

**课 程 实 验 报 告**

**课程名称： C语言程序设计**

**专业班级： 信息安全1602班**

**学 号： U201614848**

**姓 名： 郭倜维**

**指导教师： 周时阳**

**报告日期： 2017年6月11日**

**计算机科学与技术学院**

目录

[1表达式和标准输入输出实验 6](#_Toc484626484)

[1.1实验目的 6](#_Toc484626485)

[1.2 必做题 6](#_Toc484626486)

[1.2.1 源程序改错题 6](#_Toc484626487)

[1.2.2 源程序修改替换 10](#_Toc484626488)

[1.2.3 字母大小写转换 13](#_Toc484626489)

[1.2.4 取出移位 15](#_Toc484626490)

[1.2.5 IP地址解析 17](#_Toc484626491)

[1.3 自设题 19](#_Toc484626492)

[1.4 小结 20](#_Toc484626493)

[2流程控制实验 21](#_Toc484626494)

[2.1实验目的 21](#_Toc484626495)

[2.2 必做题 21](#_Toc484626496)

[2.2.1 源程序改错 21](#_Toc484626497)

[2.2.2 源程序修改替换 24](#_Toc484626498)

[2.2.3 工资税收问题 31](#_Toc484626499)

[2.2.4 空字符缩减 37](#_Toc484626500)

[2.2.5 杨辉三角 40](#_Toc484626501)

[2.2.6 整数逆转 43](#_Toc484626502)

[2.3 选做题 46](#_Toc484626503)

[2.3.1 牛顿迭代 46](#_Toc484626504)

[2.4 自设题 48](#_Toc484626505)

[2.5 小结 49](#_Toc484626506)

[3函数与程序结构实验 50](#_Toc484626507)

[3.1 实验目的 50](#_Toc484626508)

[3.2 必做题 50](#_Toc484626509)

[3.2.1 源程序改错 50](#_Toc484626510)

[3.2.2 源程序修改替换 53](#_Toc484626511)

[3.2.3 跟踪调试 57](#_Toc484626512)

[3.2.4 最大公约数 62](#_Toc484626513)

[3.2.5 验证哥德巴赫猜想 66](#_Toc484626514)

[3.2.6 验证部分哥德巴赫猜想 70](#_Toc484626515)

[3.3 选做题 75](#_Toc484626516)

[3.3.1 建立多源文件程序 75](#_Toc484626517)

[3.4 自设题 78](#_Toc484626518)

[3.5 小结 78](#_Toc484626519)

[4编译预处理实验 78](#_Toc484626520)

[4.1实验目的 78](#_Toc484626521)

[4.2 必做题 78](#_Toc484626522)

[4.2.1 源程序改错 78](#_Toc484626523)

[4.2.2 源程序修改替换 82](#_Toc484626524)

[4.2.3 跟踪调试 86](#_Toc484626525)

[4.2.4 计算三角形面积 90](#_Toc484626526)

[4.2.5 条件编译电报 93](#_Toc484626527)

[4.3小结 96](#_Toc484626528)

[5数组实验 96](#_Toc484626529)

[5.1实验目的 96](#_Toc484626530)

[5.2 必做题 96](#_Toc484626531)

[5.2.1 源程序改错 96](#_Toc484626532)

[5.2.2 源程序完善、修改、替换 100](#_Toc484626533)

[5.2.3 跟踪调试 102](#_Toc484626534)

[5.2.4 矩阵的运算 105](#_Toc484626535)

[5.2.5 整数二进制输出 109](#_Toc484626536)

[5.2.6 C语言成绩排序查询 112](#_Toc484626537)

[5.3 选做题 117](#_Toc484626538)

[5.4自设题 120](#_Toc484626539)

[5.5小结 120](#_Toc484626540)

[6指针实验 120](#_Toc484626541)

[6.1 实验目的 120](#_Toc484626542)

[6.2 必做题 121](#_Toc484626543)

[6.2.1 源程序改错 121](#_Toc484626544)

[6.2.2 源程序修改替换 122](#_Toc484626545)

[6.2.3 跟踪调试 128](#_Toc484626546)

[6.2.4 十六进制输出 131](#_Toc484626547)

[6.2.5 去除空格空行 135](#_Toc484626548)

[6.2.6 命令行参数 139](#_Toc484626549)

[6.2.7 成绩处理 141](#_Toc484626550)

[6.3 选做题 152](#_Toc484626551)

[6.3.1 小数相加 152](#_Toc484626552)

[6.3.2 复杂声明 157](#_Toc484626553)

[6.4 自设题 161](#_Toc484626554)

[6.5 小结 162](#_Toc484626555)

[7结构与联合实验 162](#_Toc484626556)

[7.1 实验目的 162](#_Toc484626557)

[7.2 必做题 162](#_Toc484626558)

[7.2.1 表达式求值 162](#_Toc484626559)

[7.2.2 源程序修改替换 166](#_Toc484626560)

[7.2.3 字段函数 172](#_Toc484626561)

[7.2.4 成绩统计 180](#_Toc484626562)

[7.3 选做题 192](#_Toc484626563)

[7.3.1 双向链表 192](#_Toc484626564)

[7.4 小结 200](#_Toc484626565)

[8文件实验 201](#_Toc484626566)

[8.1实验目的 201](#_Toc484626567)

[8.2 必做题 201](#_Toc484626568)

[8.2.1 文件程序验证 201](#_Toc484626569)

[8.2.2源程序修改替换 205](#_Toc484626570)

[8.2.3 大小写转化 209](#_Toc484626571)

[8.3小结 212](#_Toc484626572)

[参考文献 212](#_Toc484626573)

# 1表达式和标准输入输出实验

## 1.1实验目的

通过完成程序实验，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；尤其是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；同时熟记运算符的优先级和结合性。其次，通过编写代码，熟练掌握常见输出输入函数getchar, putchar, scanf 和printf 的用法，从而掌握简单c程序的编写方法。

## 1.2 必做题

### 1.2.1 源程序改错题

**【题目】**

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2.1.3和2.1.4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);

}

**【错误原因分析及改错方案】**

经过修改，发现该源程序的错误主要分成以下几类：字符使用错误、逻辑错误、未定义变量、数据精度不够、宏展开错误；具体情况如下。

第2行错误，#define之后出现分号，使后面的宏展开错误；正确形式应为#define PI 3.14159。

第3行错误，voidmain之间未打空格，而且void使用不恰当；正确形式应为int main；

第10、12行错误，引号全部使用全角引号；正确形式应为scanf("%d", &f); printf(" \n %d(F) = %.2f (C)\n\n ", f, c);

第10行错误，（double）r的类型是double，所以输入的%f的格式错误；正确形式应为scanf("%d", &lf);

第11行错误，数据5/9的精度不够，；正确形式应为c = 5.0/ 9.0 \* (f - 32);

第17行错误，printf数据后使用了寻址运算符，使得所取数据错误；正确形式应为printf("\nThe acreage is %.2f\n\n", s)；

第20行错误，输入的格式是“%x”，但要求输入的是短整型；正确形式应为scanf("%hd %hd", &k, &p);；

第21行错误，未定义变量newint，右移八位逻辑错误；正确形式应为newint = (p&0xff00) | (k&0xff00) >> 8; 同时在函数前加声明语句int newint

第22行错误，输出格式为%x十六进制，按照要求应输出短整型数据；正确形式应为printf("new int = %hd\n\n",newint);

将源程序错误修改后使用编译器测试。

**【修改后代码】**

#include<stdio.h>

#define PI 3.14159

int main(void)

{

int f;

int newint;

short p, k;

double c, r, s;

/\* for task 1 \*/

printf("Input Fahrenheit : ");

scanf("%d", &f);

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

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

/\* for task 2 \*/

printf("input the radius r:");

scanf("%lf", &r);

s = PI \* r \* r;

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

/\* for task 3 \*/

printf("input hex int k, p :");

scanf("%hd %hd", &k, &p);

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

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

}

**【测试数据、测试理论结果及测试结果】**

表1-1 实验一必做题1.2.1测试数据、测试理论结果及测试结果

|  |  |  |
| --- | --- | --- |
| 测试数据 | 测试理论结果 | 测试结果 |
| 5 | -15.00 | 如图1-1 |
| 12 | 452.39 |
| 5 8 | 0 |

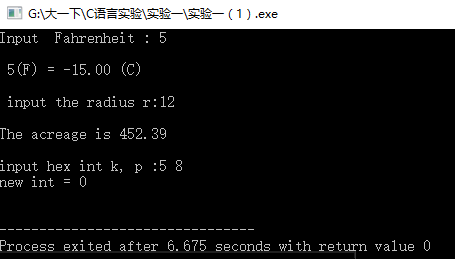


图1-1 实验1必做题1.2.1测试结果

### 1.2.2 源程序修改替换

**【题目】**

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；

prinf(“\na=%d,b=%d”,a,b);

}

**【解题思路】**

要求不使用中间变量t，只要先将a+b的值赋给a，然后将a-b的值赋给b，则此时b的值为初始a的值，最后将a-b的值赋给a，则a和b的值就交换了。

**【代码改写】**

#include<stdio.h>

int main( )

{

int a, b;

printf("Input two integers:");

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

a=a+b;

b=a-b;

a=a-b;

printf("\na=%d,b=%d",a,b);

}

**【测试数据、测试理论结果及测试结果】**

表1-2 实验一必做题1.2.2测试数据、测试理论结果及测试结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 样例 | 测试数据 | | 测试理论结果 | | 备注 | 测试结果 |
| a | b | a | b |
| （1） | 8 | 15 | 15 | 18 | a>b | 图1-2 |
| （2） | 24 | 4 | 4 | 124 | a>>b | 图1-3 |
| （3） | 7 | 879 | 879 | 67 | a<b | 图1-4 |

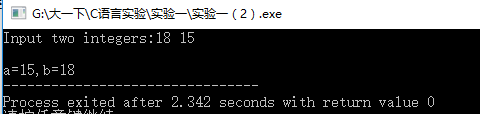


图1-2实验一必做题1.2.2样例（1）测试结果

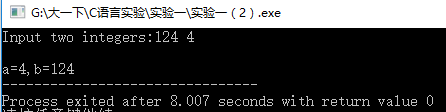


图1-3实验一必做题1.2.2样例（2）测试结果

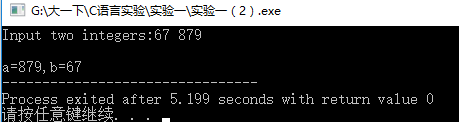


图1-4实验一必做题1.2.2样例（3）测试结果

### 1.2.3 字母大小写转换

**【题目】**

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。

**【解题思路】**

根据题目要求，我将字母转换成ASCII码来设计算法。输入一个字符c，如果他的ASCII码在“A”和“Z”之间，则他是大写字母。根据ASCII码中，一个字母的大写和小写所对应的的数字相差32，只需将大写字母对应的ASCII码加上32，再转换成字母输出就是它对应的小写字母。

**【程序清单】**

#include <stdio.h>

int main(void)

{

char m;

printf("Please input a letter:");

putchar(((m = getchar()) >='A'&&m <= 'Z') ? m +'a'-'A': m);

return 0;

}

**【测试数据、测试理论结果及测试结果】**

表1-3 实验一必做题1.2.3测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | A | a | 如图1-5 |
| （2） | m | m | 如图1-6 |
| （3） | D | d | 如图1-7 |

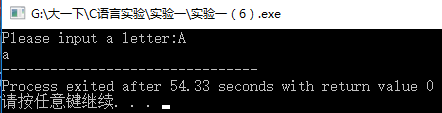


图1-5 实验一必做题1.2.3 样例（1）测试结果

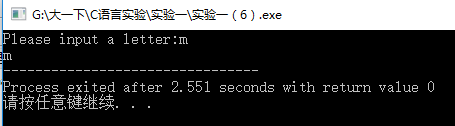


图1-6实验一必做题1.2.3 样例（2）测试结果

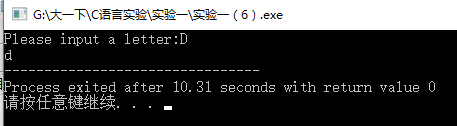


图1-7实验一必做题1.2.3 样例（3）测试结果

### 1.2.4 取出移位

**【题目】**

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。

**【解题思路】**

输入x，m，n，为了方便分析测试结果，x的输入采用16进制。

如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，首先x>>m，将要处理的n位移动到最右，再将上一步的结果左移位，用16进制输出结果；

