

**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： 2017.3.23**

**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 4](#_Toc481647040)

[1.1实验目的 4](#_Toc481647041)

[1.2实验内容 4](#_Toc481647042)

[1.2.1源程序改错 4](#_Toc481647043)

[1.2.2 源程序修改替换 6](#_Toc481647044)

[1.2.3 程序设计 7](#_Toc481647045)

[1.3自设题 11](#_Toc481647046)

[1.4实验小结 13](#_Toc481647047)

[实验2 流程控制实验 13](#_Toc481647048)

[2.1、实验目的 13](#_Toc481647049)

[2.2、实验内容 14](#_Toc481647050)

[2.2.1源程序改错 14](#_Toc481647051)

[2.2.2源程序修改替换 14](#_Toc481647052)

[2.2.3程序设计 16](#_Toc481647053)

[2.3自设题 26](#_Toc481647054)

[2.4实验小结 28](#_Toc481647055)

[实验3 函数与程序结构实验 28](#_Toc481647056)

[3.1实验目的 28](#_Toc481647057)

[3.2实验内容 29](#_Toc481647058)

[3.2.1源程序改错 29](#_Toc481647059)

[3.2.2源程序修改替换 30](#_Toc481647060)

[3.2.3跟踪调试题 32](#_Toc481647061)

[3.2.4程序设计 34](#_Toc481647062)

[3.3自设题 40](#_Toc481647063)

[3.4实验小结 44](#_Toc481647064)

[实验4 编译预处理实验 44](#_Toc481647065)

[4.1实验目的 44](#_Toc481647066)

[4.2实验内容 45](#_Toc481647067)

[4.2.1源程序改错 45](#_Toc481647068)

[4.2.2源程序修改替换 46](#_Toc481647069)

[4.2.3跟踪调试题 50](#_Toc481647070)

[4.2.4程序设计 52](#_Toc481647071)

[4.3实验小结 56](#_Toc481647072)

[实验5 数组实验 56](#_Toc481647073)

[5.1实验目的 56](#_Toc481647074)

[5.2实验内容 56](#_Toc481647075)

[5.2.1源程序改错 56](#_Toc481647076)

[5.2.2源程序修改替换 58](#_Toc481647077)

[5.2.3跟踪调试题 62](#_Toc481647078)

[5.2.4程序设计 65](#_Toc481647079)

[5.3实验小结 76](#_Toc481647080)

# 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；

1. 掌握getchar, putchar, scanf 和printf 函数的用法。
2. 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：
2. 第2行的符号常量定义后不能有分好，正确形式为：

define PI 3.14159

1. 第3行的voidmain中间应该有空格，正确形式为：

void main (void)

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

printf("Input Fahrenheit: " ) ;

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

scanf("%d", f ) ;

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

scanf("%d",& f ) ;

1. 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) ;

1. 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 ) ;

1. 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);

1. 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);

1. 第21行的newint为定义，正确形式为：

在开始加上 short newint

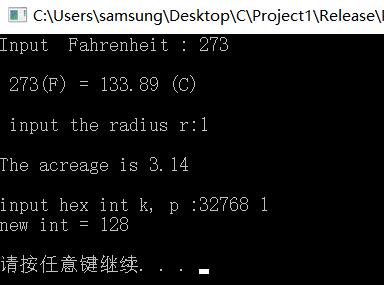
1. 第21行newint = (p&0xff00)|(k&0xff00)<<8;中后面应该是右移8位，正确形式为：

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

1. 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);

1. 错误修改后运行结果：

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：**

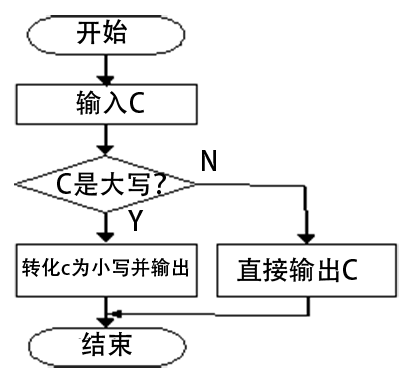
1. ****算法流程如图1.1所示

图1-3 编程题1的程序流程图

1. 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;

