
printf("\nNow a=%d,b=%d\n", a, b);
```

3) 第三处：第二第三行的定义里面应该加上括号，正确形式为：

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

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



```
C:\Users\samsung\Desktop\C\Project1\Debug\Project1.exe
Input two integers a, b:1,5
SUM=6
the difference between square of a and square of b is:-24
Now a=5, b=1
请按任意键继续. . . _
```

（2）错误修改后运行结果

图 4-1 错误改正之后的测试图

4.2.2 源程序修改替换

下面是用函数实现求三个数中最大数、计算两数之和的程序，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;
}
```

(1)解答:

1) 替换后的程序如下:

```
#include <stdio.h>
#include <windows.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);  
    _max(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);  
    sum(d, e);  
    printf("\nthe sum of them is   %f\n", sum(d, e));  
    system("pause");  
}
```

```
int _max(int x, int y, int z)
```

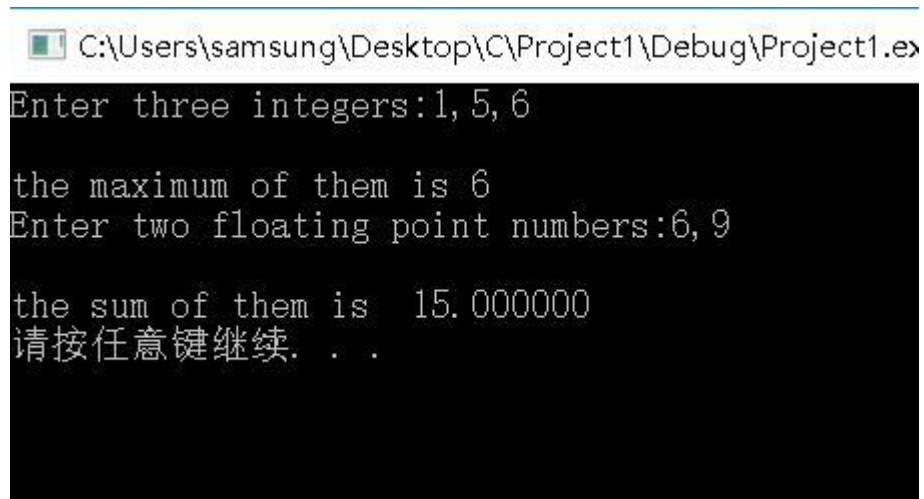
```
{  
    int t;  
    if (x>y)  
        t = x;  
    else  
        t = y;  
    if (t<z)  
        t = z;  
    return t;  
}
```

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

```

{
    return x + y;
}

```



2) 测试:

图 4-2 源程序替换之后的测试图

(2) 解答:

1) 替换后的程序如下:

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

```
#define max(x,y,z)    int t;if (x>y)t=x;else t=y;if (t<z)t=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);
    max(a, b, c);
    printf("\nthe maximum of them is %d\n", t);
}

```

```

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

```

```

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

```

2) 测试结果:

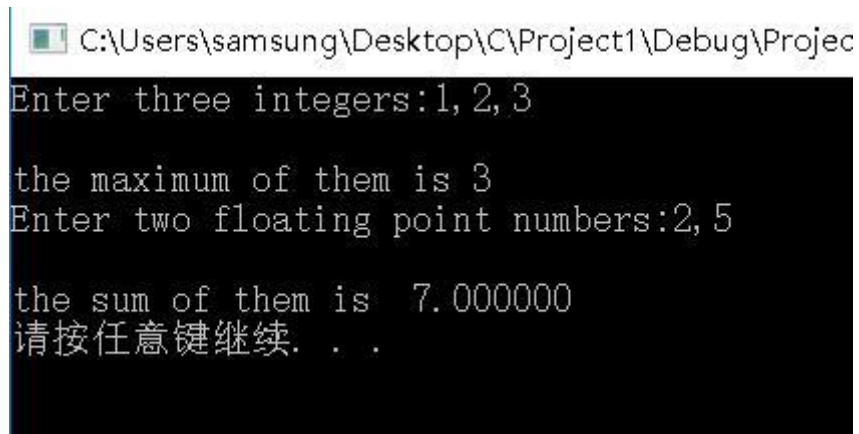


图 4-3 源程序替换之后的测试图

4.2.3 跟踪调试题

下面程序利用 R 计算圆的面积 s，以及面积 s 的整数部分。

```

#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；
- 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`。

解答：1) 修改后的程序如下：

```

#include <stdio.h>
#include <assert.h>
#define R
int integer_fraction(float x);
void main(void)
{
    float r, s;
    int s_integer = 0;
    printf("input a number: ");
    scanf("%f", &r);

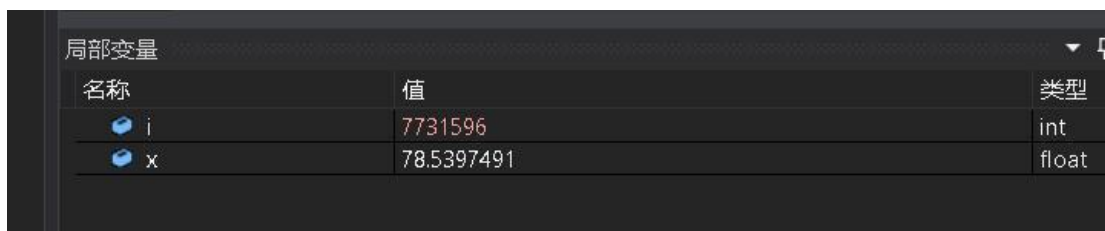
```

```

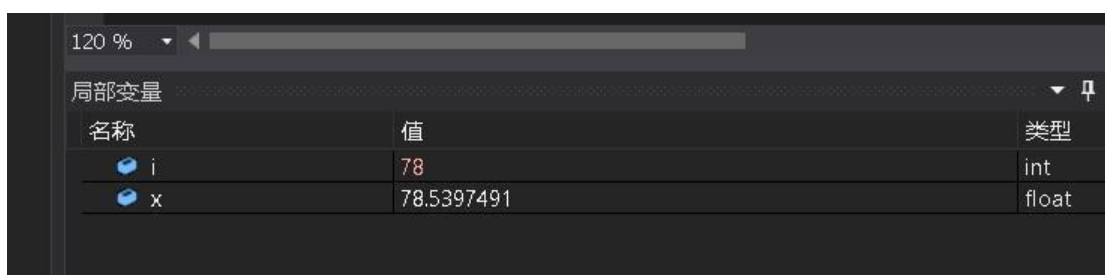
#ifdef R
    s = 3.14159*r*r;
    printf("area of round is: %f\n", s);
    s_integer = integer_fraction(s);
    printf("the integer fraction of area is %d\n", s_integer);
    assert((s - s_integer)<1.0);
#endif
}

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

```



名称	值	类型
i	7731596	int
x	78.5397491	float



名称	值	类型
i	78	int
x	78.5397491	float

2)

图 4-3 程序进入函数时的测试图
图 4-4 程序离开函数时的测试图

4.2.4 程序设计

(1) 三角形的面积是 $area = \sqrt{s(s-a)(s-b)(s-c)}$, 其中 $s = (a+b+c)/2$, a,b,c 为三角形的三边, 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. 定义一个宏为 s, 用以计算三角形的半周长

2. 定义一个宏为 area, 用以计算三角形的面积

2) 程序清单:

```
#include <stdio.h>

#include <math.h>

#define s(a, b, c) (a + b + c)

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

int main(void) {
    int a, b, c, s;
    double area;
    while(scanf("%d %d %d", &a, &b, &c) != EOF) {
        s = s(a, b, c) / 2;
        area = area(s, a, b, c);
        printf("%d %lf\n", s, area);
    }
    return 0;
}
```

3) 测试:

a) 测试数据

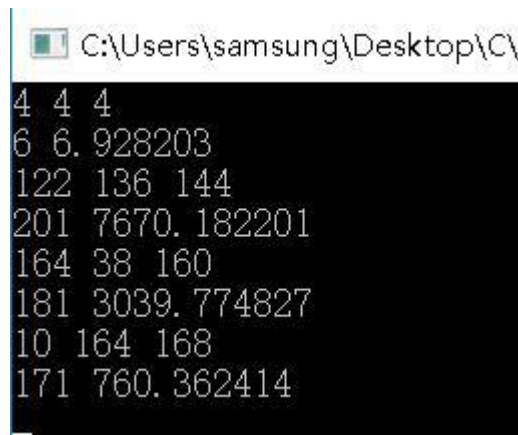
4 4 4

122 136 144

164 38 160

10 164 168

b) 测试结果:



```
C:\Users\samsung\Desktop\C\  
4 4 4  
6 6.928203  
122 136 144  
201 7670.182201  
164 38 160  
181 3039.774827  
10 164 168  
171 760.362414
```

图 4-5 编程题 1 用测试数据得到的测试图

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

解答：1) 解题思路：

1. 用一个数组储存输入的字符串
2. 检验该数组的第一个元素对应的 ASCII 码是否可以被 2 整除，若可以，则进行 2-1，否则进行 2-2
- 2-1: 直接输出该数组
- 2-2: 变换大小写之后输出该数组

2) 程序清单：

```
#include <stdio.h>  
  
#include <string.h>  
  
int main(void) {  
    int n, i;  
    scanf("%d", &n);  
    getchar();
```

```

for (i = 0; i < n; i++) {
    int j;
    char ch[100];
    fgets(ch, 100, stdin);
    int CHANGE = ch[0] % 2;
    int len = strlen(ch);
    if (!CHANGE) {
        for (j = 0; j < len; j++)
            putchar(ch[j]);
    }
    else {
        for (j = 0; j < len; j++) {
            if (ch[j] >= 'A' && ch[j] <= 'Z')
                putchar(ch[j] + 'a' - 'A');
            else if (ch[j] >= 'a' && ch[j] <= 'z')
                putchar(ch[j] - 'a' + 'A');
            else
                putchar(ch[j]);
        }
    }
}
return 0;
}

```

3) 测试:

a) 测试数据:

3

Hello!world#

abcd0j\$%#01_sd

Uq;V42?/\Ziggbn

b) 测试结果:



```
C:\Users\samsung\Desktop\C\Project
3
Hello!world#
Hello!world#
abcd0j$%#01_sd
ABCD0J$%#01_SD
Uq;V42?/\ZiggbIn
uQ;v42?/\zIGGBIN
请按任意键继续. . .
```

图 4-7 编程题 2 用测试数据得到的测试图

4.3 实验小结

本次实验练习宏定义的处理，还是亲身体会了下它与函数的区别的，另外本次实验可以说是目前最简单的一次吧，完成的也挺快的。

实验 5 数组实验

5.1 实验目的

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

5.2 实验内容

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）第一处：第五行 `void shot(int [], int);`应该定义在 `main` 函数外侧。

2）第二处：第七行 `sort(a[0], 10);`应该传递 `a[0]`的地址，正确形式为：

`sort(a, 10);`

3）第三处：`sort` 函数得出的结果是降序排列，第十八行应该是大于号，正确形式为：

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

（2）错误修改后运行结果：

```

panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc 1.c -o 001
panyue@Master ~/文档 ./001
0      3      5      13     17     27     32     39     46     55
panyue@Master ~/文档

```

图 5-1 错误修改后得到的测试图

5.2.2 源程序修改替换

（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? : ; /* 将报数为 N 的人的编号存入数组 b */
        if(j)
            for(k = --j; k < i; k++) /* 压缩数组 a，使报数为 N 的人出圈 */
                ;
    }

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