如果m、n不满足上述要求，则显示输入错误信息。

**【程序清单】**

#include<stdio.h>

int main()

{

unsigned short x, m, n;

printf("Please input three unsigned numbers x,m,n\n

Remember( 0<=m<=15,1<=n<=16-m) \n");

scanf("%hx", &x);

scanf("%hx", &m);

scanf("%hx", &n);

if (m < 0 || m>15 || n<1 || n>(16 - m))

{

printf("error!Remember 0<=m<=15,1<=n<=16-m ");

}

else

{

x = (x << (16 - m - n)) &(~0<<16-n);

printf("x=%hx", x);

scanf("%hx", x);

}

return 0;

}

**【程序测试数据、测试理论结果及测试结果】**

表1-4 实验一必做题1.2.4测试数据、测试理论结果及测试结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 样例 | 程序测试数据 | | | 测试理论结果 | 测试结果 |
| x | m | n |
| （1） | 4680 | 7 | 4 | d000 | 如图1-8 |
| （2） | d583 | 13 | 5 | error！ | 如图1-9 |
| （3） | 5492 | 5 | 9 | 5200 | 如图1-10 |

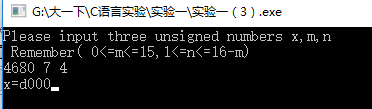


图1-8实验一必做题1.2.4样例（1）测试结果

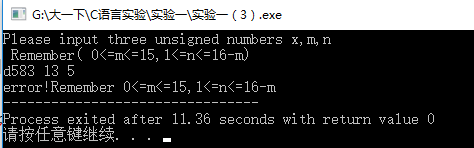


图1-9实验一必做题1.2.4样例（2）测试结果

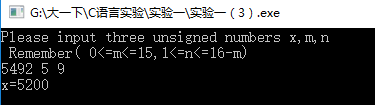


图1-10实验一必做题1.2.4样例（3）测试结果

### 1.2.5 IP地址解析

**【题目】**

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。

**【解题思路】**

输入一个无符号长整形数字，计算机将其转换成32位二进制，定义四个十六进制数r1、r2、r3、r4，再取这四个十六进制数分别与长整形数字的与运算结果，得到的就是IP地址的四部分，再通过移码后输出就获得了IP地址。

**【程序清单】**

#include <stdio.h>

#define r1 0xff000000

#define r2 0x00ff0000

#define r3 0x0000ff00

#define r4 0x000000ff

int main()

{

long int x,ip1,ip2,ip3,ip4=0;

printf("Please input a number:");

scanf("%ld",&x);

ip1=(r1&x)>>24;

ip2=(r2&x)>>16;

ip3=(r3&x)>>8;

ip4=(r4&x);

printf("%ld,%ld,%ld,%ld",ip4,ip3,ip2,ip1);

return 0;

}

**【程序测试数据、测试理论结果及测试结果】**

表1-5 实验一必做题1.2.5测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 676879571 | 211，92，88，40 | 如图1-11 |
| （2） | 56558584 | 248,3,95,3 | 如图1-12 |

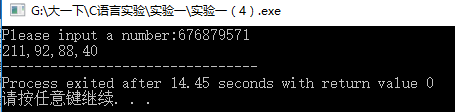


图1-11实验一必做题1.2.5样例（1）测试结果

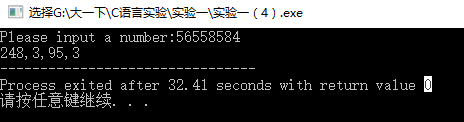


图1-12实验一必做题1.2.5样例（2）测试结果

## 自设题

**1.3.1 大小写字母转换**

**【题目】**

输入一个字母，若为大写字母，则转化为小写字母，若为小写字母，则转化为大写字母。

**【解题思路】**

利用大小写字母二进制码的差别。大小写字母仅对应的自右第5位相反，可利用与32（00010000）进行异或运算达到直接转化的目的，而不须经过判断。这样可以减少选择结构，简化代码。

**【程序清单】**

#include<stdio.h>

int main()

{

char m;

printf(“Please input a char:”):

scanf("%c", &letter);

printf("%c\n", letter ^ 32);

return 0;

}

**【程序运行结果】**



图1-13 实验一自设题运行结果

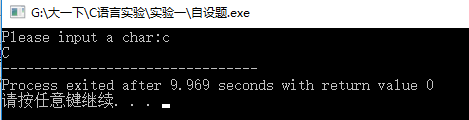


图1-14 实验一自设题运行结果

## 1.4 小结

在这次实验中，第一个程序改错题在不使用编译器的情况下没有将所有的错误修改过来，但是借助VS的帮助将所有的错误更正过来了，其中像5/9数据精度不够这种错误让我意识到程序编写的严谨性；经过这次改错我几乎掌握了程序出错的常见类型，如逻辑错误，字符错误等。

程序设计题中，熟练掌握了移码来变换数字，同时练习了运算符中的与运算。同时发现运算符的优先级和结合规则的重要性。在完成这几道程序设计题时的困难主要是一些语句格式的书写错误和字符的使用不当，尤其是进制的使用，如%hd和%hx。每次修改对照错误列表一个一个修改知道最终的成功。

# 2流程控制实验

## 2.1实验目的

通过完成实验，掌握复合语句、if语句、switch语句的使用；练习循环结构for、while、do-while语句的使用，从而熟练掌握for、while、do-while三种基本的循环控制语句的恰当使用，掌握重复循环技术。同时了解转移语句与标号语句。

## 2.2 必做题

### 2.2.1 源程序改错

**【题目】**

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。源程序如下。

1 #include <stdio.h>

2 int main()

3 { int i,n,s=1;

4 printf("Please enter n:");

5 scanf("%d",n);

6 for(i=1,i<=n,i++)

7 s=s\*i;

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

9 }

10 return 0;

【**错误原因分析及改错方案**】

经过调试和运行，发现这个程序一共存在三处错误,分析如下。

第3行错误，int i,n,s=1;处数据类型使用不恰当，在计算阶乘的时候，由于阶乘值会上升很快，所以不建议对s使用int类型，可以改为long或者double型。正确形式应为long i,n,s=1;。

第5行错误，scanf(“%d”,n);处scanf函数获取输入值时，引用的是一个变量地址，而不是变量本身。正确形式应为scanf(“%d”,&n);。

第10行错误，return 0;位置错误，return的返回值应该在main函数内部。故将return 0;移至第9行前面。

逻辑错误，当输入负数时未报错而是输出错误结果，所以在输入之后计算之前，加入如下负数报错代码

if (n < 0)

{

printf("Error!\n");

return 1;

}

**【修改后代码】**

#include <stdio.h>

int main()

{

long i,n,s=1;

printf("Please enter n:");

scanf("%d",&n);

if (n < 0)

{

printf("Error!\n");

return 1;

}

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

s=s\*i;

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

return 0;

}

【**测试数据、测试理论结果及测试结果**】

由于特殊数据有0、-1，再加上一组普通数据。共设计3组数据，对应3种情况。

表2-1 实验二必做题2.2.1测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据（n） | 测试理论结果（s） | 测试结果 |
| （1） | 8 | 40320 | 如图2-1 |
| （2） | 0 | 1 | 如图2-2 |
| （3） | -1 | Error！ | 如图2-3 |

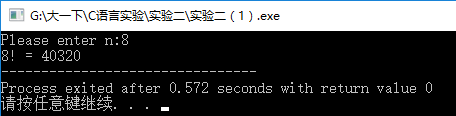


图2-1 实验二必做题2.2.1 样例（1）测试结果

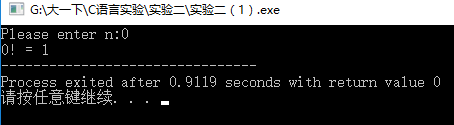


图2-2 实验二必做题2.2.1 样例（2）测试结果

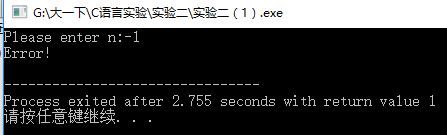


图2-3 实验二必做题2.2.1 样例（3）测试结果

### 2.2.2 源程序修改替换

**【题目】**

（1）修改上述计算s=n!的源程序，要求分别用while和do-while语句替换for语句。

（2）修改上述计算s=n!的源程序，要求输入改为“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.

**【改写算法流程图】**

（1）while型循环结构流程图和do-while型循环结构流程图如下图2-4和2-5，

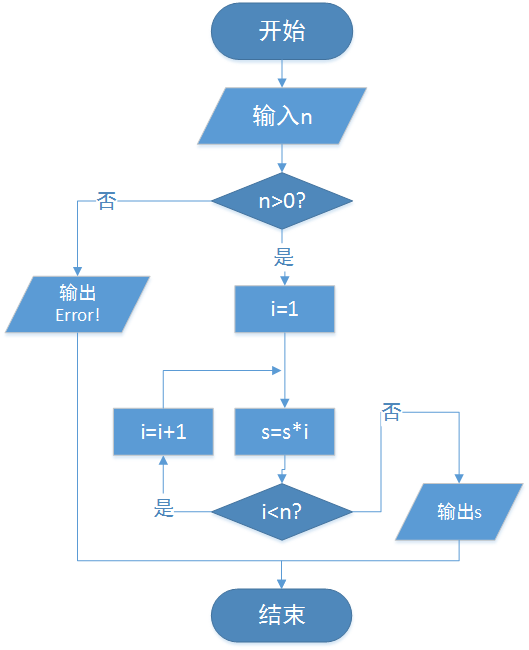
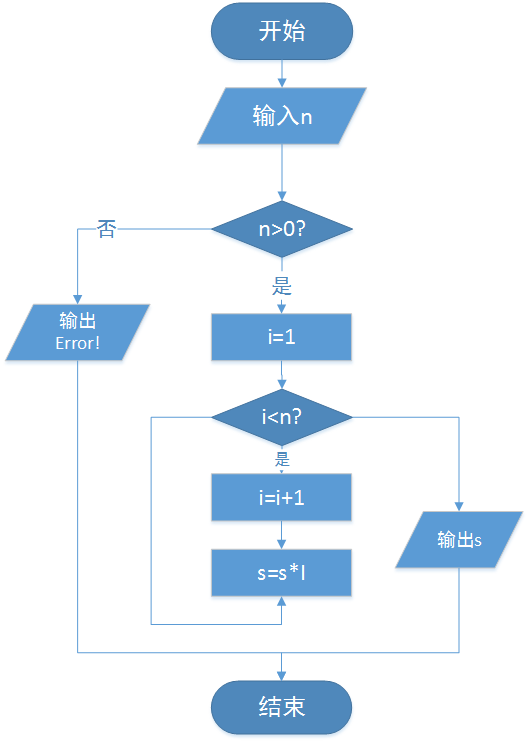


图2-4 while型循环结构流程图 图2-5 do-while型结构流程图

（2）同样计算n!并将n!赋值给不是s的值（比如说m），然后令m和s比较大小，找到符合条件的第一个m值，并返回n值。流程图如下图2-7，

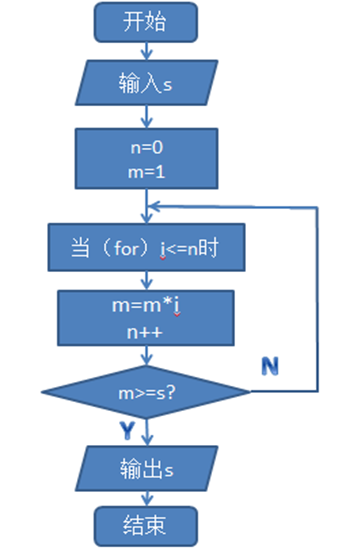


图2-7 实验二必做题2.2.2（2）流程图

**【改写程序清单】**

（1）while型循环结构替换for型的程序代码如下。

#include<stdio.h>

int main()

{

long i=1,n,s=1;

printf("Please enter n:");

scanf("%d",&n);

if (n < 0)

{

printf("Error!\n");

return 1;

}

while(i<=n){

s=s\*i;

++i;

}

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

return 0;

}

do-while型循环结构替换for型的程序代码如下。

#include<stdio.h>

int main()

{

long i=1,n,s=1;

printf("Please enter n:");

scanf("%d",&n);

if (n < 0)

{

printf("Error!\n");

return 1;

}

do{

s=s\*i;

++i;

}while(i<=n);

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

return 0;

}

（2）根据流程图，写出的程序清单如下。

#include <stdio.h>

int main()

{

int i=1,n=1,s;

printf("Please enter s:");

scanf("%d",&s);

if (n < 0)

{

printf("Error!\n");

return 1;

}

do

{

n=n\*i;

++i;

}

while (n<=s);

i=i-1;

printf("%d",i);

return 0;

}

**【改写程序测试】**

(1.1)由于特殊数据有0、1，再加上一组普通数据。共设计3组数据，对应3种情况：

1、输入n=6时，应输出6!=720；测试结果如图

2、输入n=0时，应输出0!=1

3、输入n=-1时，应输出Error!