}

1. 测试
2. 测试数据：

Z s g A

1. 对应测试数据的运行结果截图

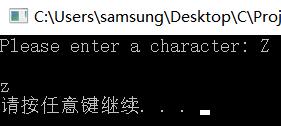


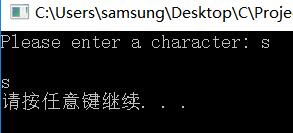
图1-4 编程题1用‘Z’进行测试的测试图

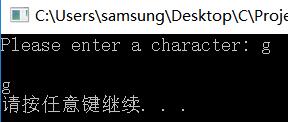
图1-5 编程题1用‘s’进行测试的测试图

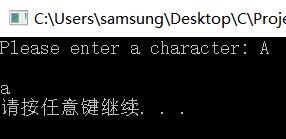
图1-6 编程题1用‘g’进行测试的测试图

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 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ）,取出x从第ｍ位开始向左的ｎ位（ｍ从右至左编号为0～15），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 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，转2.1，否则转3.

2.1 首先x>>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>>m) <<(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



图1-8 编程题2的测试用例一的运行结果

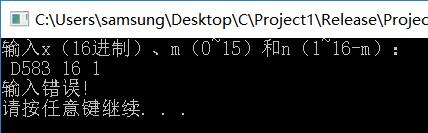


图1-9 编程题2的测试用例二的运行结果

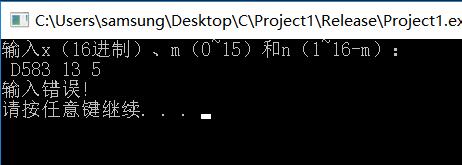


图1-10 编程题2的测试用例三的运行结果

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

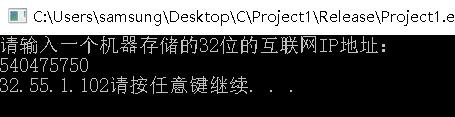


图1-11 编程题3的测试用例一的运行结果

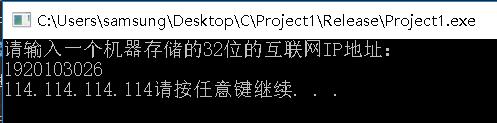


图1-12 编程题3的测试用例二的运行结果

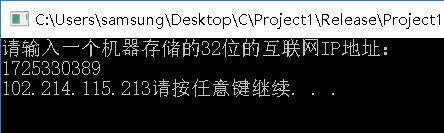


图1-13 编程题3的测试用例三的运行结果

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ecdba，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
2. 第六行scanf("%d",n);用错，正确形式为：

scanf("%d", &n);

1. 第七行for(i=1,i<=n,i++)用错，正确形式为：

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

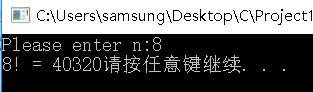
1. 错误修改后运行结果：

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！≥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
2. 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;

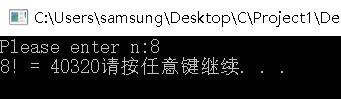
}

图2-2 程序替换之后的测试图

1. 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;

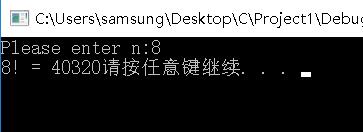
}

图2-3 程序替换之后的测试图

1. 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≤x＜2000，收取%5的税金；2000≤x＜3000，收取%10的税金；3000≤x＜4000，收取15%的税金；4000≤x＜5000，收取20%的税金；x≥5000，收取25%的税金。编写一个程序，输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

1. 解题思路：根据对应税金区域列式计算即可。
2. 程序清单：
3. 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;

}

1. 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;