```

解答：（1）修改后的程序如下：

```

#include <stdio.h>

#define M 10

#define N 3

```

```

int main(void) {
    /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int a[M], b[M];
    int i, j, k;
    /* 对圈中人按顺序编号 1—M */
    for(i = 0; i < M; i++)
        a[i] = i + 1;
    /* 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(i = M, j = 0; i > 1; i--) {
        /* 1 至 N 报数 */
        for(k = 1; k <= N; k++)
            /* 最后一个人报数后第一个人接着报，形成一个圈 */
            if(++j > i - 1)
                j = 0;
        /* 将报数为 N 的人的编号存入数组 b */
        b[M-i] = j? a[j - 1] : a[M - i];
        if(j)
            /* 压缩数组 a，使报数为 N 的人出圈 */
            for(k = --j; k < i - 1; k++)
                a[k] = a[k + 1];
    }
    /* 按次序输出出圈人的编号 */
    for(i = 0; i < M - 1; i++)
        printf("%6d", b[i]);
    /* 输出圈中最后一个人的编号 */
    printf("%6d\n", a[0]);
    return 0;
}

```

(2) 测试:



```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc 2.c -o 001
panyue@Master ~/文档 ./001
3 6 9 2 7 1 8 5 10 4
panyue@Master ~/文档
```

图 5-2 源程序修改替换后得到的测试图

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

解答：1) 修改后的程序如下：

```
#include <stdio.h>

#define M 10
#define N 3

int main(void) {
    /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号，数组 c 存放圈中的
    的标记 */
    int a[M], b[M], c[M];
    int i, j, k;
    /* 对圈中人按顺序编号 1—M，并将标记均记作 0 */
    for(i = 0; i < M; i++) {
        a[i] = i + 1;
        c[i] = 0;
    }
    /* 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(i = M, j = 0; i > 1; i--) {
    /* 1 至 N 报数 */
    for(k = 1; k <= N; k++) {
        /*若某人被标记则跳过他 */
        while(c[j])
            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 > M - 1)
            j = 0;
    }
    /* 将报数为N的人的编号存入数组 b, 并将出圈的人的标记置为 1
*/

    if(j) {
        b[M - i] = a[j - 1];
        c[j - 1] = 1;
    }
    else {
        b[M - i] = a[M - 1];
        c[M - 1] = 1;
    }
}
/* 按次序输出出圈人的编号 */
for(i = 0; i < M - 1; i++)
    printf("%6d", b[i]);
/* 输出圈中最后一个人的编号 */
for(i = 0; i < M - 1; i++)
    if(!c[i])
        printf("%6d\n", a[i]);
return 0;

```

}

(2) 测试:



```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~$ cd 文档
panyue@Master ~/文档$ gcc 3.c -o 001
panyue@Master ~/文档$ ./001
3      6      9      2      7      1      8      5      10     4
panyue@Master ~/文档$
```

图 5-3 源程序修改替换后得到的测试图

5.2.3 跟踪调试题

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

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

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

源程序：

```
#include<stdio.h>

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

int main(void)
{
    char a[50]="The adopted symbol is
",b[27]="abcdefghijklmnopqrstuvwxyz";

    strncat(a, b, 4);

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

    return 0;
```



```

}

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

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

```

解答：（1）各种情形下的调试图如下

```

(gdb) r
Starting program: /home/panyue/文档/a.out

Breakpoint 1, _strncat (s=0x7fffffffef0d0 "The adopted symbol is ",
    t=0x7fffffffef0b0 "abcdefghijklmnopqrstuvwxyz", n=4) at debug.c:13
13         int i = 0, j;
(gdb) p s
$1 = 0x7fffffffef0d0 "The adopted symbol is "
(gdb) p t
$2 = 0x7fffffffef0b0 "abcdefghijklmnopqrstuvwxyz"
(gdb) p i
$3 = 0

```

图 5-4 调试中当光标达到_strncat 时的调试图

```

(gdb) c
Continuing.

Breakpoint 2, _strncat (s=0x7fffffffef0d0 "The adopted symbol is ",
    t=0x7fffffffef0b0 "abcdefghijklmnopqrstuvwxyz", n=4) at debug.c:15
15         for(j = 0; j < n && t[j];)
(gdb) p i
$4 = 23

```

图 5-5 调试中当光标达到 for 语句时的调试图

```

(gdb) c
Continuing.

Breakpoint 3, _strncat (s=0x7fffffffef0d0 "The adopted symbol is ",
    t=0x7fffffffef0b0 "abcdefghijklmnopqrstuvwxyz", n=4) at debug.c:18
18     } (gdb) p s
$5 = 0x7fffffffef0d0 "The adopted symbol is "
(gdb) p t
$6 = 0x7fffffffef0b0 "abcdefghijklmnopqrstuvwxyz"

```

图 5-6 调试中当光标达到函数结尾大括号时的调试图

(2) 函数出错的原因：没有考虑消去原字符串 s 中的'\0'字符

(3) 修改后的代码：

```
#include <stdio.h>

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

int main(void) {
    char a[50]="The adopted symbol is
    ",b[27]="abcdefghijklmnopqrstuvwxyz";
    _strncat(a, b, 4);
    printf("%s\n",a);
    return 0;
}

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

(2) 测试：

```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ ➔ cd 文档
panyue@Master ~/文档 ➔ gcc debug.c -o 001
panyue@Master ~/文档 ➔ ./001
The adopted symbol is abcd
panyue@Master ~/文档 ➔ █
```

图 5-7 程序修改后的测试图

5.2.4 程序设计

编写并上机调试运行能实现以下功能的程序。

(1) 编写一个程序,从键盘读取数据,对一个 3×4 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) 解题思路:

- 1.以 i, j 循环输入矩阵
- 2.以 i, j 循环打印矩阵
- 3.以 j, i 循环打印矩阵

(2) 程序清单:

```
#include <stdio.h>

int main(void) {
    int matrix[3][4];
    int i, j;
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            scanf("%d", &matrix[i][j]);
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 4; j++)
            printf("%5d", matrix[i][j]);
        printf("\n");
    }
}
```

```

printf("\n");
for (j = 0; j < 4; j++) {
    for (i = 0; i < 3; i++)
        printf("%5d", matrix[i][j]);
    printf("\n");
}
return 0;
}

```

(3) 测试:

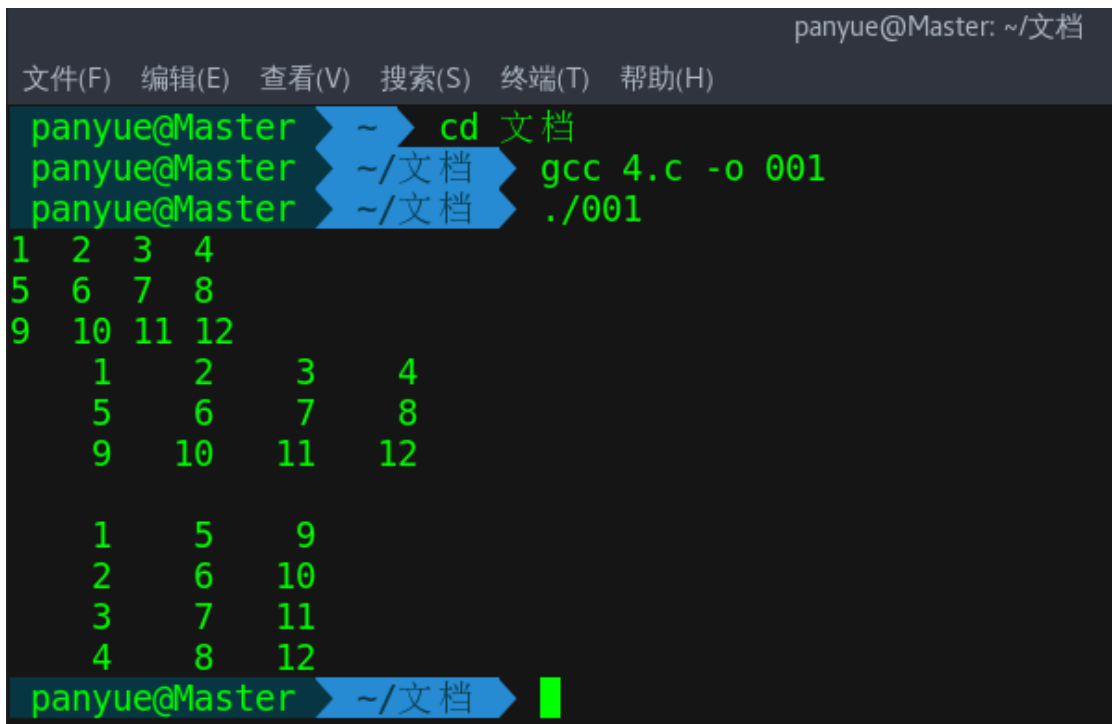
a)测试数据:

```

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

```

b) 测试结果:



```

panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc 4.c -o 001
panyue@Master ~/文档 ./001
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
panyue@Master ~/文档

```

图 5-7 编程题 1 用测试数据得到的测试图

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

解答：（1）解题思路：利用构造一个 $\text{mask} = 1 \ll \text{size} - 1$ ，每次让它和数字的最高位比较，若相同则将字符 ‘1’ 存入数组，否则将字符 ‘0’ 存入数组

（2）程序清单：

```
#include <stdio.h>

int main(void) {
    int n, i, j, size = sizeof(int) * 4;
    int num[100];
    char bin[100][100];
    scanf("%d", &n);
    int mask = 1 << (size - 1);
    for (i = 0; i < n; i++)
        scanf("%d", &num[i]);
    for (i = 0; i < n; i++) {
        for (j = 0; j < size; j++) {
            bin[i][j] = (num[i] & mask)? '1' : '0';
            num[i] <<= 1;
        }
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < size; j++)
            printf("%c", bin[i][j]);
        printf("\n");
    }
    return 0;
}
```

（3）测试：

a) 测试数据：

8

1 1024 26 0 -1 65 4 -1024

b) 测试结果:

```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc 5.c -o 001
panyue@Master ~/文档 ./001
8
1 1024 26 0 -1 65 4 -1024
000000000000000001
000001000000000000
00000000000011010
000000000000000000
111111111111111111
0000000001000001
00000000000000100
1111100000000000
panyue@Master ~/文档
```

图 5-8 编程题 2 用测试数据得到的测试图

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

解答: (1) 解题思路:

1. 排序使用冒泡排序算法, 并且在交换两数大小的同时, 也交换他们的名字
2. 查询使用二分查找算法, 若找到即输出结果。

(2) 程序清单:

```
#include <stdio.h>

void swap(char a[], char b[], int n);
```

```

void sort(int grades[], char name[][20], int n);

int search(int grades[], int x, int n);

int main(void) {
    int n, i;
    int N;
    char name[100][20];
    int grades[100];
    int num[100];
    scanf("%d", &n);
    getchar();
    for (i = 0; i < n; i++)
        scanf("%s %d", name[i], &grades[i]);
    sort(grades, name, n);
    for(i = 0; i < n; i++)
        printf("%-20s %d\n", name[i], grades[i]);
    printf("\n");
    scanf("%d", &N);
    for (i = 0; i < N; i++)
        scanf("%d", &num[i]);
    for (i = 0; i < N; i++) {
        int index;
        index = search(grades, num[i], n);
        if (index != -1)
            printf("%-20s %d\n", name[index], grades[index]);
        else
            printf("Not found!\n");
    }
    return 0;
}

```