实际输出结果完全符合预测，如下图所示。

(1.2)共设计3组数据，对应3种情况：

1、输入n=6时，应输出6!=720

2、输入n=0时，应输出0!=1

3、输入n=-1时，应输出Error!

(2)共设计1组数据，对应1种情况：

1、输入721时，由于6!=720，故应输出n=7

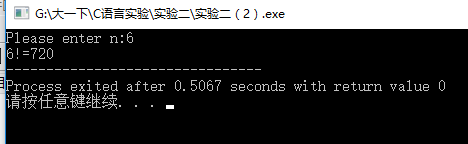


图2-8 实验二必做题2.2.2（1.1）样例1测试结果

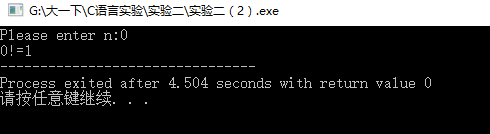


图2-9 实验二必做题2.2.2（1.1）样例2测试结果

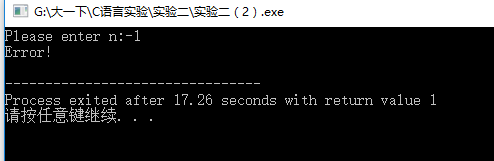


图2-10 实验二必做题2.2.2（1.1）样例3测试结果

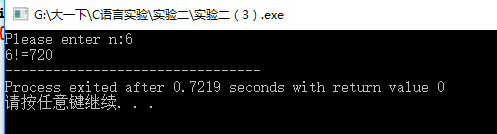


图2-11 实验二必做题2.2.2（1.2）样例1测试结果

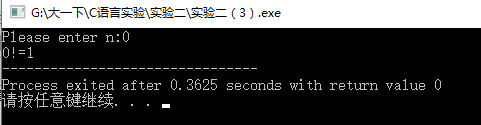


图2-12 实验二必做题2.2.2（1.2）样例2测试结果

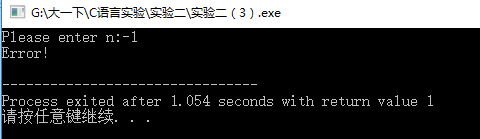


图2-13 实验二必做题2.2.2（1.2）样例3测试结果

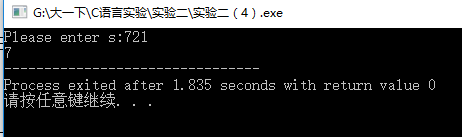


图2-14 实验二必做题2.2.2（2.1）样例1测试结果

### 2.2.3 工资税收问题

**【题目】**

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%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。

**【算法流程图】**

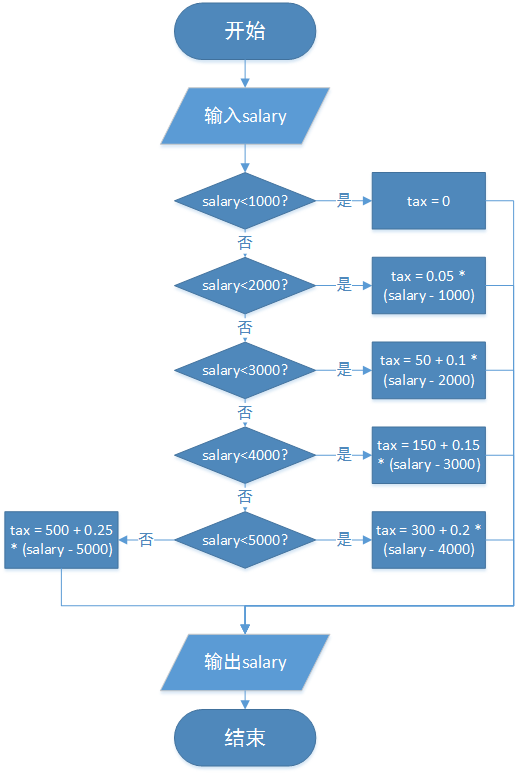
****

图2-15 实验二必做题2.2.3流程图

**【程序清单】**

（1）用if语句书写程序用的是分段函数的思维，根据流程图编写的程序如下。

#include<stdio.h>

int main()

{

float x, w,n1=0.00,n2=0.05,n3=0.10,n4=0.15,n5=0.20,n6=0.25;

printf("Please enter x:");

scanf("%f", &x);

if (x < 1000)

w = 0;

else

if (x >= 1000 && x < 2000)

w=1000\*n1+(x-1000)\*n2;

else

if (x >= 2000 && x < 3000)

w=1000\*n1+1000\*n2+(x-2000)\*n3;

else

if (x >= 3000 & x < 4000)

w=1000\*n1+1000\*n2+1000\*n3+(x-3000)\*n4;

else

if (x >= 4000 && x < 5000)

w=1000\*n1+1000\*n2+1000\*n3+1000\*n4+(x-4000)\*n5;

else

if (x > 5000)

w=1000\*n1+1000\*n2+1000\*n3+1000\*n4+1000\*n5+(x-5000)\*n6;

printf("%.2f", w);

return 0;

}

（2）在应用switch语句的时候，结合分段区间的特点设置合适的间断点，对不同区间的x进行不同的处理得到相应的税收。根据算法流程图程序如下。

#include<stdio.h>

int main()

{

float x,w,n1=0,n2=0.05,n3=0.10,n4=0.15,n5=0.20,n6=0.25;

printf("Please enter your salary x:");

scanf("%f",&x);

if(x<5000)

switch((int)x/1000)

{

case 4:w=1000\*n1+1000\*n2+1000\*n3+1000\*n4+(x-4000)\*n5;break;

case 3:w=1000\*n1+1000\*n2+1000\*n3+(x-3000)\*n4;break;

case 2:w=1000\*n1+1000\*n2+(x-2000)\*n3;break;

case 1:w=1000\*n1+(x-1000)\*n2;break;

case 0:w=x\*n1;break;

}

else

w=1000\*n1+1000\*n2+1000\*n3+1000\*n4+1000\*n5+(x-5000)\*n6;

printf("%.2f",w);

return 0;

}

**【测试数据、测试理论结果及测试结果】**

由于这是一个分段函数问题，所以选择多个不同区间的数据x来进行测试来验证程序的正确性。经过将测试结果与理论结果的比对，程序正确。

表2-2 实验二必做题2.2.3测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| If-else型  样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 500 | 0 | 如图2-16 |
| （2） | 1500 | 75 |
| （3） | 2500 | 100 |
| （4） | 3500 | 225 |
| （5） | 4500 | 400 |
| （6） | 6000 | 750 |
| Switch型  样例 | 测试数据 | 测试理论结果 | 如图2-17 |
| （1） | 500 | 0 |
| （2） | 1500 | 75 |
| （3） | 2500 | 100 |
| （4） | 3500 | 225 |  |
| （5） | 4500 | 400 |
| （6） | 6000 | 750 |

为了一次性获得多组数据，降低测试复杂度，在程序代码中加入一个选择结构，while (scanf("%f", &salary) != EOF)。

使用上述测试数据，得到的测试结果如下。

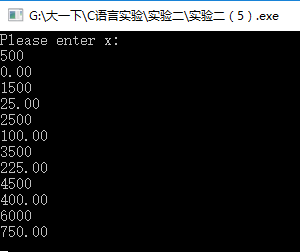


图2-16 实验二必做题2.2.3If-else 型样例测试结果

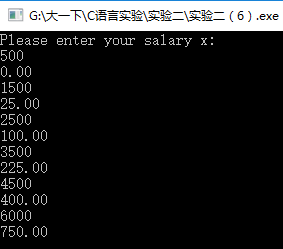


图2-17 实验二必做题2.2.3 switch型样例测试结果

### 2.2.4 空字符缩减

**【题目】**

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。

**【算法流程图】**

复制过程中若识别到非空格字符，则选择复制输出这个字符；当识别到空格字符时，先不进行复制输出，读取下一个字符；若下一个仍然是空格字符，则复制输出一个空格字符，然后停止输出；直到读取到一个非空格字符，复制输出这个字符；读取到换行符结束。流程图如下图2-18。

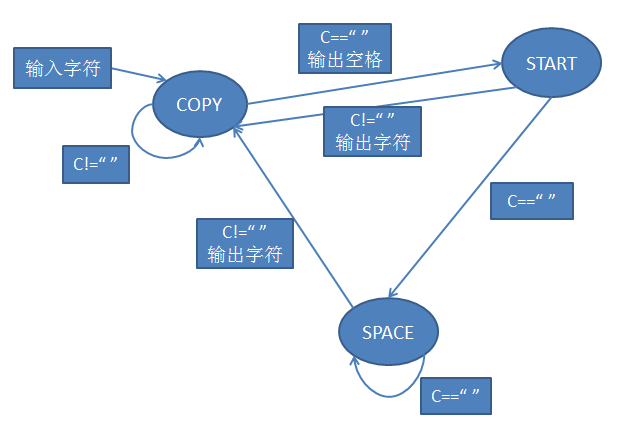


图2-18 实验二必做题2.2.4算法流程图

**【程序清单】**

根据上文的解题思路和算法流程图写出的程序如下

#include<stdio.h>

enum {COPY,SPACE,START};

int main()

{

char c;

int state=COPY;

printf("Please input a char:\n");

while((c=getchar())!='\n'){

switch(state){

case COPY:

if(c==' ')state=START;

else putchar(c);

break;

case START:

putchar(' ');

if(c==' ')state=SPACE;

else

{state=COPY;putchar(c);}

break;

case SPACE:

if(c!=' ')

{state=COPY;putchar(c);}

else state=SPACE;

break;

}

}

return 0;

}

**【测试数据、测试理论结果及测试结果】（注明：数据开头结尾处的□表示空格）**

表2-3 实验二必做题2.2.4测试数据、测试理论结果及测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 | 备注 |
| （1） | Hello world ！□ | Hello world ！ | 如图2-19 | 结尾留空格检验状态基是否初始化 |
| （2） | □Hi! | □Hi! | 如图2-20 | 开头留一个空格检验状态基初始化 |
| （3） | □□How are you ? | □How are you ? | 如图2-21 | 开头留许多空格检验 |