}

1. 测试
2. 测试数据

500

1000

1500

2000

2500

3000

3500

4000

4500

5000

5500

0

1. 测试结果

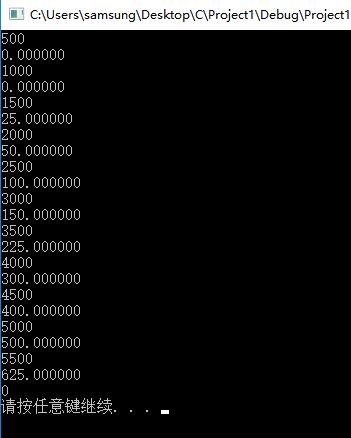


图2-5 编程题1用if语句得到的测试图

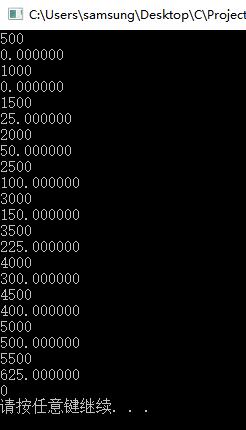


图2-6编程题1用switch语句得到的测试图

（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
2. 定义两种状态，COPY和SPACE；
3. 开始时将状态设置为COPY
4. 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；

3.1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。

1. 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;

}

}

printf("\n");

}

return 0;

}

1. 测试
2. 测试数据

3

Hello world !

Super Mario World

a b c d e

1. 测试结果

图2-7 编程题2测试数据得到的测试图

（3）打印如下杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

每个数据值可以由组合计算（表示第i行第j列位置的值），而的计算如下：

 (i=0,1,2,…)

 (j=0,1,2,3,…,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");

}

1. 测试：
2. 测试数据：

输入：

2

10

11

0

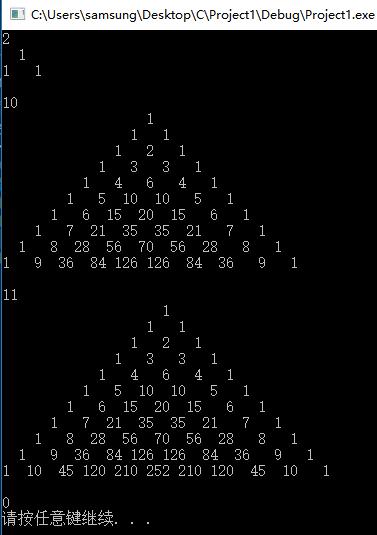
1. 测试结果：

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。

1. 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

1. 测试结果：

图2-9 编程题3用测试数据得到的测试图

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

1. 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;

}

1. 测试
2. 测试数据

1

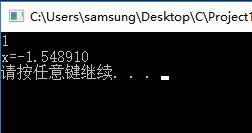
1. 测试结果

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

ABCDCBA

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，应该十一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

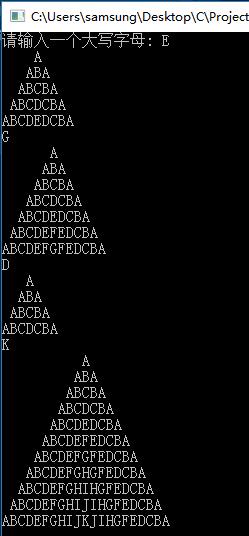
E

G

D

K

b）测试结果

 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!+…+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++);

1. 第十二行fac没有赋初始值，正确形式为：

long fac = 1;

1. 第十四行的for循环缺少大括号，正确形式为：

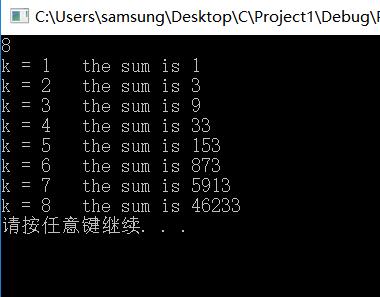
for (i = 1; i <= n; i++) {

fac \*= i;

s += fac;

}

1. 错误修改后运行结果：

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。

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;

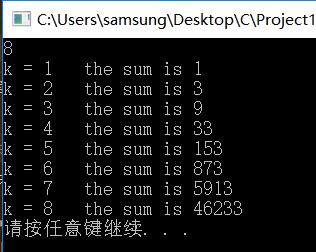
}

图3-2 程序替换之后的测试图

1. **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;

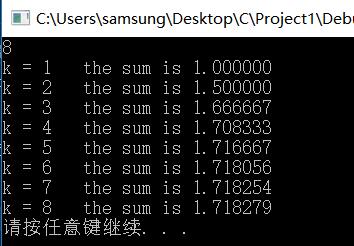
}

图3-3 程序替换之后的测试图

### 3.2.3跟踪调试题

计算fabonacci数列前n项和的程序如下：

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

void main(void)