```

void swap(char a[], char b[], int n) {
    int i;
    char ch[100];
    for (i = 0; i < n; i++)
        ch[i] = a[i];
    for (i = 0; i < n; i++)
        a[i] = b[i];
    for (i = 0; i < n; i++)
        b[i] = ch[i];
}

```

```

void sort(int grades[], char name[][20], int n) {
    int i, j, t;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++)
            if (grades[j] < grades[j + 1]) {
                t = grades[j], grades[j] = grades[j + 1], grades[j + 1] = t;
                swap(name[j], name[j + 1], 20);
            }
    }
}

```

```

int search(int grades[], int x, int n) {
    int front = 0, back = n - 1, middle;
    while (front <= back) {
        middle = (front + back) / 2;
        if (x < grades[middle])
            front = middle + 1;
        else if (x > grades[middle])

```



```

        back = middle - 1;
    else
        return middle;
    }
    return -1;
}

```

(3)测试:

a) 测试数据:

5

ZhangChuanChao 88

XiaoHong 95

XiaoMing 90

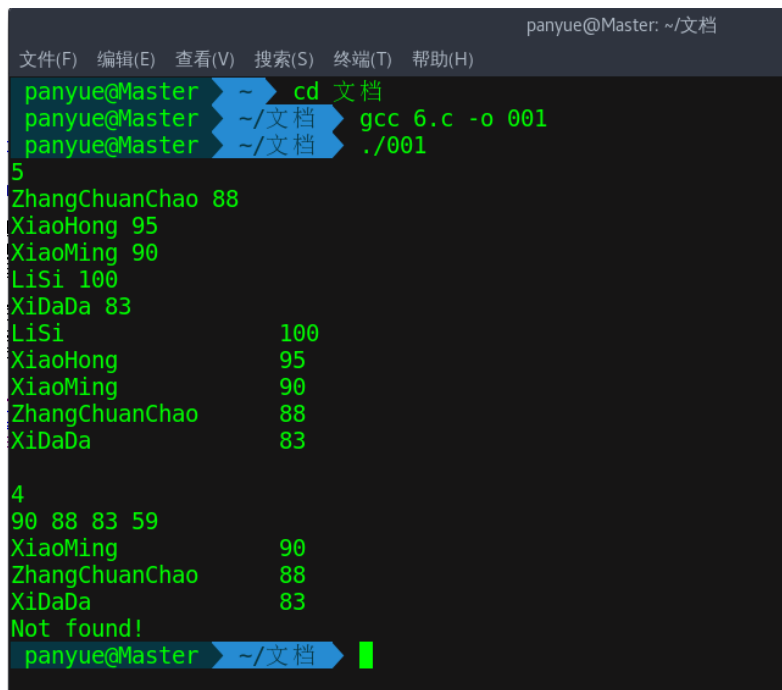
LiSi 100

XiDaDa 83

4

90 88 83 59

b) 测试结果:



```

panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ ➤ cd 文档
panyue@Master ~/文档 ➤ gcc 6.c -o 001
panyue@Master ~/文档 ➤ ./001
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!
panyue@Master ~/文档 ➤

```

图 5-9 编程题 3 用测试数据得到的测试图

(4) 选做题：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）解题思路：

- 1.遍历字符数组 `s` 中的前 `n` 个字符，什么也不做
- 2.将数组 `t` 中的字符赋值给数组 `s` 的第 `n` 个字符以后的位置

（2）程序清单：

```
#include <stdio.h>

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

int main(void) {
    char        a[50]="The        adopted        symbol        is
",b[27]="abcdefghijklmnopqrstuvwxyz";
    _strncat(a, b, 4);
    printf("%s\n",a);
    return 0;
}

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

（3）测试：

a) 测试数据：

`s = "The adopted symbol is "` `t = "abcdefghijklmnopqrstuvwxyz"`

b) 测试结果:



```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ ➔ cd 文档
panyue@Master ~/文档 ➔ gcc 8.c -o 001
panyue@Master ~/文档 ➔ ./001
The abcdefghijklmnopqrstuvwxyz
panyue@Master ~/文档 ➔
```

图 5-10 选做题 1 用测试数据得到的测试图

(5) 选做题: 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八个皇后, 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) 解题思路:

程序使用回溯算法, 核心由三个函数执行, 皇后的位置用一个数组储存

主要函数: 用来递归摆放皇后的位置, 若摆放到了第八个就调用打印函数

矛盾函数: 用来判断摆放皇后之后是否产生矛盾, 若是则在 `Ans` 数组中添加新的皇后, 否则退回。

打印函数: 按照既定的格式, 在每次得出一个解之后, 输出答案, 最后给出解的个数

(2) 程序清单:

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

```
int ways[8], sum = 0;
```

```
int abs(int x);
```

```
void printAns(void);
```

```
int isConflict(int n);
```

```
void getAnsOfEightQueen(int n);
```

```

int main(void) {
    getAnsOfEightQueen(0);
    printf("There are %d ways to solve this problem.\n", sum);
    return 0;
}

```

```

int abs(int x) {
    if (x >= 0)
        return x;
    else
        return -x;
}

```

```

void printAns(void) {
    int i;
    printf("Ans%d: \n", sum + 1);
    for (i = 0; i < 8; i++)
        printf("(%d,%d)", i + 1, ways[i] + 1);
    printf("\n");
    sum++;
}

```

```

int isConflict(int n) {
    int conflict = 0;
    int i;
    for (i = 0; i < n; i++)
        if (ways[i] == ways[n] || abs(ways[i] - ways[n]) == (n - i))
            conflict = 1;
    return conflict;
}

```

}

```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Ans79:
(1,6) (2,4) (3,7) (4,1) (5,8) (6,2) (7,5) (8,3)
Ans80:
(1,6) (2,8) (3,2) (4,4) (5,1) (6,7) (7,5) (8,3)
Ans81:
(1,7) (2,1) (3,3) (4,8) (5,6) (6,4) (7,2) (8,5)
Ans82:
(1,7) (2,2) (3,4) (4,1) (5,8) (6,5) (7,3) (8,6)
Ans83:
(1,7) (2,2) (3,6) (4,3) (5,1) (6,4) (7,8) (8,5)
Ans84:
(1,7) (2,3) (3,1) (4,6) (5,8) (6,5) (7,2) (8,4)
Ans85:
(1,7) (2,3) (3,8) (4,2) (5,5) (6,1) (7,6) (8,4)
Ans86:
(1,7) (2,4) (3,2) (4,5) (5,8) (6,1) (7,3) (8,6)
Ans87:
(1,7) (2,4) (3,2) (4,8) (5,6) (6,1) (7,3) (8,5)
Ans88:
(1,7) (2,5) (3,3) (4,1) (5,6) (6,8) (7,2) (8,4)
Ans89:
(1,8) (2,2) (3,4) (4,1) (5,7) (6,5) (7,3) (8,6)
Ans90:
(1,8) (2,2) (3,5) (4,3) (5,1) (6,7) (7,4) (8,6)
Ans91:
(1,8) (2,3) (3,1) (4,6) (5,2) (6,5) (7,7) (8,4)
Ans92:
(1,8) (2,4) (3,1) (4,3) (5,6) (6,2) (7,7) (8,5)
There are 92 ways to solve this problem.
```

```
void getAnsOfEightQueen(int n) {
    int i;
    for (i = 0; i < 8; i++) {
        ways[n] = i;
        if (!isConflict(n)) {
            if (n == 7)
                printAns();
            else
                getAnsOfEightQueen(n + 1);
        }
    }
}
```

(3) 测试:

图 5-11 选做题 2 用测试数据得到的测试图

5.3 实验小结

本次实验题目较多，不过也都很快的顺利完成，刚刚配置好了 arch linux 系统，这次所有编译采用命令行下的 gcc 编译器，调试采用命令行下的 gdb 编译器。数组实验涉及了不少指针的知识，很多内容还有待下一次实验来进一步熟悉。

实验 6 指针实验

6.1 实验目的

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

6.2 实验内容

6.2.1 源程序改错

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

```
1  #include "stdio.h"
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：应该给 p 赋初始值，让其指向一个浮点数 k，正确形式为：

在第四行之前加上一句 `float k, *p;`

（2）错误修改后运行结果：

```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc correct.c -o 001
panyue@Master ~/文档 ./001
5
5.000000
x panyue@Master ~/文档
```

图 6-1 错误修改后得到的测试图

6.2.2 源程序修改替换

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

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

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

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

```

(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）修改后的程序如下：

```
#include "stdio.h"
#include "string.h"
void main(void)
{
    char *(*p)(char *a,const char *b);
    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;
```

```

do a[i] = getchar();
    while (a[i++] != '\n');
    a[i - 1] = '\0';
printf("input the second string please!\n");
i=0;
do b[i] = getchar();
    while (b[i++] != '\n');
    b[i - 1] = '\0';
result = (*p)(a,b);
printf("the result is %s\n",result);

down:
    ;
}

```

(2) 测试:

```

panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ ➤ cd 文档
panyue@Master ~/文档 ➤ gcc changel.c -o 001
panyue@Master ~/文档 ➤ ./001
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.
x panyue@Master ~/文档 ➤

```

图 6-2 源程序修改替换后得到的测试图

6.2.3 跟踪调试题

```

#include "stdio.h"

#include "string.h"

```

```

void main(void)
{
    char *(*p)(char *a,const char *b);
    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;
    do a[i] = getchar();
    while (a[i++] != '\n');
    a[i - 1] = '\0';
    printf("input the second string please!\n");
    i=0;

```

```

do b[i] = getchar();
    while (b[i++] != '\n');
    b[i - 1] = '\0';
    result = (*p)(a,b);
    printf("the result is %s\n",result);
down:
    ;
}
#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)

```

(gdb) r
Starting program: /home/panyue/文档/a.out

Breakpoint 1, strcpy (s=0x7fffffff0e0 "\340\005@",
    t=0x7fffffff0a0 "there is a boat on the lake.") at debug.c:13
13      char *p = s;
(gdb) p s
$1 = 0x7fffffff0e0 "\340\005@"
(gdb) p *s
$2 = -32 '\340'

```

图 6-3 进入 strcpy 和返回 main 时 s 的值

(2) 修改后的程序如下：

```
#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);
```

```
}
```

运行结果如图：

```

(gdb) c
Continuing.
there is a boat on the lake.
[Inferior 1 (process 4828) exited with code 035]
(gdb) █

```

图 6-4 程序修正后的测试图

(3) 在 Linux shell 的 gdb 环境下不需要考虑这个问题

6.2.4 程序设计

(1) 一个长整型变量占 4 个字节，其中每个字节又分成高 4 位和低 4 位。试从该长整型变量的高字节开始，依次取出每个字节的高 4 位和低 4 位并以数字字符的形式进行显示。

解答：(1) 解题思路：去一个 mask 为 1111，每次循环取出该变量的四个字节，转换为十六进制。

(2) 程序清单：

```
#include <stdio.h>

void printHex(int a);

int main(void) {
    int N, i, j;
    int nums[100];
    unsigned int bit;
    scanf("%d", &N);
    for (i = 0; i < N; i++)
        scanf("%d", &nums[i]);
    for (i = 0; i < N; i++) {
        unsigned int mask = 15 << 28;
        for (j = 0; j < 8; j++) {
            bit = *(nums + i) & mask;
            printHex(bit >> (4 * (7 - j)));
            if (j < 7)
                printf(" ");
            mask >>= 4;
        }
    }
}
```