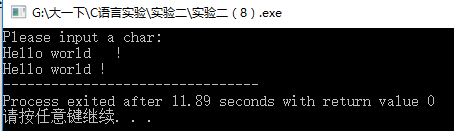


图2-19 实验二必做题2.2.4 样例（1）测试结果

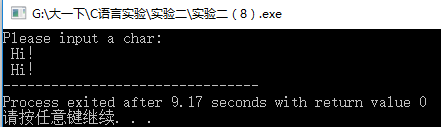


图2-20 实验二必做题2.2.4 样例（2）测试结果

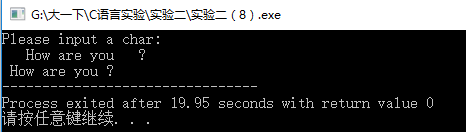


图2-20 实验二必做题2.2.4 样例（3）测试结果

### 2.2.5 杨辉三角

【题目】

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。

【解题思路】

根据所给公式利用循环实现杨辉三角数的计算，根据题目中特别提到的输出格式，从而将每个数字设置为四个字符位，并循环输出空格符，实现金字塔效果。

【算法流程图】

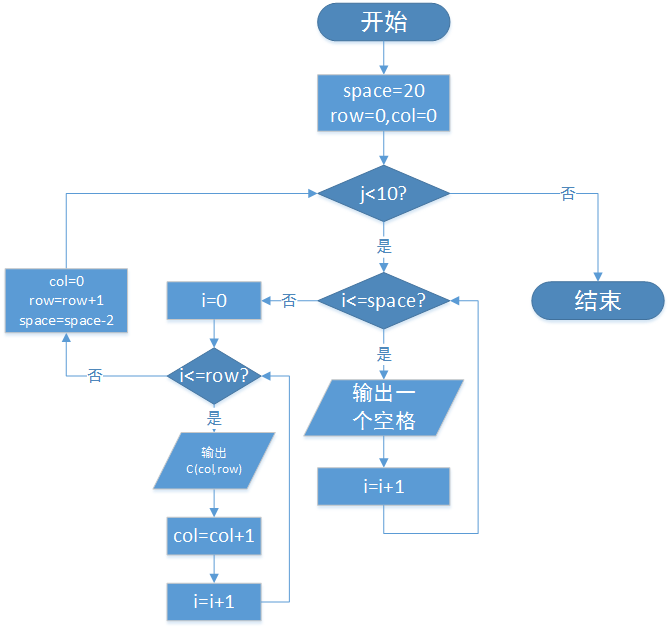


图2-22 实验二必做题2.2.5流程图

【程序清单】

根据上述思路写出的程序清单如下。

#include <stdio.h>

#define n 10

int main(void)

{

int i,j,k,c,m;

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

{

for(m=0;m<=n-i;m++){

printf(" ");

}

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

{

if(j==0)c=1;

else

c=c\*(i+1-j)/j;

printf("%4d",c);

}

printf("\n");

}

return 0;

}

**【程序运行结果】**

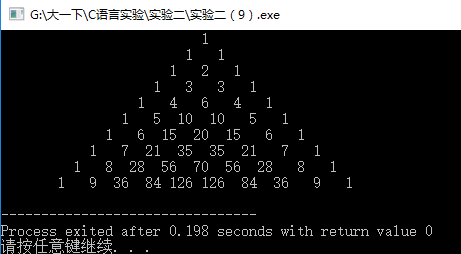


图2-23 实验二必做题2.2.5运行结果

### 2.2.6 整数逆转

**【题目】**

编写一个程序，将用户输入的任意正整数逆转。

**【算法流程图】**

多次除10取余，每次都分别输出余数，然后除以10得到的整数部分并赋给原数重复计算。根据这种算法写出的流程图如下图2-23。

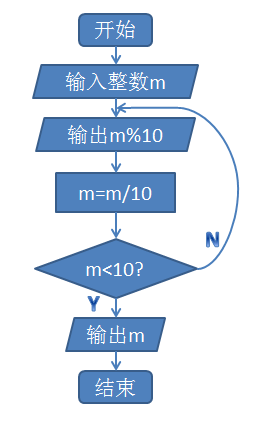


图2-24 实验二必做题2.2.6 流程图

**【程序清单】**

根据上面的流程图写出的程序如下。

#include<stdio.h>

int main()

{

int x,m;

printf("Please enter a positive integer x:");

scanf("%d",&x);

while(x>0)

{

m=x%10;

x=x/10;

printf("%d",m);

}

return 0;

}

【**测试数据、测试理论结果及测试结果**】

表2-4 实验二必做题2.2.6测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 456956 | 659654 | 如图2-25 |
| （2） | 156 | 651 | 如图2-26 |

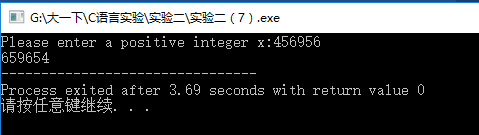


图2-25 实验二必做题2.2.6 样例（1）测试结果

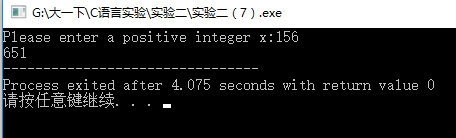


图2-26 实验二必做题2.2.6 样例（2）测试结果

## 2.3 选做题

### 2.3.1 牛顿迭代

**【题目】**

编写并上机调试运行能实现以下功能的程序。

编写一个程序，用牛顿迭代法求方程满足精度e=**10-6**的一个近似根，并在屏幕上输出所求近似根。

牛顿迭代法求方程近似根的迭代公式为：



其中,是函数f(x)的导函数。牛顿迭代法首先任意设定的一个实数来作为近似根的迭代初值x0，然后用迭代公式计算下一个近似根x1。如此继续迭代计算x2, x3, …, xn, 直到，此时值xn即为所求的近似根。

**【算法】**

这个程序的设计主要运用的迭代的思想，我设计的程序中，运用的迭代公式为了将计算简单化，令x为0。同时将迭代公式((((3\*x-4)\*x)-5)\*x+13)/((9\*x-8)\*x-5)这种方法表达可以降低计算次数。最后程序如下。

**【程序清单】**

#include<stdio.h>

#include<math.h>

#define EPS 1e-6

int main(void)

{

double x,d;

printf("input initial root:\n");

scanf("%1f",&x);

do{

d=-((((3\*x-4)\*x)-5)\*x+13)/((9\*x-8)\*x-5);

x += d;

}while(fabs(d)>EPS);

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

return 0;

}

**【程序运行结果】**

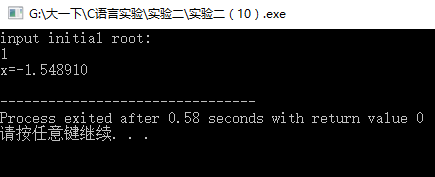


图2-27 实验二选做题运行结果

将x=-1.548910带入方程，发现成立。因此，此输出符合要求，结果正确。

## 2.4 自设题

**2.4.1 找特殊数字**

**【题目】**

编程找出这样的数字，它是三位数，它的三个数字的立方和等于它本身。

**【程序清单】**

#include<stdio.h>

int main()

{

int x,a,b,c;

for(x=100;x<1000;x++)

{

a = x%10;

b = (x/10)%10;

c = x/100;

if(a\*a\*a+b\*b\*b+c\*c\*c==x)

printf("%5d",x);

}

return 0;

}

**【测试结果】**

程序输出结果预测为：153 370 371 407，实际输出结果如下图2-28。

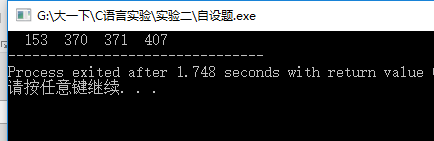


图2-28 实验二自设题输出结果

本次自设题中，运用了循环结构，通过这个实验熟练掌握了运算符%和/的基本用法和根本区别。

## 2.5 小结

源程序改错中遇到的那些错误也是我们平时容易出错的地方，通过这次实验，我总结出一套检查错误的方法：在输入过程中检查出大多数语法错误，或者拼写错误。接下来再检查一下逻辑是否有误，无误则进行多组数据的编译检验，确认最终无误。

源程序修改与替换题锻炼我们举一反三触类旁通的能力，在平时写程序时就应该思考对于一个问题有没有多钟解决方案，使用不同的逻辑结构多种尝试寻求最优解。

程序设计题则是自主的设计实践，在这次实验中我熟练掌握了流程图来表达程序的算法，同时增强对于所学知识尤其循环语句的熟练度和理解程度，学会了在什么地方使用哪种循环语句更加恰当。

在选做题中，运用了多项式的简便计算法的表示，基本掌握了迭代的程序的写法。

# 3函数与程序结构实验

## 3.1 实验目的

熟悉和掌握函数的定义、声明以及函数调用与参数传递方法，能够正确使用函数返回值类型的定义和返回值。将不同存储类型变量的灵活运用。熟悉多文件编译技术。

## 3.2 必做题

### 3.2.1 源程序改错

**【题目】**

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 }

【**错误原因分析及改错方案**】

经过调试和运行，发现这个程序一共存在三处错误,分析如下。

源程序中使用了函数sum\_fac，但是主函数前面没有进行声明；所以在使用前先声明函数sum\_fac，即在main函数之前；

第12行错误，源程序对变量fac使用\*=运算前未对变量fac赋初始值，使得源程序不能正常运行；只需在声明时对fac赋初始值1，即改为long fac = 1;

第14、15行错误，函数中的fac \*= i; s += fac;未括起，导致后一语句在循环外；则用花括号将这两个语句括起表示为代码块。

**【修改后代码】**

#include<stdio.h>

long sum\_fac(int n);

int main(void)

{

int k;

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

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

return 0;

}

long sum\_fac(int n)

{

long s = 0;

int i;

long fac = 1;

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

{

fac \*= i;

s += fac;

}

return s;

}

**【程序运行结果】**

该程序没有输入，直接输出。

输出结果预测如下：

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

实际输出结果如图，发现实际输出结果与预测完全相符。

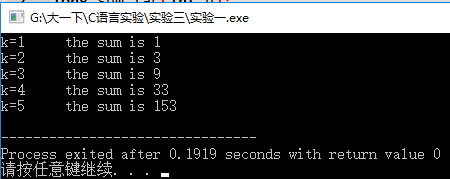


图3-1 实验三必做题3.2.1 运行结果

### 3.2.2 源程序修改替换

**【题目】**

（1）修改上述源程序中的sum\_fac函数，使其计算量最小。

（2）修改上述源程序中的sum\_fac函数，计算



**【解题思路】**

（1）由于按照题目的思路，计算乘法的次数为次计算加法的次数为n-1次，例如当n=5时，原题需计算19次，改写思路为：按照1+2\*（1+3\*（1+4\*（1+5）））来计算，总计算次数仅为7次，大大减少了计算量。同时，巧妙利用for循环将复杂的循环总结成一个语句，减少了代码复杂度。程序框图如图。

（2）由于不要求计算效率，沿用原程序的计算方法，只讲数据类型与计算方改变即可。

**【算法流程图】**

**（1）**

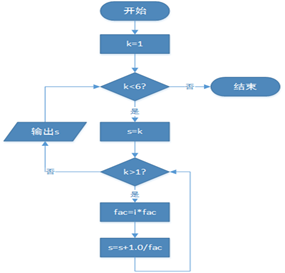
****

图3-2 实验三必做题3.2.2（1）流程图

**【改写代码】**

1. 修改上述源程序中的sum\_fac函数，使其计算量最小。

#include<stdio.h>

long sum\_fac(int n);

int main(void)

{

int k;

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

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

return 0;

}

long sum\_fac(int n)

{

long s;

for (s = n; n-- > 1; s = (s + 1) \* n);

return s;

}

（2） 修改上述源程序中的sum\_fac函数，计算



#include<stdio.h>

double sum\_fac(int n);

int main(void)

{

double k;

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

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

return 0;

}

double sum\_fac(int n)

{

double s = 0;

int i;

double fac = 1;

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

{

fac \*= i;

s += 1 / fac;

}

return s;