{

int i,k;

long sum=0,\*p=&sum;

scanf("%d",&k);

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

sum+=fabonacci(i);

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

}

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

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fabonacci函数返回后光条停留在哪个语句上？

（3）进入fabonacci函数，watch窗口显示的是什么？

（4）当i=3，从调用fabonacci函数到返回，n值如何变化？

**解答**：

（1）

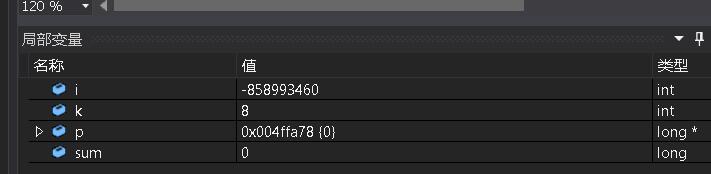


图3-4 （1）问对应的截图

（2）

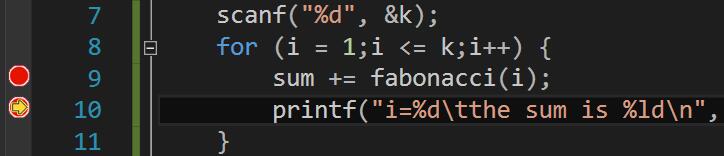


图3-5 （2）问对应的截图

（3）

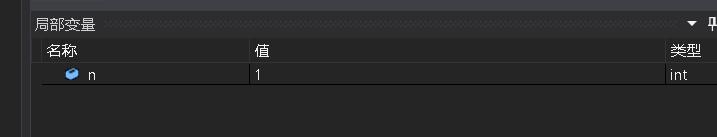


图3-6 （3）问对应的截图

（4）

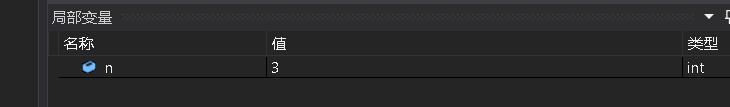


图3-7 （4）问对应的截图1

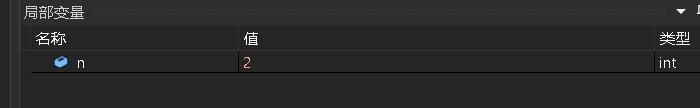


图3-8 （4）问对应的截图2

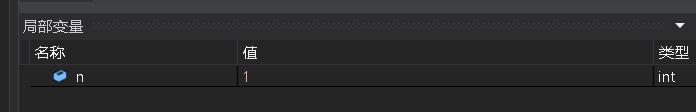


图3-9 （4）问对应的截图3

### 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. 若min能整除max，则最大公约数为min
4. 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 |

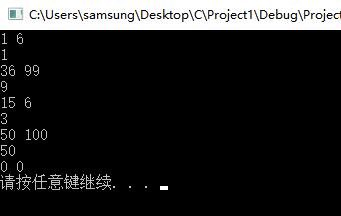
 b）测试结果：

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>=4 is the sum of two primes.

10=3+7

12=5+7

……

20=3+17

**解答：**

1. 解题思路
2. 用一个函数输出素数表，并用一个数组储存；
3. 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。
4. 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");

}

1. 测试
   1. 测试数据

6 10

10 20

7 15

8 8

0 0

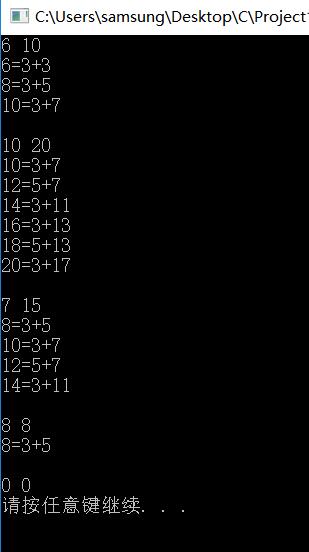
* 1. 测试结果

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：
2. file.h

#include <stdio.h>

extern int x, y; /\* 外部变量的引用性说明 \*/

extern char ch; /\*

1. 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();