```

    }

    printf("\n");
}

return 0;
}

void printHex(int a) {
    if (a >= 0 && a <= 9)
        printf("%d", a);
    else if (a < 16)
        printf("%c", 'A' + a - 10);
}

```

(3) 测试:

a) 测试数据:

6

1 0 88888 90 -23 10

b) 测试结果:

```

panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc 3.c -o 001
panyue@Master ~/文档 ./001
6
1 0 88888 90 -23 10
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 1 5 B 3 8
0 0 0 0 0 0 5 A
F F F F F F E 9
0 0 0 0 0 0 0 A
panyue@Master ~/文档
panyue@Master ~/文档

```

图 6-5 编程题 1 用测试数据得到的测试图

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

解答：（1）解题思路：定义一个 `flag`，执行如下操作：

- 1.若要接下来一个字符不是空格，`flag = 0`；
- 2.若 `flag` 为 0，复制；
- 3.若刚刚复制的是空格，`flag = 1`；

（2）程序清单：

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

```
void space(char **line, int n);
```

```
int main(void) {  
    int n;  
    scanf("%d", &n);  
    getchar();  
    while (n != 0) {  
        int i;  
        char ch[100][80];  
        char *p[100];  
        for (i = 0; i < n; i++) {  
            fgets(ch[i], 100, stdin);  
            p[i] = ch[i];  
        }  
        space(p, n);  
        scanf("%d", &n);  
        getchar();  
    }  
    return 0;  
}
```

```

void space(char **line, int n) {
    int i;
    for (i = 0; i < n; i++) {
        char ans[1000];
        int flag = 0;
        int j = 0, s = 0;
        while (*(line + i) + j) != '\n') {
            if (*(line + i) + j) != ' ')
                flag = 0;
            if (flag == 0)
                *(ans + s++) = (*(line + i) + j);
            if (*(line + i) + j) == ' ')
                flag = 1;
            j++;
        }
        *(ans + s) = '\0';
        if (*(ans) != '\0')
            printf("%s\n", ans);
    }
    printf("\n");
}

```

(3) 测试:

a) 测试数据:

3

zhang chuan chao

xi da da

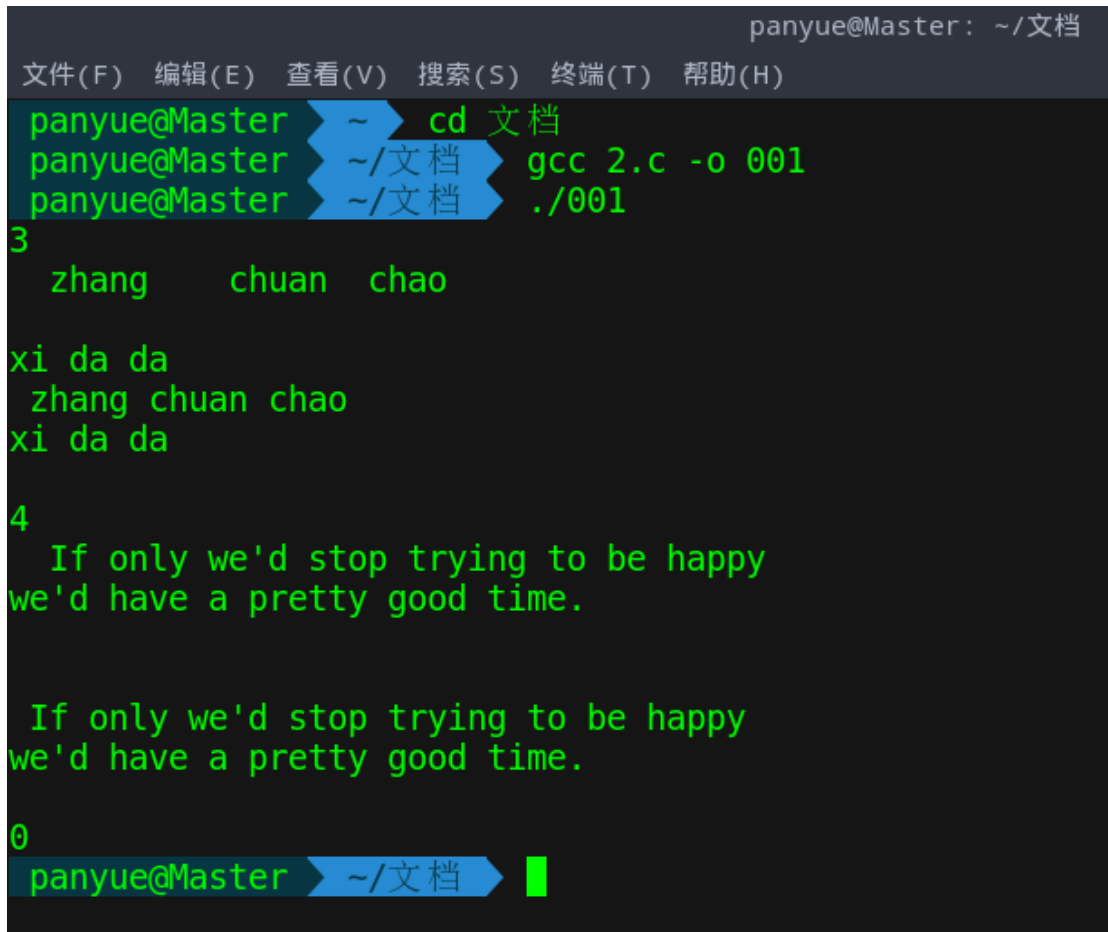
4

If only we'd stop trying to be happy

we'd have a pretty good time.

0

b) 测试结果:



```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 gcc 2.c -o 001
panyue@Master ~/文档 ./001
3
zhang chuan chao
xi da da
zhang chuan chao
xi da da
4
If only we'd stop trying to be happy
we'd have a pretty good time.

If only we'd stop trying to be happy
we'd have a pretty good time.
0
panyue@Master ~/文档
```

图 6-6 编程题 2 用测试数据得到的测试图

(3) 编写一个程序，输入 n 个整数，排序后输出。排序的原则可由命令行可选参数 `-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。

解答：(1) 程序清单：

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

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

```
void swap(int *x, int *y);
```

```
void sortup(int *nums, int n);  
void sortdown(int *nums, int n);
```

```
int main(int argc, char const *argv[])  
{  
    int n, i;  
    int nums[100];  
    char ch[] = "-d";  
    scanf("%d", &n);  
    for (i = 0; i < n; i++) {  
        scanf("%d", nums + i);  
    }  
    if (argc > 1)  
        if (!strcmp(argv[1], ch))  
            sortdown(nums, n);  
        else  
            sortup(nums, n);  
    else  
        sortup(nums, n);  
    for (i = 0; i < n - 1; i++)  
        printf("%d ", nums[i]);  
    printf("%d\n", nums[n - 1]);  
    return 0;  
}
```

```
void swap(int *x, int *y) {  
    int t;  
    t = *x;  
    *x = *y;  
    *y = t;  
}
```

```
}
```

```
void sortup(int *nums, int n) {  
    int i, j;  
    for (i = 0; i < n - 1; i++)  
        for (j = 0; j < n - i - 1; j++)  
            if (*(nums + j) < *(nums + j + 1))  
                swap(nums + j, nums + j + 1);  
}
```

```
void sortdown(int *nums, int n) {  
    int i, j;  
    for (i = 0; i < n - 1; i++)  
        for (j = 0; j < n - i - 1; j++)  
            if (*(nums + j) > *(nums + j + 1))  
                swap(nums + j, nums + j + 1);  
}
```

(2) 测试:

```
panyue@Master: ~/文档/c_demos  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
panyue@Master ~ cd 文档  
panyue@Master ~/文档 cd c demos  
panyue@Master ~/文档/c_demos gcc 4.c -o 001  
panyue@Master ~/文档/c_demos ./001  
6  
1 4 5 6 3 2  
6 5 4 3 2 1  
panyue@Master ~/文档/c_demos ./001 -d  
6  
1 4 5 6 3 2  
1 2 3 4 5 6  
panyue@Master ~/文档/c_demos
```

图 6-7 编程题 3 用测试数据得到的测试图

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

- 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) 解题思路: 直接利用指针操作计算

(2) 程序清单:

```
#include <stdio.h>

#define M 5 //课程
#define N 5 //学生

double classSum[M];

void getStuAverage(double *grades, char *stu);
void getClassAverage(double *grades, char *name);
void getNumOfLows(double *grades, char *name);
void getNumOffailersAndTops(double *grades, char *name);

int main(void) {
    int i, j;
    char name[M][10];
    char stu[N][10];
    double grades[N][M];
    for (i = 0; i < M; i++)
        scanf("%s", *(name + i));
    getchar();
    for (i = 0; i < N; i++) {
        scanf("%s", *(stu + i));
        for (j = 0; j < M; j++)
```

```

        scanf("%lf", *(grades + i) + j);
    }
    getStuAverage(*grades, *stu);
    getClassAverage(*grades, *name);
    getNumOfLows(*grades, *name);
    getNumOffailersAndTops(*grades, *name);
    return 0;
}

```

```

void getStuAverage(double *grades, char *stu) {
    int i, j;
    double sum[N];
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            *(sum + i) += *(grades + i * N + j);
        }
        printf("Average score of %s is %.2lf\n", stu + i * 10, *(sum + i) / N);
    }
}

```

```

void getClassAverage(double *grades, char *name) {
    int i, j;
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++) {
            *(classSum + i) += *(grades + j * N + i);
        }
        printf("Average score of %s is %.2lf\n", name + i * 10, *(classSum + i)
/ M);
    }
}

```

```

void getNumOfLows(double *grades, char *name) {
    int i, j;
    int lows[M] = {0, 0, 0, 0, 0};
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++) {
            if (*(grades + j * N + i) < *(classSum + i) / N)
                *(lows + i) += 1;
        }
    }
    for (i = 0; i < M; i++) {
        printf("Number of students lower than avg of %s is %d\n", name + i *
10, *(lows + i));
    }
}

```

```

void getNumOffailersAndTops(double *grades, char *name) {
    int i, j;
    int fails[M] = {0, 0, 0, 0, 0};
    int tops[M] = {0, 0, 0, 0, 0};
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++) {
            if (*(grades + j * N + i) < 60)
                *(fails + i) += 1;
            if (*(grades + j * N + i) >= 90)
                *(tops + i) += 1;
        }
    }
    for (i = 0; i < M; i++) {
        printf("Number of students %s fail is %d\n", name + i * 10, *(fails +

```



```

i));
    }
    for (i = 0; i < M; i++) {
        printf("Number of students %s perfect is %d\n", name + i * 10, *(tops
+ i));
    }
}

```

(3) 测试:

a) 测试数据:

A B C D E

Zhang

87 88 77 87 95

Li

88 98 100 48 75

Wang

85 68 95 47 59

Han

86 89 75 85 88

Gan

87 68 87 89 100

b) 测试结果:

```
panyue@Master: ~/文档
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ > cd 文档
panyue@Master ~/文档 > gcc 1.c -o 001
panyue@Master ~/文档 > ./001
A B C D E
Zhang
87 88 77 87 95
Li
88 98 100 48 75
Wang
85 68 95 47 59
Han
86 89 75 85 88
Gan
87 68 87 89 100
Average score of Zhang is 86.80
Average score of Li is 81.80
Average score of Wang is 70.80
Average score of Han is 84.60
Average score of Gan is 86.20
Average score of A is 86.60
Average score of B is 82.20
Average score of C is 86.80
Average score of D is 71.20
Average score of E is 83.40
Number of students lower than avg of A is 2
Number of students lower than avg of B is 2
Number of students lower than avg of C is 2
Number of students lower than avg of D is 2
Number of students lower than avg of E is 2
Number of students A fail is 0
Number of students B fail is 0
Number of students C fail is 0
Number of students D fail is 2
Number of students E fail is 1
Number of students A perfect is 0
Number of students B perfect is 1
Number of students C perfect is 2
Number of students D perfect is 0
Number of students E perfect is 2
panyue@Master ~/文档 >
```

图 6-8 编程题 4 用测试数据得到的测试图

(5) 选做题：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。

eg: 12345678912345678912.1234567891 + 98765432109876543210.0123456789

解答：(1) 程序清单：

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

```
#include <ctype.h>
```

```

#define N 21

#define M 11

void rightShift(int *a, int n);
void leftShift(int *a, int n);

int main(void) {
    int x[N], y[N], z[N + 1];
    int a[M], b[M], c[M + 1];
    int i, carry = 0, flag;
    for (i = 0; i < N + 1; i++)
        *(z + i) = 0;
    for (i = 0; i < M + 1; i++)
        *(c + i) = 0;

    i = N - 1;
    while (i >= 0 && isdigit(*(x + i) = getchar()))
        *(x + i--) -= '0';
    if (i >= 0)
        rightShift(x, i);
    i = M - 1;
    while (i >= 0 && isdigit(*(a + i) = getchar()))
        *(a + i--) -= '0';
    if (i >= 0)
        leftShift(a, i);
    i = N - 1;
    while (i >= 0 && isdigit(*(y + i) = getchar()))
        *(y + i--) -= '0';
    if (i >= 0)
        rightShift(y, i);

```

```

i = M - 1;
while (i >= 0 && isdigit(*(b + i) = getchar()))
    *(b + i--) -= '0';
if (i >= 0)
    leftShift(b, i);

for (i = 0; i < M; i++) {
    *(c + i) = *(a + i) + *(b + i) + carry;
    carry = (*(c + i) - *(c + i) % 10) / 10;
    *(c + i) %= 10, *(c + i) += '0';
}

for (i = 0; i < N; i++) {
    *(z + i) = *(x + i) + *(y + i) + carry;
    carry = (*(z + i) - *(z + i) % 10) / 10;
    *(z + i) %= 10, *(z + i) += '0';
}

*(z + N) = carry + '0';
flag = 0;
for (i = N; i >= 0; i--) {
    if ((flag == 0 && *(z + i) != '0') || i == 0)
        flag = 1;
    if (flag == 1)
        putchar(*(z + i));
}

putchar('.');
for (i = M; i >= 0; i--) {
    if (flag == 0 && *(c + i) != '0')
        flag = 1;
    if (flag == 1)
        putchar(*(c + i));
}

```

```

    }
    putchar('\n');
    return 0;
}

void rightShift(int *a, int n) {
    int k, len, sft;
    len = N - 1 - n;
    sft = N - len;
    for (k = 0; k < len; k++)
        *(a + k) = *(a + k + sft);
    for (k = len; k < N; k++)
        *(a + k) = 0;
}

```

```

void leftShift(int *a, int n) {
    int k, len, sft;
    len = M - 1 - n;
    sft = M - len;
    for (k = 0; k < sft; k++)
        *(a + k) = 0;
}

```

(2) 测试:

a) 测试数据:

12345678912345678912.1234567891

98765432109876543210.0123456789

b) 测试结果:

```
panyue@Master: ~/文档/c_demos
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 cd c_demos
panyue@Master ~/文档/c_demos gcc 5.c -o 001
panyue@Master ~/文档/c_demos ./001
12345678912345678912.1234567891+98765432109876543210.0123456789
1111111110222222222122.13580246800
panyue@Master ~/文档/c_demos
```

图 6-9 选做题 5 用测试数据得到的测试图

(6) 选做题：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) 程序清单：

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

char *minstr(const char *a, const char *b);
char *maxstr(const char *a, const char *b);

int main(void) {
    char *(*p[2])(const char *a, const char *b);
    char a[100], b[100], c[2][100], *d[2];
    int i;
    for (i = 0; i < 2; i++)
        d[i] = c[i];
    p[0] = minstr;
    p[1] = maxstr;
    printf("Please input the first string.\n");
    i = 0;
    do a[i] = getchar();
    while (a[i++] != '\n');
    a[i - 1] = '\0';
```

```

printf("Please input the second string.\n");
i = 0;
do b[i] = getchar();
    while (b[i++] != '\n');
    b[i - 1] = '\0';
    for (i = 0; i < 2; i++)
        d[i] = (*(p+i))(a,b);
    for (i = 0; i < 2; i++)
        printf("the result%d is %s\n", i, *(d + i));
    return 0;
}

```

```

char *minstr(const char *a, const char *b) {
    char *x = a;
    char *y = b;
    if (strcmp(a, b) >= 0)
        return y;
    else
        return x;
}

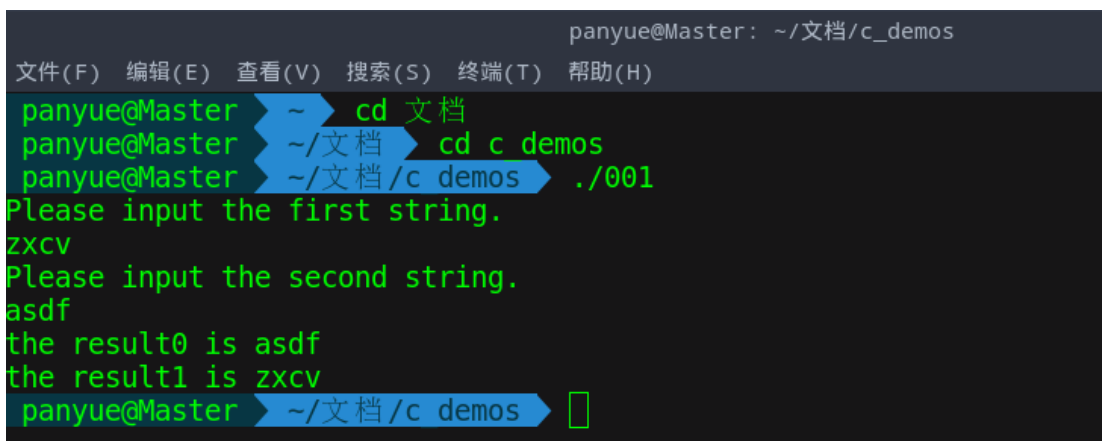
```

```

char *maxstr(const char *a, const char *b) {
    char *x = a;
    char *y = b;
    if (strcmp(a, b) >= 0)
        return x;
    else
        return y;
}

```

(2) 测试:



```
panyue@Master: ~/文档/c_demos
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ cd 文档
panyue@Master ~/文档 cd c demos
panyue@Master ~/文档/c demos ./001
Please input the first string.
zxcv
Please input the second string.
asdf
the result0 is asdf
the result1 is zxcv
panyue@Master ~/文档/c demos
```

图 6-10 选做题 6 测试数据得到的测试图

6.3 实验小结

本次实验很好的熟练了指针操作，主要是指向数组和指向函数的指针。在十六进制的那个题上面耽误了很多时间，因为没有明白右移位时补数字的情况：gcc 编译器在右移位时前面补的数字以符号位为准。

实验 7 结构与联合实验

7.1 实验目的

(1) 通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。

(2) 通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。

(3) 了解字段结构和联合的用法。

7.2 实验内容

7.2.1 表达式求值与程序验证

设有说明：

```
char u[]="UVWXYZ";
```

```
char v[]="xyz";
```

```
struct T{
```

```
    int x;
```

```
    char c;
```

```
    char *t;
```

```
}a[]={11, 'A', u}, {100, 'B', v}}, *p=a;
```

请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

序号	表达式	计算值	验证值
1	$(++p) \rightarrow x$	100	100
2	$p++, p \rightarrow c$	B	B
3	$*p++ \rightarrow t, *p \rightarrow t$	x	x
4	$* (++p) \rightarrow t$	x	x
5	$* ++p \rightarrow t$	V	V
6	$++ * p \rightarrow t$	V	V

测试：

```

panyue@Master: ~/cod
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~ ➔ cd code
panyue@Master ~/code ➔ cd c_demo
panyue@Master ~/code/c_demo ➔ gcc 001.c -o 001
panyue@Master ~/code/c_demo ➔ ./001
100
B
X
X
V
V
panyue@Master ~/code/c_demo ➔
```

图 7-1 由题目中的测试数据得到的测试图

7.2.2 源程序修改替换

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

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

源程序如下：

```
#include "stdio.h"

#include "stdlib.h"

struct s_list{
int data; /* 数据域 */
struct s_list *next; /* 指针域 */
};

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

void main(void)
{
    struct s_list *head=NULL,*p;
```

```

int s[]={1,2,3,4,5,6,7,8,0}; /* 0 为结束标记 */
create_list(head,s); /* 创建新链表 */
p=head; /*遍历指针 p 指向链头 */
while(p){
    printf("%d\t",p->data); /* 输出数据域的值 */
    p=p->next; /*遍历指针 p 指向下一结点 */
}
printf("\n");
}

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

```

(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）源程序无运行结果，因为头指针没有传到函数外部，应该将 headp 的地址传递进入函数，修改后的程序如下：

```
#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 */
}

```

(2) 修改后的程序如下:

```

#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);

int main(void)
{
    struct s_list *head=NULL, *p;

```

```

int s[]={1,2,3,4,5,6,7,8,0}; /* 0 为结束标记 */
create_list(&head,s); /* 创建新链表 */
p=head; /*遍历指针 p 指向链头 */
while(p){
    printf("%d\t",p->data); /* 输出数据域的值 */
    p=p->next; /*遍历指针 p 指向下一结点 */
}
printf("\n");
return 0;
}

void create_list(struct s_list **headp,int *p)
{
    struct s_list * loc_head=NULL,*tail,*t;
    if(p[0]==0) /* 相当于*p==0 */
        ;
    else { /* loc_head 指向动态分配的第一个结点 */
        tail=(struct s_list *)malloc(sizeof(struct s_list));
        tail->data=*p++; /* 对数据域赋值 */
        tail->next = NULL; /* 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 */
            t = (struct s_list *)malloc(sizeof(struct s_list));
            t->data = *p++;
            t->next = tail;
            tail = t;
        }
    }
    *headp = tail;
}

```

(3) 测试:

```
panyue@Master ~/code/c_demo gcc 002.c -o 001
panyue@Master ~/code/c_demo ./001
1      2      3      4      5      6      7      8
x panyue@Master ~/code/c_demo
```

图 7-2 源程序修改正确后的测试图

```
x panyue@Master ~/code/c_demo gcc 003.c -o 001
panyue@Master ~/code/c_demo ./001
8      7      6      5      4      3      2      1
panyue@Master ~/code/c_demo
```

图 7-3 源程序修改为后进先出链表的测试图

7.2.3 程序设计

(1) 设计一个字段结构 `struct bits`，它将一个 8 位无符号字节从最低位向最高位声明为 8 个字段，各字段依次为 `bit0`, `bit1`, ..., `bit7`，且 `bit0` 的优先级最高。同时设计 8 个函数，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：利用一个短无符号整形 `all` 和一个结构体的联合。