}

**【程序预测输出结果】**

（1）优化后计算结果应该与原程序相同，理论输出如下。

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

（2）计算结果预测输出如下

k=1.000000 the sum is 1.000000

k=2.000000 the sum is 1.500000

k=3.000000 the sum is 1.666667

k=4.000000 the sum is 1.708333

k=5.000000 the sum is 1.71666**7**

【**程序运行结果**】

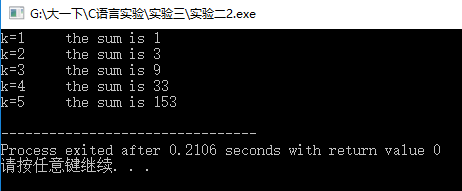


图3-3 实验三必做题3.2.2（1）运行结果

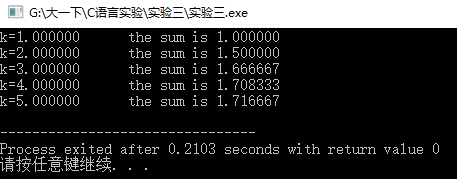


图3-4 实验三必做题3.2.2（2）运行结果

### 3.2.3 跟踪调试

**【题目】**

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

（1）刚执行完scanf("%d",&k);语句时，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函数时，watch窗口显示的是什么？

（4）当i=3时，从调用fabonacci函数到返回，n值如何变化？

#include<stdio.h>

long fabonacci(int);

int 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);

}

【**调试过程**】

（1）使用Visual Studio 2015加入断点进行调试，输入k=5，待scanf函数执行完之后。通过数据窗格（如图3-5）知：指针p的值是0x00cffb80，指向sum，i的值是-858993460

（2）按继续，刚执行完fabonacci函数返回后，光条停留在调用行。如图3-6所示。

（3）进入函数后，数据窗口中只显示显示局部变量n，因为只有n在这个域内可见。当然，可以通过设置“监视”来检测全局变量的变化，如图3-7

（4）一直继续执行，直到i=3时，刚刚调用时，n的值为3；第一次return fabonacci(n - 1) + fabonacci(n - 2);时，光标返回函数头，n值变为2，入栈；第三次return 1;时，光标调到函数尾，n的值变为3，出栈；之后return fabonacci(n - 1) + fabonacci(n - 2);，n值再次变为1，入栈；在此之后，再次执行return fabonacci(n - 1) + fabonacci(n - 2);，n变为3，出栈，返回2。

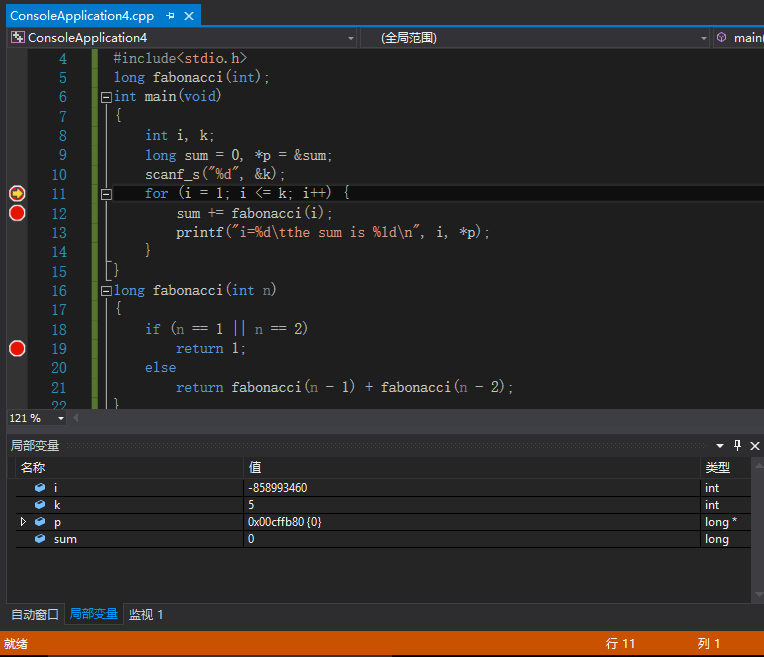


图3-5 实验三必做题3.2.3调试（1）

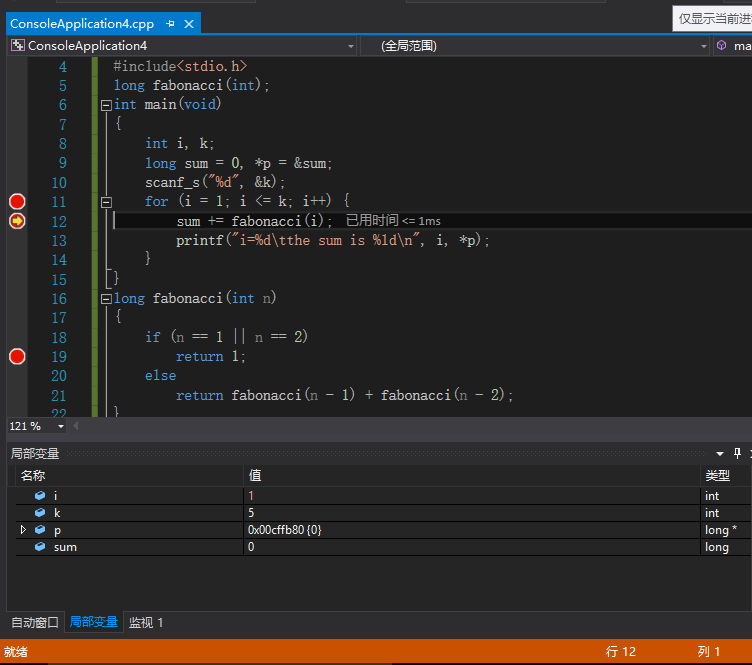


图3-6 实验三必做题3.2.3调试（2）

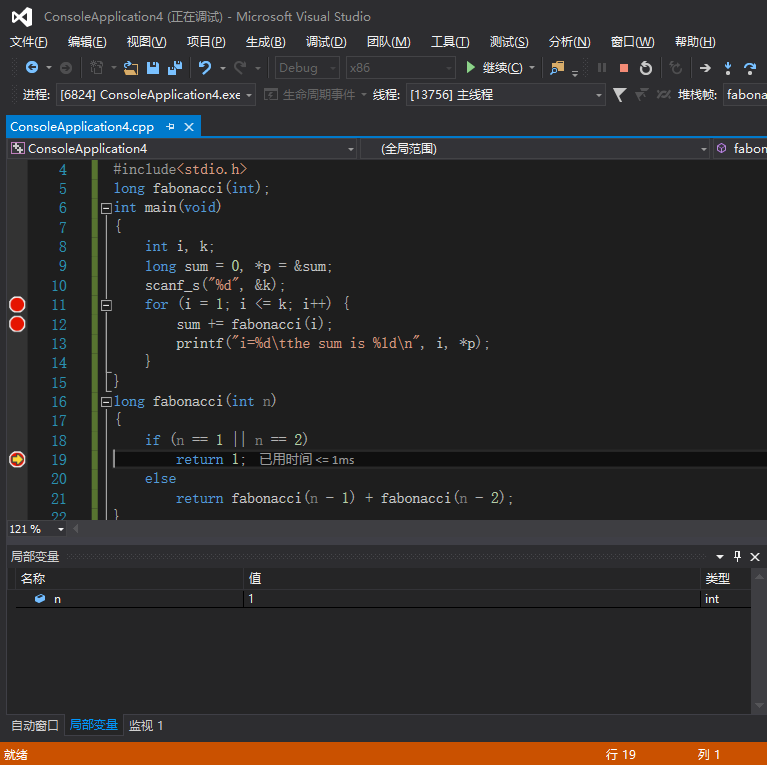


图3-7 实验三必做题3.2.3调试（3）

### 3.2.4 最大公约数

**【题目】**

编写一个程序,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，观察其递归过程。

**【解题思路】**

求解两个数的最大公约数并且使用递归思想的方法就是更相损减法，两个数a，b相减，得到结果c；然后再把b赋给a，c赋给b，执行相同的过程；但需要注意的是两个数a，b需要进行排序。其算法流程图如图

**【算法流程图】**

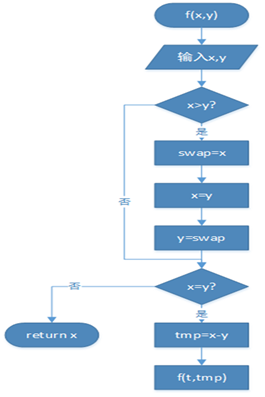
****

图3-9 实验三必做题3.2.4算法流程图

**【程序清单】**

#include<stdio.h>

int findCommonFactor(int x, int y);

int main()

{

int num1, num2;

scanf("%d %d", &num1, &num2);

if (num1 != 0)

{

int cf = findCommonFactor(num1, num2);

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

}

return 0;

}

int findCommonFactor(int x, int y)

{

int temp;

if(x<y)

{

int m=x;

x=y;

y=m;

}

if(x!=y)

{

temp=x-y;

findCommonFactor(y, temp);

}

else return x;

}

**【测试数据、测试理论结果及测试结果】**

**考虑到有x>y，x=y，x<y这三种情况，设计了如下三组样例进行测试。**

表3-1 实验二必做题3.2.4测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 45, 25 | 5 | 如图3-10 |
| （2） | 75， 150 | 75 | 如图3-11 |
| （3） | 15, 15 | 15 | 如图3-12 |

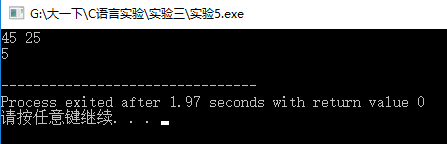


图3-10 实验三必做题3.2.4 样例（1）测试结果

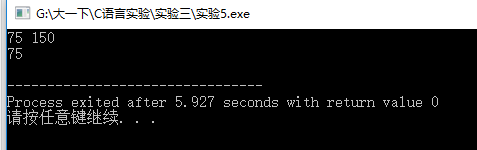


图3-11 实验三必做题3.2.4 样例（2）测试结果

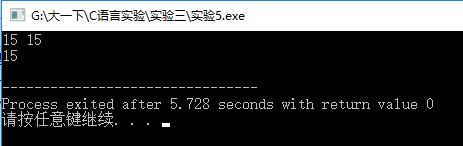


图3-12 实验三必做题3.2.4 样例（3）测试结果

### 3.2.5 验证哥德巴赫猜想

**【题目】**

编写一个程序，验证哥德巴赫猜想：一个大于等于4的偶数都是两个素数之和。

**【算法流程图】**

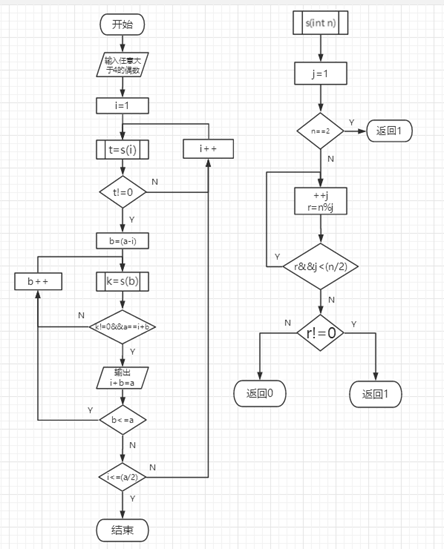


图3-13 实验三必做题3.2.5算法流程图

**【程序清单】**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int zhishu (int m)

{

int i;

if (m==2) return 1;

if(m%2==0) return 0;

for(i=3;i<=sqrt(m);i+=2)

{

if(m%i==0)

return 0;

}

return 1;

}

int main()

{

int a,b,c;

scanf("%d",&c);

a=2;

b=c-a;

for(a=2;a<=b;a++)

{

if (zhishu(a)==0) continue;

else

{

b=c-a;

if (zhishu(b)==1) break;

}

}

printf("%d=%d+%d\n",c,a,b);