}

1. 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);

}

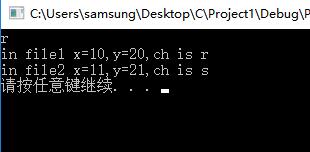
1. 测试：

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);

}

}

}

}

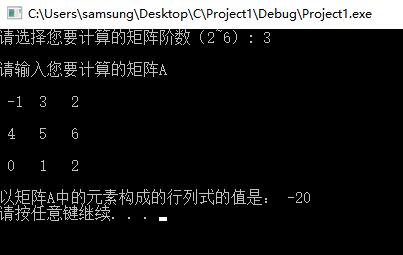
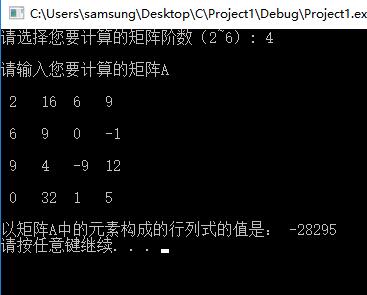
1. 实验测试：

图3-自设题用测试数据得到的测试图

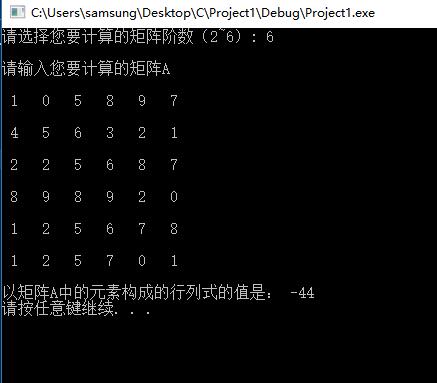
图3-自设题用测试数据得到的测试图

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；

1. 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);

1. 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)

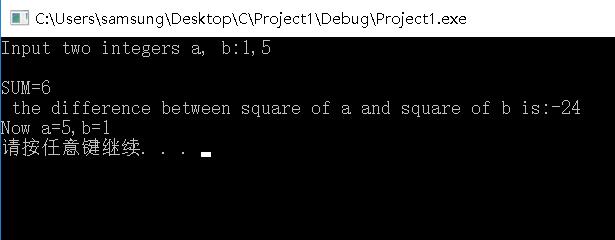
 （2）错误修改后运行结果

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;

}

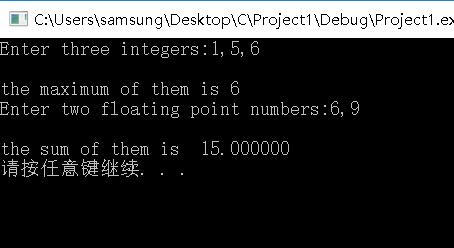
 2）测试：

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;

}

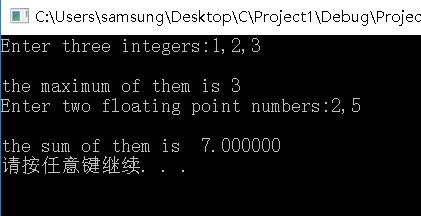
2）测试结果：

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;

}

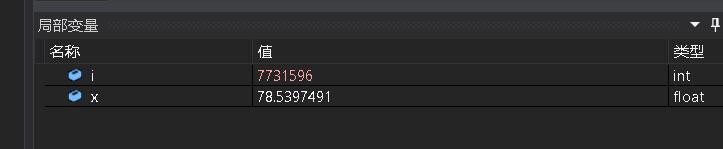
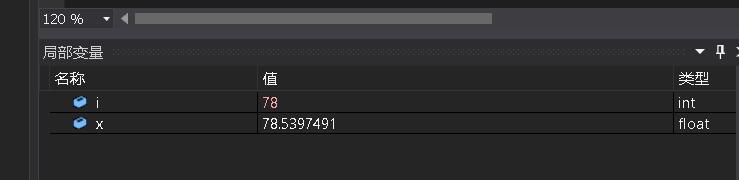
2）

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，其中，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