(2) 程序清单：

```
#include <stdio.h>

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;
};

union ISR_REG {
    unsigned short all;
    struct ISR_BITS bit;
};

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

void isr0(void) {
    printf("The Interrupt Service Routine isr0 is called!\n");
}

void isr1(void) {
    printf("The Interrupt Service Routine isr1 is called!\n");
}

void isr2(void) {
    printf("The Interrupt Service Routine isr2 is called!\n");
}

void isr3(void) {

```



```

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

void isr4(void) {
    printf("The Interrupt Service Routine isr4 is called!\n");
}

void isr5(void) {
    printf("The Interrupt Service Routine isr5 is called!\n");
}

void isr6(void) {
    printf("The Interrupt Service Routine isr6 is called!\n");
}

void isr7(void) {
    printf("The Interrupt Service Routine isr7 is called!\n");
}

int main(void) {
    int N, i;
    int num[100];
    union ISR_REG isr_reg;
    scanf("%d", &N);
    for (i = 0; i < N; i++)
        scanf("%d", &num[i]);
    for (i = 0; i < N; i++) {
        printf("%d:\n", num[i]);
        isr_reg.all = num[i];
        if (isr_reg.bit.bit0)

```

```

        isr0();
    if (isr_reg.bit.bit1)
        isr1();
    if (isr_reg.bit.bit2)
        isr2();
    if (isr_reg.bit.bit3)
        isr3();
    if (isr_reg.bit.bit4)
        isr4();
    if (isr_reg.bit.bit5)
        isr5();
    if (isr_reg.bit.bit6)
        isr6();
    if (isr_reg.bit.bit7)
        isr7();
    printf("\n");
}
return 0;
}

```

(3) 测试:

a) 测试数据: 6

18 255 23 42 1 99

b) 测试结果:

```
panyue@Master ~/code/c_demo gcc 1.c -o 001
panyue@Master ~/code/c_demo ./001
6
18 255 23 42 1 99
18:
The Interrupt Service Routine isr1 is called!
The Interrupt Service Routine isr4 is called!

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!

23:
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 isr4 is called!

42:
The Interrupt Service Routine isr1 is called!
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr5 is called!

1:
The Interrupt Service Routine isr0 is called!

99:
The Interrupt Service Routine isr0 is called!
The Interrupt Service Routine isr1 is called!
The Interrupt Service Routine isr5 is called!
The Interrupt Service Routine isr6 is called!
```

图 7-4 编程题一用测试数据得到的测试图

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

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

（3）选做题：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）解题思路：常规线性表的操作即可。

（2）程序清单：（交换数据域）

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/*结构定义：学生*/
typedef struct Student {
    char id[20];
    char name[20];
    float Eng;
    float Math;
    float Phy;
    float C;
    struct Student *next;
    float Sum;
    float Avg;
} stu;

typedef struct {
    stu *head;
    int len;
```

```

} LinkList;

/*函数原型*/

void InitList(LinkList *L, int n);
void PrintInf(LinkList *L);
stu* GetPos(LinkList *L, char *id);
void UpdateList(LinkList *L, char *id, char *class, float grade);
void PrintSumAvg(LinkList *L);
void swap(stu *a, stu *b);
void SortLinkList(LinkList *L);
void PrintAvg(LinkList *L);

int main(void) {
    int n;
    scanf("%d", &n);
    getchar();
    LinkList *L;
    InitList(L, n);
    PrintInf(L);
    int m, i;
    scanf("%d", &m);
    getchar();
    for (i = 0; i < m; i++) {
        char id[100], class[100];
        float grade;
        scanf("%s %s %f", id, class, &grade);
        UpdateList(L, id, class, grade);
    }
    printf("Alter:\n");
    PrintInf(L);
    printf("SumAndAvg:\n");
}

```

```

    PrintSumAvg(L);
    SortLinkList(L);
    printf("Sort:\n");
    PrintAvg(L);
printf("\n");
    return 0;
}

```

```

void InitList(LinkList *L, int n) {
    int i = 0;
    stu *p = (stu *)malloc(sizeof(stu));
    L->head = p;
    L->len = n;
    while(i < n) {
        scanf("%s", p->id);
        scanf("%s", p->name);
        scanf("%f", &p->Eng);
        scanf("%f", &p->Math);
        scanf("%f", &p->Phy);
        scanf("%f", &p->C);
        p->Sum = p->Eng + p->Math + p->Phy + p->C;
        p->Avg = p->Sum / 4;
        if (i < n - 1) {
            stu *q = (stu *)malloc(sizeof(stu));
            p->next = q;
            p = p->next;
        }
        else
            p->next = NULL;
        i++;
    }
}

```

```

    }
}

```

```

void PrintInf(LinkList *L) {
    printf("ID          Name          English    Math\n");
    Physics    C          \n");
    stu *p = L->head;
    while(p) {
        printf("%-15s", p->id);
        printf("%-20s", p->name);
        printf("%-10.2f", p->Eng);
        printf("%-10.2f", p->Math);
        printf("%-10.2f", p->Phy);
        printf("%-10.2f\n", p->C);
        p = p->next;
    }
    printf("\n");
}

```

```

stu* GetPos(LinkList *L, char *id) {
    stu* p = L->head;
    while(strcmp(p->id,id) != 0)
        p = p->next;
    return p;
}

```

```

void UpdateList(LinkList *L, char *id, char *class, float grade) {
    stu *p = GetPos(L, id);
    if (*class == 'E')
        p->Eng = grade;
}

```

```

else if (*class == 'M')
    p->Math = grade;
else if (*class == 'P')
    p->Phy = grade;
else if (*class == 'C')
    p->C = grade;
stu *a = L->head;
while(p) {
    p->Sum = p->Eng + p->Math + p->Phy + p->C;
    p->Avg = p->Sum / 4;
    p = p->next;
}
}

```

```

void PrintSumAvg(LinkList *L) {
    printf("ID          Name          SUM          AVG\n");
    stu *p = L->head;
    while(p) {
        printf("%-15s", p->id);
        printf("%-20s", p->name);
        printf("%-10.2f", p->Sum);
        printf("%-10.2f\n", p->Avg);
        p = p->next;
    }
    printf("\n");
}

```

```

void swap(stu *a, stu *b) {
    char id[20], name[20];

```



```

float Avg, Sum, Eng, Math, Phy, C;

strcpy(id, a->id), strcpy(name, a->name);

Avg = a->Avg, Sum = a->Sum, Eng = a->Eng, Math = a->Math, Phy =
a->Phy, C = a->C;

strcpy(a->id, b->id), strcpy(a->name, b->name);

a->Avg = b->Avg, a->Sum = b->Sum, a->Eng = b->Eng, a->Math =
b->Math, a->Phy = b->Phy, a->C = b->C;

strcpy(b->id, id), strcpy(b->name, name);

b->Avg = Avg, b->Sum = Sum, b->Eng = Eng, b->Math = Math, b->Phy =
Phy, b->C = C;

}

```

```

void SortLinkedList(LinkedList *L) {
    stu *p, *tail, *piror;
    for(p = L->head; p != NULL; p = p->next)
        tail = p;
    while (L->head->next != tail) {
        for (p = L->head; p != tail; p = p->next) {
            if (p->Avg > p->next->Avg) {
                swap(p, p->next);
            }
            piror = p;
        }
        tail = piror;
    }
}

```

```

void PrintAvg(LinkedList *L) {
    printf("ID          Name          AVG          \n");
    stu *p = L->head;

```

```

while(p) {
    printf("%-15s", p->id);
    printf("%-20s", p->name);
    printf("%-10.2f\n", p->Avg);
    p = p->next;
}
printf("\n");
}

```

(3) 测试:

a) 测试数据:

5

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

U20140101 Math 95.6

U20140359 C 100

U20140359 English 100

b) 测试结果:

```

panyue@Master: ~/code/c_demo
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
panyue@Master ~/code/c_demo ./001
5
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
ID      Name      English  Math    Physics  C
U20140101 ZhangChuanChao 85.00   86.00   87.00   88.00
U20140126 MaiDouDou 99.00   99.00   99.00   99.00
U20140158 XiaoDouDou 56.00   85.00   89.00   59.00
U20140312 DaoDaoDog 84.00   89.00   65.00   100.00
U20140359 XiDaDa 88.80   88.80   88.80   88.80
3
U20140101 Math 95.6
U20140359 C 100
U20140359 English 100
Alter:
ID      Name      English  Math    Physics  C
U20140101 ZhangChuanChao 85.00   95.60   87.00   88.00
U20140126 MaiDouDou 99.00   99.00   99.00   99.00
U20140158 XiaoDouDou 56.00   85.00   89.00   59.00
U20140312 DaoDaoDog 84.00   89.00   65.00   100.00
U20140359 XiDaDa 100.00   88.80   88.80   100.00
SumAndAvg:
ID      Name      SUM      AVG
U20140101 ZhangChuanChao 355.60   88.90
U20140126 MaiDouDou 396.00   99.00
U20140158 XiaoDouDou 289.00   72.25
U20140312 DaoDaoDog 338.00   84.50
U20140359 XiDaDa 377.60   94.40
Sort:
ID      Name      AVG
U20140158 XiaoDouDou 72.25
U20140312 DaoDaoDog 84.50
U20140101 ZhangChuanChao 88.90
U20140359 XiDaDa 94.40
U20140126 MaiDouDou 99.00

```

图 7-4 编程题二用测试数据得到的测试图

(4) 选做题：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：修改后的函数如下：

```

void SortLinkList(LinkList *L) {
    stu *p1, *piror1, *p2, *piror2, *t;
    p1 = L->head;
    for (p2 = p1->next, piror2 = p1; p2 != NULL; piror2 = p2, p2 = p2->next) {
        if (p1->Avg > p2->Avg) {
            t = p1->next;
            L->head = p2;
            p1->next = p2->next;

```

```

        piror2->next = p1;
        p2->next = t;
        p1 = p2;
    }
}
for (piror1 = L->head, p1 = piror1->next; p1->next != NULL; piror1 = p1,
p1 = p1->next) {
    for (p2 = p1->next, piror2 = p1; p2 != NULL; piror2 = p2, p2 =
p2->next) {
        if (p1->Avg > p2->Avg) {
            t = p2->next;
            piror1->next = p2;
            piror2->next = p1;
            p2->next = p1->next;
            p1->next = t;
            p1 = p2;
        }
    }
}
}
}

```

（5）选做题：采用双向链表重做编程设计题第（2）题。

解答：（1）程序清单：

```

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

/*结构定义：学生*/
typedef struct Student {
    char id[20];

```

```

    char name[20];

    float Eng;

    float Math;

    float Phy;

    float C;

    struct Student *piror;

    struct Student *next;

    float Sum;

    float Avg;
} stu;

typedef struct {
    stu *head, *tail;

    int len;
} LinkList;

/*函数原型*/

void InitList(LinkList **ReportCard);

void PrintInf(LinkList *L);

void UpdateList(LinkList *L);

void SortLinkList(LinkList *L);

void SumAndAvg(LinkList *L);

void swap(stu *a, stu *b);

void SortList(LinkList *L);

int main(void) {
    LinkList *ReportCard = NULL;

    InitList(&ReportCard);

    PrintInf(ReportCard);

    UpdateList(ReportCard);

```

```

SumAndAvg(ReportCard);

SortList(ReportCard);

return 0;