}

**【测试数据、测试理论结果及测试结果】**

表3-2 实验二必做题3.2.5测试数据、测试理论结果及测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 | 备注 |
| （1） | 562 | 562=5+557 | 如图3-14 | 三位数 |
| （2） | 1000 | 3+97 | 如图3-15 | 四位数 |
| （3） | 12 | 12=5+7 | 如图3-16 | 二位数 |

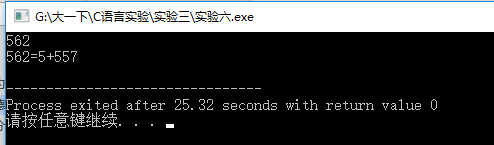


图3-14 实验三必做题3.2.5样例（1）测试结果

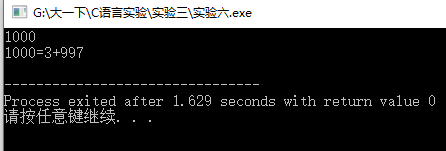


图3-15 实验三必做题3.2.5样例（2）测试结果

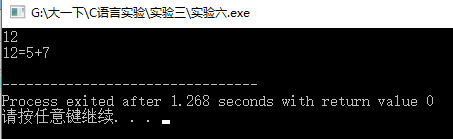


图3-16 实验三必做题3.2.5样例（3）测试结果

### 3.2.6 验证部分哥德巴赫猜想

**【题目】**

编写一个程序,证明对于在符号常量BEGIN和END之间的偶数哥德巴赫猜想成立。

**【算法流程图】**

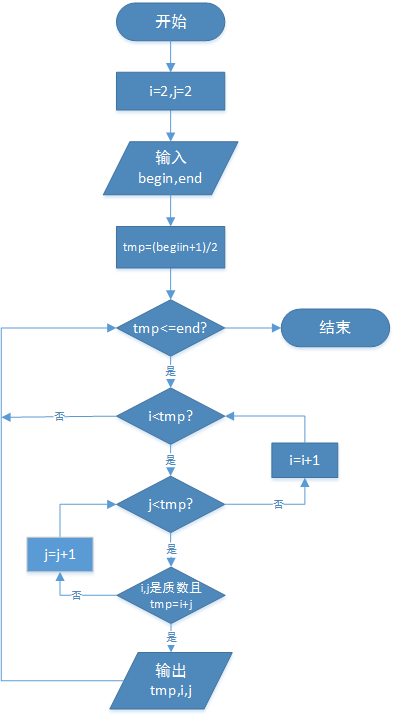
****

图3-17 实验三必做题3.2.6 算法流程图

**【程序清单】**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int zhishu (int m)

{

int i;

if (m==2) return 1;

if(m%2==0) return 0;

for(i=3;i<=sqrt(m);i+=2)

{

if(m%i==0)

return 0;

}

return 1;

}

int main()

{

int a,b,c,BEGIN,END;

printf("GOLDBACH'S CONIJECYURE:\n");

printf("Every even number n>=4 is the sum of two primes.\n");

scanf("%d %d",&BEGIN,&END);

if(BEGIN%2==0)

c=BEGIN;

else c=BEGIN+1;

while(c>=BEGIN&&c<=END)

{

a=2;

b=c-a;

for(a=2;a<=b;a++)

{

if (zhishu(a)==0) continue;

else

{

b=c-a;

if (zhishu(b)==1) break;

}

}

printf("%d=%d+%d\n",c,a,b);

c+=2;

}

}

**【测试数据、测试理论结果及测试结果】**

表3-3 实验二必做题3.2.6测试数据、测试理论结果及测试结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 样例 | 测试数据 | | 测试理论结果 | 测试结果 | 备注 |
| Begin | End |
| （1） | 4 | 12 | 4=2+2 6=3+3  8=3+5 10=3+7  12=5+7 | 如图3-18 | Begin与End都是偶数 |
| （2） | 5 | 13 | 6=3+3 8=3+5  10=3+7 12=5+7 | 如图3-19 | Begin与End都是奇数 |
| （3） | 8 | 15 | 8=3+5 10=3+7  12=5+7 14=3=11 | 如图3-20 | Begin是偶数，End是奇数 |
| （4） | 7 | 14 | 8=3+5 10=3+7  12=5+7 14=3=11 | 如图3-21 | Begin是奇数，End是偶数 |

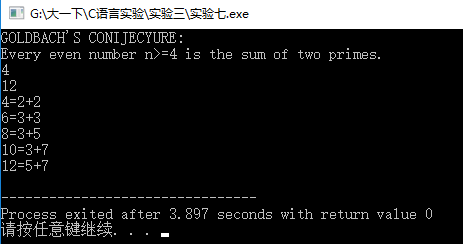


图3-18 实验三必做题3.2.6样例（1）测试结果

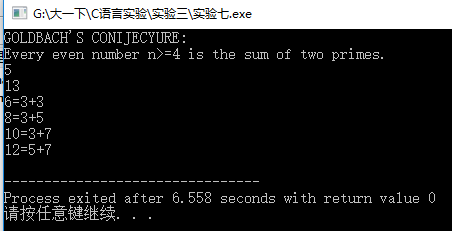


图3-19 实验三必做题3.2.6样例（2）测试结果

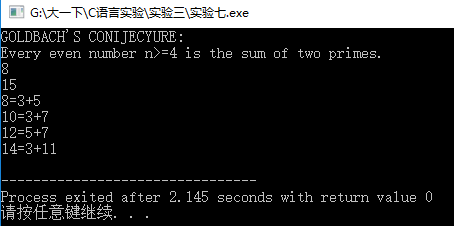


图3-20 实验三必做题3.2.6样例（3）测试结果

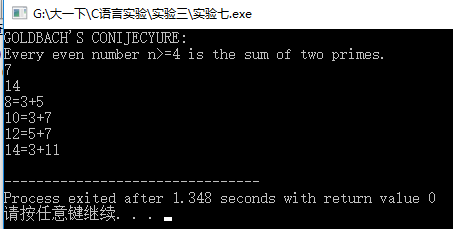


图3-21 实验三必做题3.2.6样例（4）测试结果

## 3.3 选做题

### 3.3.1 建立多源文件程序

**【题目】**

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

源文件file1.c的内容为：

#include "file.h"

int x, y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x = 10;

y = 20;

ch = getchar();

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

func1();

return 0;

}

源文件file2.c的内容为：

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

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

}

头文件file.h的内容为：

#include <stdio.h>

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); /\* func1函数原型 \*/

**【方法】**

按照教材上的一步步建立多源文件，从建立工程，去除main函数，写入头函数与两个源文件，编译器自动分类不同文件，当运行file1的时候就会调用file.h头文件，和file2中的funcl函数，如当输入1时，则此时x=10，y=20，ch=1,输出此时的值，之后跨文件调用file2.c的funcl函数，x=11，y=21，ch=2，并输出，运行结果如图

**【运行结果】**

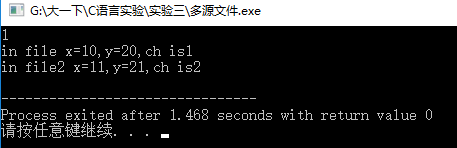


图3-22 实验三选做题运行结果

## 3.4 小结

在源程序修改替换中，要求更好的优化算法。于是借助秦九韶算法的思想，设计了一套求阶乘的简单算法，减少了大半的计算量。同时，巧妙地使用for循环一句代码完成了整个函数而又不失效率，体会到了代码之美。跟踪调试中，通过逐步调试体会到了迭代的思想；选做题中，了解了头文件与源文件的联系与区别，同时在自设题中联系了这种将源文件与头文件分开封装的方法，更深的体会到这种方法。在验证哥德巴赫猜想中，体会到了计算机在其他领域的应用以及发展前景。

# 4编译预处理实验

## 4.1实验目的

掌握文件包含、宏定义、条件编译、assert宏的使用；练习带参数的宏定义、条件编译的使用；学会使用assert；能够使用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 必做题

### 4.2.1 源程序改错

**【题目】**

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 }

【**错误原因分析及改错方案**】

经过调试和运行，发现这个程序一共存在三处错误,分析如下。

第11行错误，printf函数首字母大写；正确形式应为将printf函数改为全部小写。

第2行错误，第一个#define SUM a+b，a，b未加括号，导致宏展开时出现错误；正确形式应该为#define SUM ((a)+(b))。

第3行错误，第二个#define DIF a-b，a，b未加括号，导致宏展开时出现错误；正确形式应该为#define DIF ((a)-(b))。

第4行错误，SWAP宏定义不能达到交换的效果；正确形式应该为#define SWAP(a,b) t=a,a=b,b=t。

源程序未定义变量a，故需要在之前定义变量a，即改为int a, b, t。

**【修改后代码】**

#include <stdio.h>

#define SUM (a+b)

#define DIF (a-b)

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

int main()

{

int a, b, t;

printf("Input two integers a, b:");

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

printf("DIF=%d",DIF);

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

SWAP(a,b);

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

return 0;

}

【**测试数据、测试理论结果及测试结果】**

表4-1实验四必做题4.2.1测试数据、测试理论结果及测试结果

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 样例 | 测试数据 | | 测试理论结果 | | | | | 测试结果 | 备注 |
| a | b | SUM | DIF | a | b | Result |
| （1） | 9 | 12 | 21 | -3 | 12 | 9 | -63 | 如图4-1 | a<b |
| （2） | 12 | 12 | 24 | 0 | 12 | 12 | 0 | 如图4-2 | a=b |
| （3） | 13 | 8 | 21 | 5 | 8 | 13 | 105 | 如图4-3 | a>b |

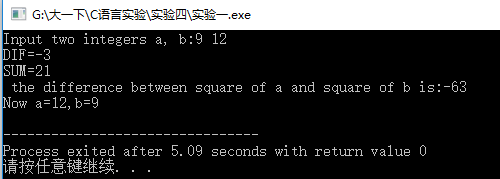


图4-1 实验四必做题4.2.1样例（1）测试结果

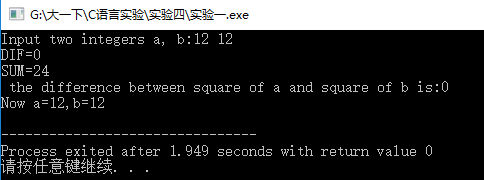


图4-2 实验四必做题4.2.1样例（2）测试结果

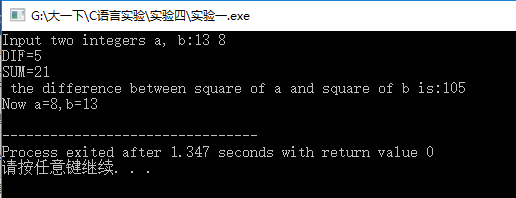


图4-3 实验四必做题4.2.1样例（3）测试结果

### 4.2.2 源程序修改替换

**【题目】**

下面是用函数实现求三个数中最大数、计算两数之和的源程序。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。

#include<stdio.h>

int main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

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

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

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

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

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

return 0;

}

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;

}

**【错误分析】**

原程序算法是输入三个数并判断最大数之后输出，之后再输入两个数，求两个数之和输出，都是用函数形式来实现。首先，其缺少头文件声明#include<stdio.h>,里面包含了printf()与scanf()函数的声明和定义。 其次，两个自定义函数没有声明，会导致编译器报错。再来因为主函数main已经声明返回值为空，则不需要后面的返回值零，可以去掉return 0;或者将void main()改为int main()。

**【修改替换后代码】**

#include<stdio.h>

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

float sum(float x, float y);

int main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

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

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

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

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

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

return 0;

}

float sum(float x, float y)

{

return x + y;

}

**【测试数据、测试理论结果及测试结果】**

表4-2 实验四必做题4.2.2 测试数据、测试理论结果及测试结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 样例 | 测试数据 | | | 测试理论结果 | 测试结果 | 备注 |
| a | b | c |
| （1） | 10 | 9 | 1 | 10 | 如图4-4 | a最大 |
| （2） | 8 | 12 | 7 | 12 | 如图4-5 | b最大 |
| （3） | 7 | 5 | 12 | 12 | 如图4-6 | c最大 |