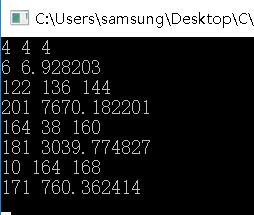
 b）测试结果：

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?/\Ziggbin

b）测试结果：

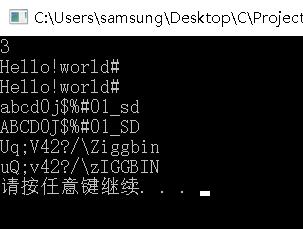


图4-7 编程题2用测试数据得到的测试图

## 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);

1. 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：



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，直到圈中只剩下一个人为止。请在源程序中的下划线处填写合适的代码来完善该程序。

源程序：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

b[M-i] = j? : ; /\* 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;

}

1. 测试：

图5-2 源程序修改替换后得到的测试图

1. 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++)

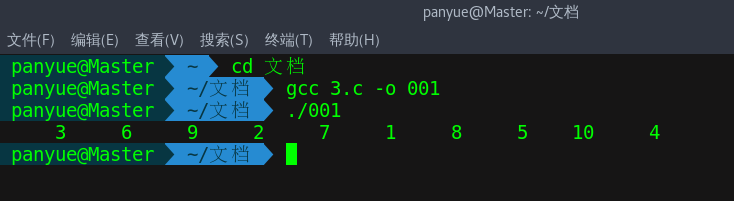
if(!c[i])

printf("%6d\n", a[i]);

return 0;

}

1. 测试:

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++]) ;

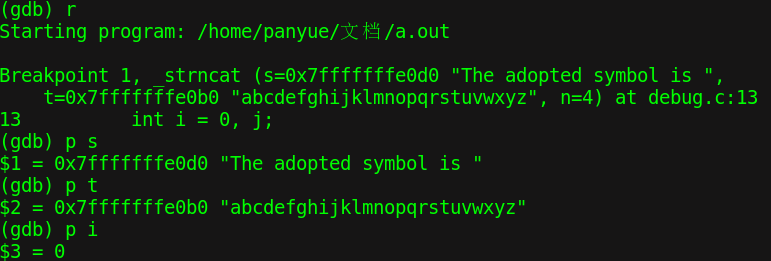
for(j = 0; j < n && t[j];)

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

s[i] = '\0';

}

**解答：**（1）各种情形下的调试图如下

图5-4 调试中当光标达到\_strncat时的调试图

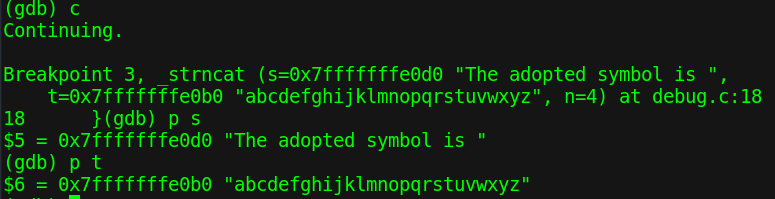
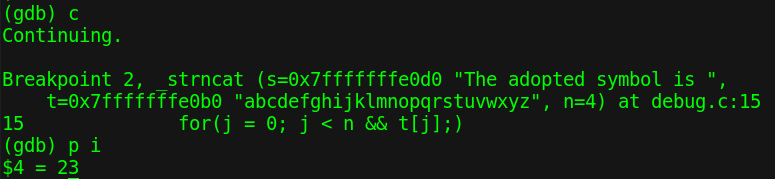
图5-5 调试中当光标达到for语句时的调试图

图5-6 调试中当光标达到函数结尾大括号时的调试图

1. 函数出错的原因：没有考虑消去原字符串s中的’\0’字符
2. 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++;

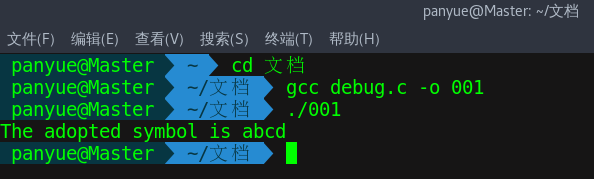
for(j = 0; j < n && t[j];)

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

s[i] = '\0';

}

1. 测试：

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：

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mask = 1 << 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‘2’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