}

```

```

void InitList(LinkList **ReportCard) {
    int n;
    scanf("%d", &n);
    getchar();
    LinkList *L = (LinkList *)malloc(sizeof(LinkList));
    int i = 1;
    stu *p = (stu *)malloc(sizeof(stu));
    scanf("%s %s %f %f %f %f", p->id, p->name, &p->Eng, &p->Math,
&p->Phy, &p->C);
    p->Sum = p->Eng + p->Math + p->Phy + p->C;
    p->Avg = p->Sum / 4;
    p->piror = NULL, p->next = NULL;

    L->head = p, L->tail = p;
    while (i < n) {
        L->tail->next = (stu *)malloc(sizeof(stu));
        L->tail->next->piror = L->tail;
        scanf("%s %s", L->tail->next->id, L->tail->next->name);
        scanf("%f %f %f %f", &L->tail->next->Eng, &L->tail->next->Math,
&L->tail->next->Phy, &L->tail->next->C);
        L->tail->next->Sum = L->tail->next->Eng + L->tail->next->Math +
L->tail->next->Phy + L->tail->next->C;
        L->tail->next->Avg = L->tail->next->Sum / 4;
        L->tail = L->tail->next;
        L->tail->next = NULL;
    }
}

```

```

        i++;
    }
    *ReportCard = L;
}

void PrintInf(LinkList *L) {
    printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "Name", "English",
"Math", "Physics", "C");
    stu *p = L->head;
    while(p) {
        printf("%-15s", p->id);
        printf("%-20s", p->name);
        printf("%-10.2f", p->Eng);
        printf("%-10.2f", p->Math);
        printf("%-10.2f", p->Phy);
        printf("%-10.2f\n", p->C);
        p = p->next;
    }
    printf("\n");
}

void UpdateList(LinkList *L) {
    int n, i;
    float grade;
    char id[15], course[10];
    scanf("%d", &n);
    getchar();
    for (i = 0; i < n; i++) {
        scanf("%s %s %f", id, course, &grade);
        stu *p = L->head;

```

```

while (p) {
    if (!strcmp(p->id, id))
        break;
    else
        p = p->next;
}
if (!strcmp(course, "English"))
    p->Eng = grade;
else if (!strcmp(course, "Math"))
    p->Math = grade;
else if (!strcmp(course, "Physics"))
    p->Phy = grade;
else if (!strcmp(course, "C"))
    p->C = grade;
}
printf("Alter:\n");
PrintInf(L);
}

```

```

void SumAndAvg(LinkList *L) {
    printf("SumAndAvg:\n");
    printf("%-15s%-20s%-10s%-10s\n", "ID", "Name", "SUM", "AVG");
    stu *p = L->head;
    while(p) {
        printf("%-15s", p->id);
        printf("%-20s", p->name);
        printf("%-10.2f", p->Sum);
        printf("%-10.2f\n", p->Avg);
        p = p->next;
    }
}

```



```

        printf("\n");
    }

void swap(stu *a, stu *b) {
    char id[20], name[20];
    float Avg, Sum, Eng, Math, Phy, C;
    strcpy(id, a->id), strcpy(name, a->name);
    Avg = a->Avg, Sum = a->Sum, Eng = a->Eng, Math = a->Math, Phy =
a->Phy, C = a->C;
    strcpy(a->id, b->id), strcpy(a->name, b->name);
    a->Avg = b->Avg, a->Sum = b->Sum, a->Eng = b->Eng, a->Math =
b->Math, a->Phy = b->Phy, a->C = b->C;
    strcpy(b->id, id), strcpy(b->name, name);
    b->Avg = Avg, b->Sum = Sum, b->Eng = Eng, b->Math = Math, b->Phy =
Phy, b->C = C;
}

void SortList(LinkList *L) {
    stu *p, *tail;
    for(p = L->head; p != NULL; p = p->next)
        tail = p;
    while (L->head->next != tail) {
        for (p = L->head; p != tail; p = p->next) {
            if (p->Avg > p->next->Avg) {
                swap(p, p->next);
            }
        }
        tail = p->next;
    }
    printf("Sort:\n");
}

```

```

printf("%-15s%-20s%-10s\n", "ID", "Name", "AVG");
p = L->head;
while(p) {
    printf("%-15s", p->id);
    printf("%-20s", p->name);
    printf("%-10.2f\n", p->Avg);
    p = p->next;
}
printf("\n");
}

```

(2) 测试:

a) 测试数据:

7

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

U20140455 PengMaMa 89.98 99.56 100 100

U20140415 WuKong 86.5 86 86 86

5

U20140101 Math 95.6

U20140359 C 100

U20140359 English 100

U20140312 C 98.65

U20140415 Physics 90

```

7
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
U20140455 PengMaMa 89.98 99.56 100 100
U20140415 WuKong 86.5 86 86 86
ID          Name          English  Math    Physics  C
U20140101  ZhangChuanChao      85.00   86.00   87.00   88.00
U20140126  MaiDouDou            99.00   99.00   99.00   99.00
U20140158  XiaoDouDou           56.00   85.00   89.00   59.00
U20140312  DaoDaoDog            84.00   89.00   65.00   100.00
U20140359  XiDaDa               88.80   88.80   88.80   88.80
U20140455  PengMaMa             89.98   99.56   100.00   100.00
U20140415  WuKong               86.50   86.00   86.00   86.00

5
U20140101 Math 95.6
U20140359 C 100
U20140359 English 100
U20140312 C 98.65
U20140415 Physics 90
Alter:
ID          Name          English  Math    Physics  C
U20140101  ZhangChuanChao      85.00   95.60   87.00   88.00
U20140126  MaiDouDou            99.00   99.00   99.00   99.00
U20140158  XiaoDouDou           56.00   85.00   89.00   59.00
U20140312  DaoDaoDog            84.00   89.00   65.00   98.65
U20140359  XiDaDa              100.00   88.80   88.80   100.00
U20140455  PengMaMa             89.98   99.56   100.00   100.00
U20140415  WuKong               86.50   86.00   90.00   86.00

SumAndAvg:
ID          Name          SUM      AVG
U20140101  ZhangChuanChao    346.00   86.50
U20140126  MaiDouDou          396.00   99.00
U20140158  XiaoDouDou         289.00   72.25
U20140312  DaoDaoDog          338.00   84.50
U20140359  XiDaDa             355.20   88.80
U20140455  PengMaMa           389.54   97.39
U20140415  WuKong             344.50   86.12

Sort:
ID          Name          AVG
U20140158  XiaoDouDou       72.25
U20140312  DaoDaoDog        84.50
U20140415  WuKong           86.12
U20140101  ZhangChuanChao   86.50
U20140359  XiDaDa           88.80
U20140455  PengMaMa         97.39
U20140126  MaiDouDou        99.00

```

b) 测试结果:

图 7-5 选做题用测试数据得到的测试图

7.3 自设题

(1) 自设实验题目：通过 C 语言完成贪吃蛇游戏设计。

(2) 实验目的：通过对整个游戏的逻辑把握和数据结构的合理使用，进一步掌握 C 语言的各种操作。

(3) 题目分析：

1) 蛇的处理：正好这次实验是结构与联合实验，而对于蛇来说，最好的存储方式就是链表，链表的每一节代表蛇身，蛇增长的操作用链表的增加结点来实现，直接在链表头增加一个结点即可。

2) 食物的处理：使用随机数在地图上生成食物，但是要先遍历蛇身所在的区域，不能把食物生成在蛇身上。

3) 绘制地图：每次蛇的移动后，利用 `system("cls")` 命令清空控制台，再打印下一次的图形，在计算机的极速处理下眼睛视觉不会有很大的差别

(4) 程序清单：

```
#include<stdio.h>

#include<windows.h>

#include<stdlib.h>

#include<time.h>

#include<conio.h>

/*结构定义区*/

struct snk//定义蛇的身体

{

    int x;

    int y;

    struct snk *link;

};

/*全局变量*/

int food[2];
```

```

int life;

int model;

int point;

int level;


/*函数定义区*/

void goto_xy(int x, int y);//定位光标位置到指定坐标
void creatFood(struct snk *L);//创建食物
void drawPoint(int x, int y);//在某处画蛇的身子
void drawFood(int x, int y);//在某处画食物
void drawMap(struct snk *L, int *food);//绘制地图，蛇和食物
int keydown(int z);//获取键盘输入
void move(struct snk *L, int *food, int z);//移动蛇的位置

```

/*函数编写区*/

```

void goto_xy(int x, int y) {
    HANDLE hOut;

    hOut = GetStdHandle(STD_OUTPUT_HANDLE);

    COORD pos = { x,y };

    SetConsoleCursorPosition(hOut, pos);
}


void creatFood(struct snk *L) {
    int flag = 0;

    struct snk *p;

    srand(clock(NULL));

    p = L;

    while (flag != 1) {
        food[0] = rand() % 40 + 1;

        food[1] = rand() % 20 + 1;

```

```

        flag = 1;

        while (p) {

            if (p->x == food[0] && p->y == food[1]) {

                flag = 0;

            }

            p = p->link;

        }

    }

}

void drawPoint(int x, int y) {

    goto_xy(x, y);

    printf("#");

}

void drawFood(int x, int y) {

    goto_xy(x, y);

    printf("@");

    goto_xy(0, 23);

}

void drawMap(struct snk *L, int *food) {

    int i;

    struct snk *p;

    p = L;

    printf("#####\n");

    for (i = 2; i <= 8; i++)

        printf("#

#

#\n");

```

```

printf("#
#####\n");
for (i = 10; i <= 21; i++)
    printf("#
# \n");
printf("#####
#####\n");
goto_xy(45, 3);
printf("您的分数: ");
goto_xy(50, 5);
printf("%d", point);
goto_xy(45, 11);
printf("当前等级: ");
goto_xy(48, 13);
if (model == 1) {
    printf("普通模式");
    goto_xy(45, 15);
    printf("移动速度: ");
    goto_xy(48, 17);
    printf("正常");
}
else {
    printf("第 %d 级", level);
    goto_xy(45, 15);
    printf("移动速度: ");
    goto_xy(48, 17);
    switch (level) {
    case 1:
        printf("慢速");
        break;

```

```

        case 2:
            printf("正常");
            break;
        case 3:
            printf("快速");
            break;
        case 4:
            printf("飞快");
            break;
        case 5:
            printf("疯狂");
            break;
    }
}

while (p) {
    drawPoint(p->x, p->y);
    p = p->link;
}

drawFood(food[0], food[1]);
}

```

```

int keydown(int z) {
    char ch;
    if (_kbhit()) {
        ch = getch();
        switch (ch) {
            case 'w':
                if (z != 2)
                    z = 1;
                break;

```



```

        case 's':
            if (z != 1)
                z = 2;
            break;
        case 'a':
            if (z != 4)
                z = 3;
            break;
        case 'd':
            if (z != 3)
                z = 4;
            break;
        default:
            break;
    }
}

return z;
}

```

```

void move(struct snk *L, int *food, int z) {
    struct snk *p, *q;
    p = (struct snk *)malloc(sizeof(struct snk));
    p->x = L->x, p->y = L->y;
    p->link = L->link;
    switch (z) {
        case 1:
            L->y -= 1;
            break;
        case 2:
            L->y += 1;

```