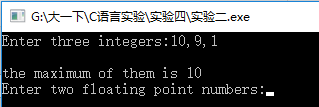


图4-4 实验四必做题4.2.2样例（1）测试结果

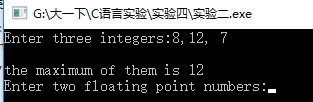


图4-5 实验四必做题4.2.2样例（2）测试结果

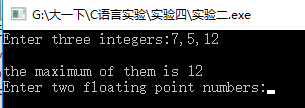


图4-6 实验四必做题4.2.2样例（3）测试结果

### 4.2.3 跟踪调试

**【题目】**

下面程序利用R计算圆的面积s，以及面积s的整数部分。

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。

#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;

}

**【错误分析】**

缺少头文件#include<stdio.h>和#include<assert.h>,会导致printf()函数,scanf()函数和assert()函数没有声明和定义，导致错误。

**【修改后代码】**

#include<stdio.h>

#include<assert.h>

int integer\_fraction(float x);

#define R

void main(void)

{

float r, s;

int s\_integer = 0;

printf("input a number: ");

scanf\_s("%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;

}

**【调试过程】**

可以直接在函数decimal\_fraction内部设置设置两个断点，分别是刚进入函数decimal\_fraction和返回主函数处，刚进入函数时命中断点，如图4-7，根据局部变量窗口，此时x的值为78.5397491。在返回main时，根据局部变量窗口，i的值变为78，如图4-8

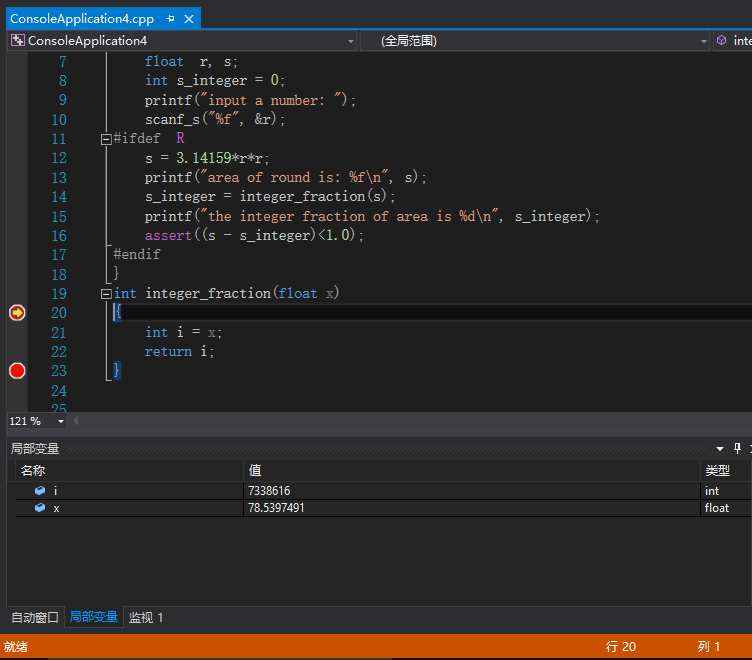


图4-7 实验四必做题4.2.3调试过程

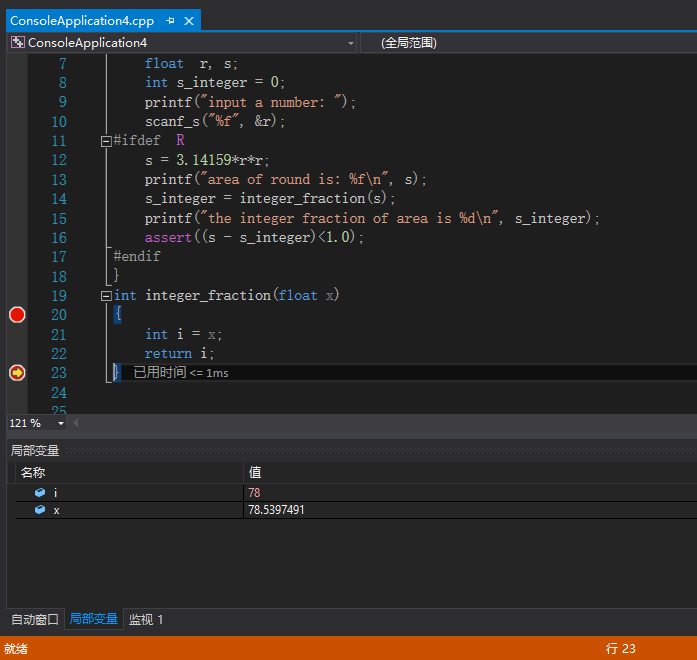


图4-8 实验四必做题4.2.3调试过程

### 4.2.4 计算三角形面积

**【题目】**

已知三角形的面积是，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,试编写一程序，用带参数的宏来计算三角形的面积。

**【算法流程图】**

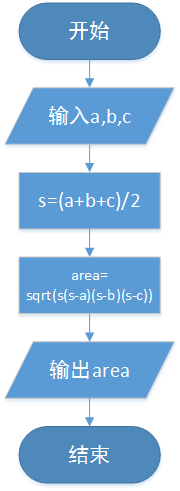


图4-9 实验四必做题4.2.4 算法流程图

**【程序清单】**

#include<stdio.h>

#include<math.h>

#define sum(a,b,c) (a+b+c)/2.0

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

int main()

{

float a,b,c,s,area;

printf("Please input three integer a,b,c:");

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

s=sum(a,b,c);

area=area(a,b,c);

printf("%f",area);

return 0;

}

**【测试数据、测试理论结果及测试结果】**

表4-3 实验四必做题4.2.4测试数据、测试理论结果及测试结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 样例 | 测试数据 | | | | 测试理论结果 | 测试结果 |
| a | b | | c |
| （1） | 3 | 4 | | 5 | 6.000000 | 如图4-10 |
| （2） | 4 | 5 | | 9 | 0.000000 | 如图4-11 |
| （3） | 10 | 12 | 14 | | 58.787754 | 如图4-12 |

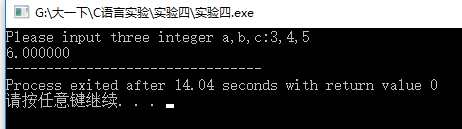


图4-10 实验四必做题4.2.4样例（1）测试结果

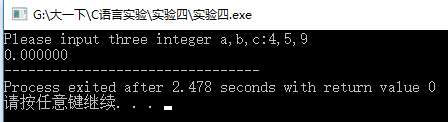


图4-11 实验四必做题4.2.4样例（2）测试结果

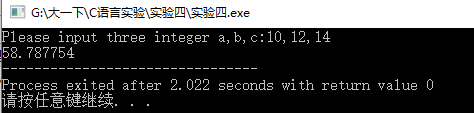


图4-12 实验四必做题4.2.4样例（3）测试结果

### 4.2.5 条件编译电报

**【题目】**

使用条件编译方法编写一程序,其功能要求是,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。

**【算法流程图】**

该程序采用单个输入字符遇到\n停止的方法，当define CHANGE不等于0的时候，执行if语句，由于大小写字母在内存中仅第五位相反，所以利用异或转换将大小写字母进行相互转化。程序框图如图

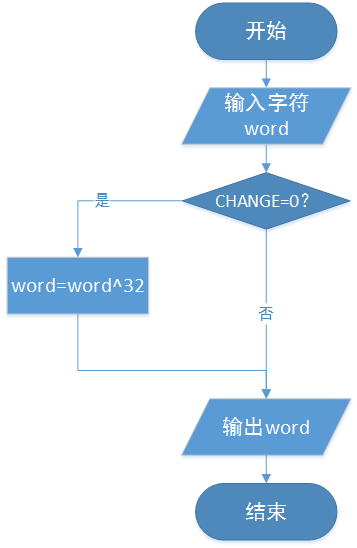


图4-13 实验四必做题4.2.5算法流程图

**【程序清单】**

#include<stdio.h>

#define CHANGE 1

int main()

{

int Word;

while ((Word = getchar()) != '\n')

{

#if CHANGE

printf("%c", Word ^ 32);

#else

printf("%c", Word);

#endif

}

printf("\n");

return 0;

}

**【测试数据、测试理论结果】**

本程序分为两种情况：CHANGE为0和1。根据这两种情况各设计一组测试数据如下：

1、当定义CHANGE为0时，输入HelloWorld，应输出原文HelloWorld；测试结果如图4-14。

2、当应以CHANGE为1时，输入helloworld，应输出大小写相互转化后的结果HELLOWORLD；测试结果如图4-15。

**【测试结果】**

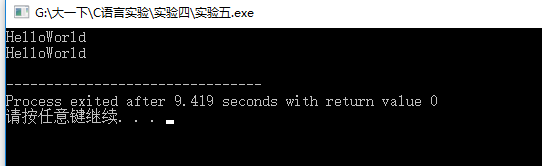
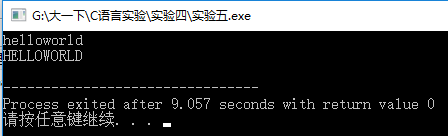


图4-14 实验四必做题4.2.5 样例1测试结果



**图4-15 实验四必做题4.2.5 样例2测试结果**

## 4.3小结

通过此次实验，我进一步的了解了预编译的原理和作用，以及头文件的使用。还有关于带参宏定义的使用。同时发现宏定义表达式带上括号的重要性，这样可以避免优先级不同产生一些逻辑错误。在程序修改替换实验中，学会了用宏定义代替某些简单函数的方法。跟踪调试中，对静态断言assert函数的了解有了进一步加深，也对数据的强制转换类型进行了深刻的理解。

# 5数组实验

## 5.1实验目的

本次实验需要掌握数组的说明、初始化和使用；一维数组作为函数参数时实参和形参的用法。同时学会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。熟练使用基于分治策略的二分查找算法和选择法排序算法，同时能够实现相关算法。

## 5.2 必做题

### 5.2.1 源程序改错

**【题目】**

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。

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 }

**【错误原因分析及改错方案】**

源程序存在两处错误，分别是函数声明错误和逻辑错误，具体情况如下。

第5行错误，调用函数sort时第一个参数与函数声明不相符；正确形式应该为void sort(a,10)；

第18行错误，题目要求将数组中的元素按升序排列，而源程序为降序排序，故正确形式应该为if（b[j]>b[j+1]）。

**【修改后代码】**

#include<stdio.h>

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

int main(void)

{

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

void sort(int[], int);

int i;

sort(a, 10);

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

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

printf("\n");

return 0;

}

void sort(int b[], int n)

{

int i, j, t;

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

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

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

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

}

**【测试数据、程序测试理论结果及测试结果】**

本程序有两种情况：10个数中有相同的数和10个数中没有相同的数。故将数组进行两侧更改，设计以下两组测试数据。

表5-1 实验五必做题5.2.1测试数据、程序测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 理论结果 | 实际结果 |
| （1） | 27, 13, 5, 32, 23, 3, 17, 43, 55, 39 | 3 5 13 17 23 27 32 39 43 55 | 如图5-1 |
| （2） | 9， 8，4，5，6，1，2，2，12，25 | 1 2 2 4 5 6 8 9 | 如图5-2 |

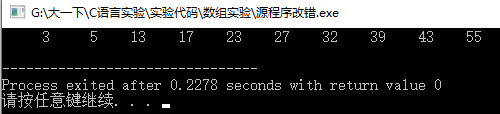


图5-1 实验五必做题5.2.1样例（1）运行结果

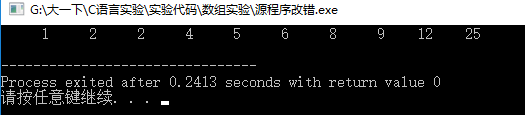


图5-2 实验五必做题5.2.1样例（2）运行结果

### 5.2.2 源程序完善、修改、替换

【题目】

(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。

源程序如下，

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

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

int i, j, k;