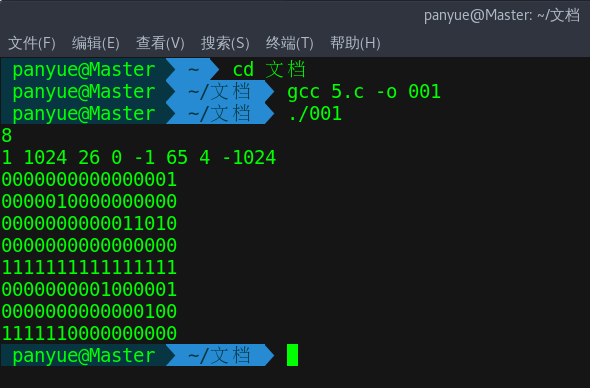
 b）测试结果：

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

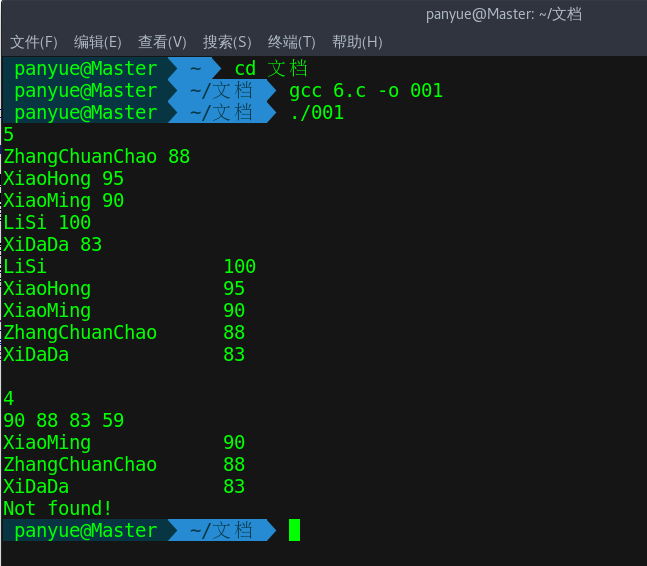
b）测试结果：

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"

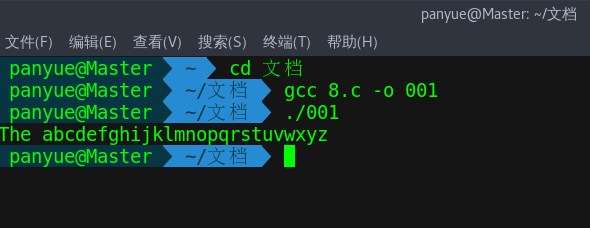
 b）测试结果：

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

* 1. 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
  2. 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。
  3. 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;

}

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);

}

}

}

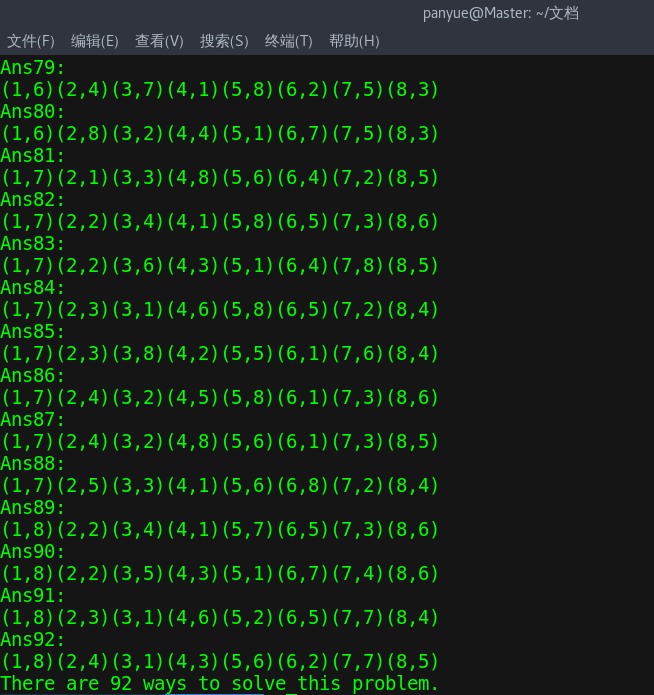
（3）测试：

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。