```

        break;
case 3:
    L->x -= 1;
    break;
case 4:
    L->x += 1;
    break;
default:
    break;
}
L->link = p;
if (L->x == food[0] && L->y == food[1]) {
    point += 100;
    int flag = 0;
    struct snk *r;
    srand(clock(NULL));
    r = L;
    while (flag != 1) {
        food[0] = rand() % 40 + 1;
        food[1] = rand() % 20 + 1;
        flag = 1;
        while (r) {
            if (r->x == food[0] && r->y == food[1]) {
                flag = 0;
            }
            r = r->link;
        }
    }
}
else {

```

```

        q = L;
        while (q->link->link)
            q = q->link;
        free(q->link);
        q->link = NULL;
    }
}

/*主函数*/
int main(void) {

    system("mode con cols=62 lines=32");//设置窗体大小

                                /*游戏开始界面*/

    int j;
    printf("#####\n");
    printf("#####\n");
    for (j = 2; j <= 26; j++)
        printf("##
        ##\n");
    printf("##    tip: 请将输入法切换成英文进行游戏。By: 潘越
    ##\n");
    printf("#####\n");
    printf("#####\n");

```

```

/*打印标题等等*/
goto_xy(0, 4);
printf("##          #                      #                      #
##\n");
printf("##          #  #                      #                      #
##\n");
printf("##          #  ###  #                      # #####          #  #####
##\n");
printf("##          #          ##### #                      #####  #  #
##\n");
printf("##          #####          #  #          #####          #  #  #  #
##\n");
printf("##          #  #  #          #  #          #          #  #  #          #  #
##\n");
printf("##          #  #  #          #  #          #          #  #  #          #  #
##\n");
printf("##          #  #  #          #####          #          #####          ##
##\n");
printf("##          #  #                      #                      #          #
##\n");
printf("##          #  #                      #          #          #  #          #
#  ##\n");
printf("##          #          #                      #####          #####          #####
##\n");

goto_xy(20, 17);
printf("方向控制： "), goto_xy(20, 19);
printf(" w 上 s 下 a 左 d 右"), goto_xy(20, 21);
printf("按 1 或 2 选择模式： "), goto_xy(20, 23);
printf("      1 → 一般模式"), goto_xy(20, 24);

```

```

printf("    2 → 挑战模式"), goto_xy(20, 25);
printf("您的选择:  ");
scanf("%d", &model);
char t = 'y';
/*游戏大循环*/
while (t == 'y' || t == 'Y') {
    /*游戏初始化*/

    life = 1;//生命值
    point = 0;//分数
    level = 1;//等级
    int z = 4;//开始蛇向右移动
    struct snk *snake;//定义蛇头
    struct snk *body;//遍历蛇的身体用
    snake = (struct snk *)malloc(sizeof(struct snk));
    snake->x = 21, snake->y = 11;
    snake->link = (struct snk *)malloc(sizeof(struct snk));
    snake->link->x = 20, snake->link->y = 11;
    snake->link->link = (struct snk *)malloc(sizeof(struct snk));
    snake->link->link->x = 19, snake->link->link->y = 11;
    snake->link->link->link = (struct snk *)malloc(sizeof(struct snk));
    snake->link->link->link->x = 18, snake->link->link->link->y = 11;
    snake->link->link->link->link = NULL;//长度为 4 的初始蛇身
    creatFood(snake);

    /*游戏主循环*/

    while (1) {
        system("cls");

```

22)

```
drawMap(snake, food); //绘制图形
z = keydown(z); //获取输入
move(snake, food, z); //移动蛇身
if (snake->x == 0 || snake->x == 42 || snake->y == 0 || snake->y ==

    life = 0; //判定蛇是否撞墙
body = snake->link;
while (body) {
    if (body->x == snake->x && body->y == snake->y)
        life = 0;
    body = body->link;
} //判定蛇是否撞到自己
if (!life)
    break;
if (point >= 600 && point < 1200)
    level = 2;
else if (point >= 1200 && point < 2000)
    level = 3;
else if (point >= 2000 && point < 2800)
    level = 4;
else if (point >= 2800)
    level = 5;
if (model == 1)
    level = 2;
switch (level) {
case 1:
    Sleep(400);
    break;
case 2:
    Sleep(300);
```

```

        break;
    case 3:
        Sleep(200);
        break;
    case 4:
        Sleep(100);
        break;
    case 5:
        Sleep(50);
        break;
    }
}

/*游戏主循环*/

goto_xy(0, 23);
printf("你输了！ 你想再挑战一次吗？ -----Y/N  ");
char ch;
while ((ch = getchar()) != '\n' && ch != EOF);
scanf("%c", &t);//读取下一次的决定
}
return 0;
}

```

(5) 测试:



图 7-6 自测题的测试图



图 7-7 自测题的测试图

实验 8 文件实验

8.1 实验目的

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

8.2 实验内容

8.2.1 文件类型的程序验证

设有程序：

```
#include <stdio.h>

int main(void) {
    short a = 0x253f, b = 0x7b7d;
    char ch;
    FILE *fp1, *fp2;
    fp1 = fopen("E:\\abc1.bin", "wb+");
    fp2 = fopen("E:\\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。
- (2) 将两处 `sizeof(short)` 均改为 `sizeof(char)` 后，结果有什么不同？为什么？
- (3) 将 `fprintf(fp2, "%hx %hx", a, b)` 改为 `fprintf(fp2, "%d %d", a, b)` 后结果有什么不同？

解答：(1) 输出结果应为：

```

?%}{
253f 7b7d

```



图 8-1 源程序验证的测试图

- (2) 输出的结果应该为?}

因为 `char` 类型只占一个字节，因此在 `fwrite` 的时候只获取了低字节。



图 8-2 源程序验证的测试图

- (3) 此时输出的是十进制的数字



图 8-3 源程序验证的测试图

8.2.2 源程序修改替换

已知将指定的文本文件内容在屏幕上显示出来的命令行格式为

type filename

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

源程序：

```
#include <stdio.h>

#include <stdlib.h>

int main(int argc, char const *argv[]) {
    char ch;
    FILE *fp;
    if (argc != 2) {
        printf("Arguments error!\n");
        exit(-1);
    }
    if ((fp = fopen(argv[1], "r")) == NULL) {
        printf("Can't open %s file!\n", argv[1]);
        exit(-1);
    }
    while (ch = fgetc(fp) != EOF)
        putchar(ch);
    fclose(fp);
    return 0;
}
```

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

解答：（1）源程序的功能为：读取文件中的内容并输出。

源程序的逻辑错误为：最后一个 **ch = fgetc (fp) != EOF** 应该有一个括号。

修改后的程序如下：

```
#include <stdio.h>

#include <stdlib.h>

int main(int argc, char const *argv[]) {
    char ch;
    FILE *fp;
    if (argc != 2) {
        printf("Arguments error!\n");
        exit(-1);
    }
    if ((fp = fopen(argv[1], "r")) == NULL) {
        printf("Can't open %s file!\n", argv[1]);
        exit(-1);
    }
    while ((ch = fgetc(fp)) != EOF)
        putchar(ch);
    fclose(fp);
    return 0;
}
```

测试：

A screenshot of a Windows command prompt window. The title bar shows 'C:\windows\system32\cmd.exe'. The window content displays the Microsoft Windows logo, version 10.0.14393, and copyright information for 2016 Microsoft Corporation. The user has entered the command 'E:\project1.exe E:\abc2.txt' and the program has outputted '9535 31613'. The prompt is now 'C:\Users\samsung>'.

```
C:\windows\system32\cmd.exe
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

C:\Users\samsung>E:\project1.exe E:\abc2.txt
9535 31613
C:\Users\samsung>
```

图 8-4 源程序修改后的测试图

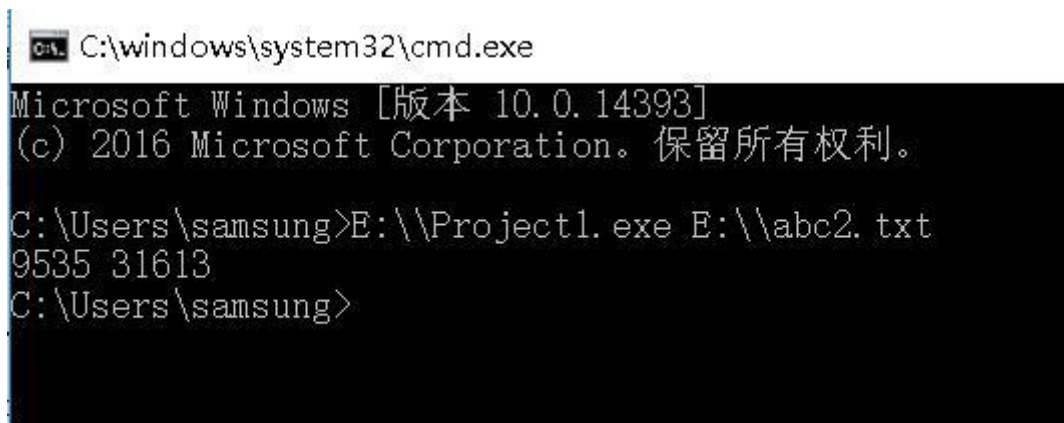
(2) 用重定向改写的函数如下：

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

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

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

测试:



```
C:\windows\system32\cmd.exe  
Microsoft Windows [版本 10.0.14393]  
(c) 2016 Microsoft Corporation。保留所有权利。  
C:\Users\samsung>E:\Project1.exe E:\abc2.txt  
9535 31613  
C:\Users\samsung>
```

图 8-5 源程序使用重定向输出的测试图

8.2.3 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，实现以下功能：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：

- 1.定义两个状态 0 和 1。
- 2.若状态为 1，则将小写改为大写（首字母）
- 3.若状态为 0，检测该字符是否为空格且下一个字符是否是字母
若是则将状态改为 1。

（2）程序清单：

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

int main(void) {
    char ch[100];
    fgets(ch, 100, stdin);
    int i = 0, flag = 1;
    while (i < strlen(ch)) {
        if (flag == 1) {
            if (ch[i] > 'a' && ch[i] < 'z')
                ch[i] = ch[i] - ('a' - 'A');
            flag = 0;
        }
        if (ch[i] == ' ' && ch[i+1] != ' ')
            flag = 1;
        i++;
    }
    FILE *fp;
    fp = fopen("E:\\test.txt", "a");
```

```
fwrite(ch, sizeof(char) * strlen(ch), 1, fp);  
fclose(fp);  
return 0;  
}
```

(3) 测试:

a) 测试数据:

he likes math and C programming language.

b) 测试结果:

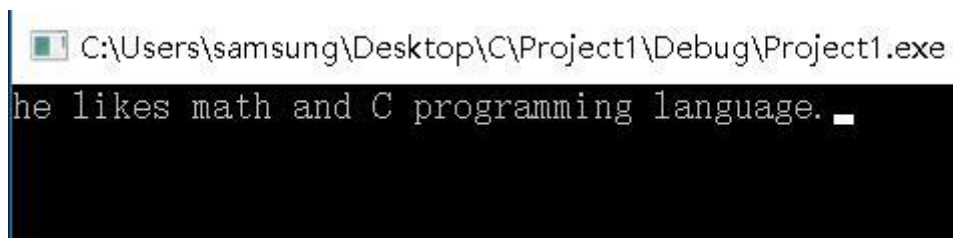


图 8-6 程序输入的测试图



图 8-7 程序运行的结果图（写入的文件）

8.3 实验小结

文件操作是 C 语言中非常重要的一环，在暑假的课设中也要使用。不过本次实验涉及到的东西并不是太多，还是要再啃一啃书本，多写写代码，熟悉其他的文件操作。