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

a[i] = i + 1;

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;

}

(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。

【答案】

（1） b[M-i] = j? a[j-1] : a[i-1] ;

for(k = --j; k < i; k++) a[k]=a[k+1] ;

（2）

【程序运行结果】

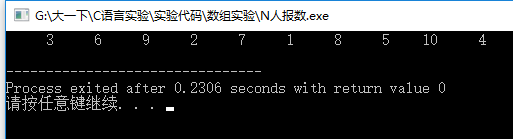


图5-3 实验五必做题5.2.2程序运行结果

### 5.2.3 跟踪调试

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。

#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';

}

**【调试过程】**

分别在进入函数strncat、for语句所在行及strncat函数块结束处设置断点。

当进入函数strncat中后，落到for语句所在行时，根据局部变量窗口观察i的值为23。如图5-4；当命中结束标志处的断点时，根据局部变量窗口，s为0x00aff944，指向字符串“The adopted symbol is ”，t为0x00aff920，指向字符串abcdefghijklmnopqrstuvwxyz，如图5-5

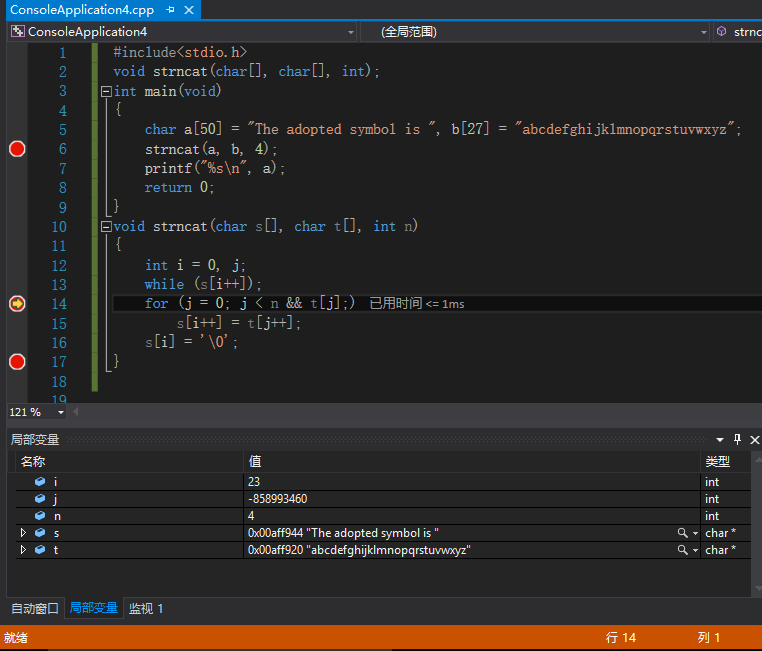


图5-4 实验五必做题5.2.3调试（1）

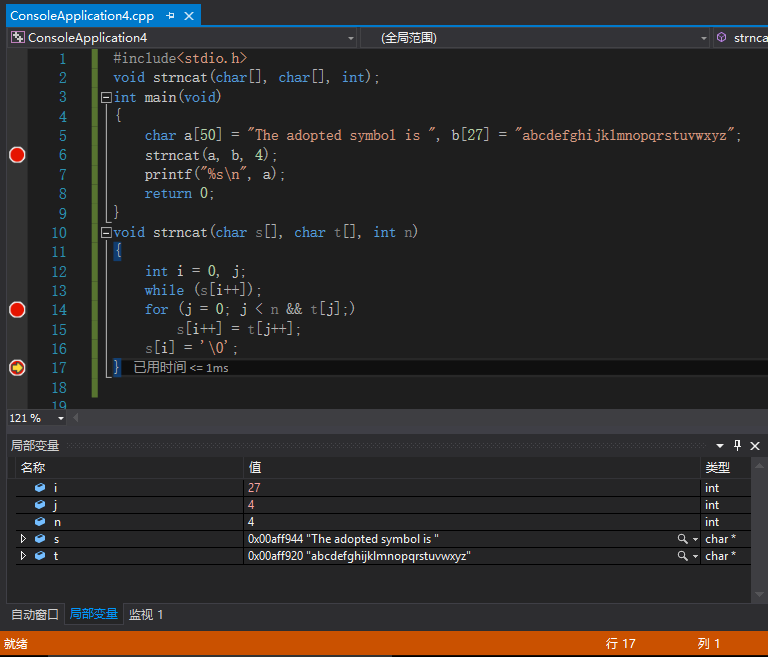


图5-5 实验五必做题5.2.3 调试（2）

**【错误分析及修改】**

根据调试过程可知，由于在while循环结束后i值为23，使得第一个字符串的a[22]即‘\0’没有被覆盖，所以输出时只有第一个字符串。

只需要将while的循环条件改为while（s[++i]!=’\0’）即可，修改后的输出结果如图5-6所示。

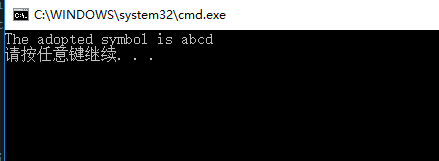


图5-6 实验五必做题5.2.3修改后运行结果

### 5.2.4 矩阵的运算

**【题目】**

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。

**【程序流程图】**

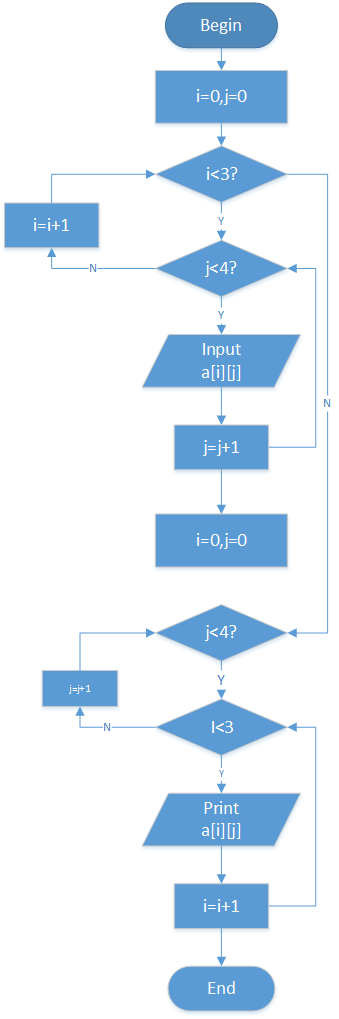


图5-7 实验五必做题5.2.4 算法流程图

**【程序清单】**

#include<stdio.h>

int main()

{

int a[3][4];

int i,j;

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

{

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

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

}

printf("原矩阵为：\n");

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

{

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

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

printf("\n");

}

printf("\n转置矩阵为：\n");

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

{

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

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

printf("\n");

}

return 0;

}

**【测试数据、测试理论结果及实际结果】**

由于该程序不需要进行数据计算，而只是交换输出的顺序，故情况较为简单，仅设计了一组测试数据。

输入为

1 2 5 4

2 8 4 6

9 7 4 1

预测输出为

原矩阵为：

1 2 5 4

2 8 4 6

9 7 4 1

转置矩阵为：

1 2 9

2 8 7

5 4 4

4 6 1

实际输出结果如下图5-8，发现与预测输出一致，故程序正确。

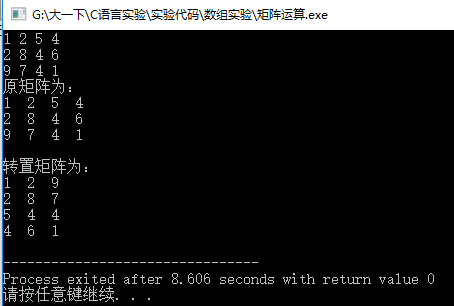


图5-8 实验五必做题5.2.4程序测试结果

### 5.2.5 整数二进制输出

**【题目】**

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。

**【解题思路流程图】**

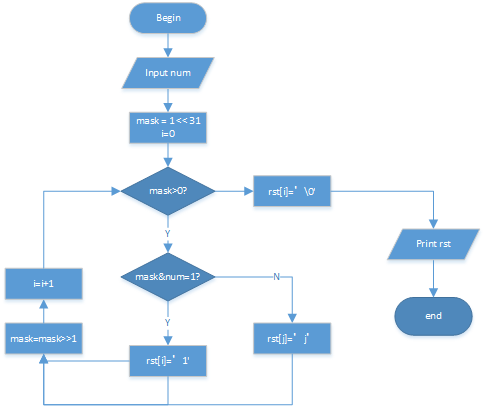


图5-9 实验五必做题5.2.5算法流程图

**【程序清单】**

#include<stdio.h>

int main()

{

unsigned int mask = 1 << 31;

int num;

char rst[40] = { 0 };

while (scanf("%d", &num) != EOF)

{

int i = 0;

for (mask = (1 << 31), i = 0; mask > 0; mask >>= 1, i++)

{

if (mask&num) //若此位置为1，则将rst中赋值为字符1

rst[i] = '1';

else rst[i] = '0';

}

rst[i] = '\0';//最后赋值结束符\0

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

}

return 0;

}

**【测试数据、理论结果及测试结果】**

表5-2 实验五必做题5.2.5测试数据、理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 理论结果 | 测试  结果 |
| （1） | 0 | 00000000000000000000000000000000 | 如  图  5  -10 |
| （2） | 124 | 00000000000000000000000000111110 |
| （3） | -2 | 11111111111111111111111111111110 |
| （4） | 2147483647 | 01111111111111111111111111111111 |
| （5） | -2147483647 | 10000000000000000000000000000001 |

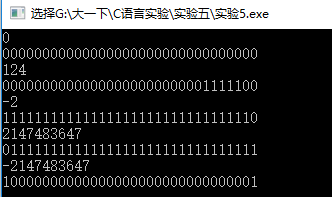


图5-10 实验五必做题5.2.5程序测试结果

### 5.2.6 C语言成绩排序查询

**【题目】**

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!”。

【解题算法流程图】

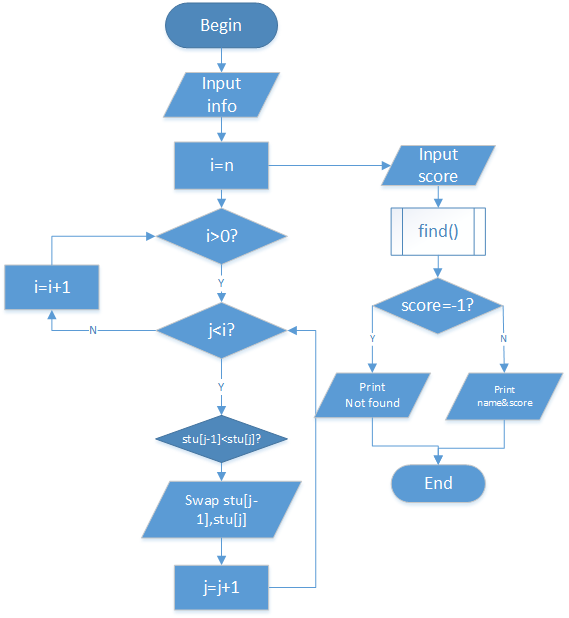


图5-11 实验五必做题5.2.6算法流程图

【程序清单】

#include<stdio.h>

typedef struct Stu {

char name[20];

float score;

}Stu;

Stu scan(Stu stu[], int begin, int end, float score);

const Stu null = { "Null",-1 };

int main()

{

int i, j, n;

Stu stu[50];

printf("How many students in class?\n");

scanf("%d", &n);

printf("Input all your students' name(no space!) and score(seperated by space) :\n");

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

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

//对输入的信息按照分数由高到低进行排序

for (i = n; i > 0; i--)

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

if (stu[j - 1].score < stu[j].score)

{

Stu tmp = stu[j - 1];

stu[j - 1] = stu[j];

stu[j] = tmp;

}

//调用二分查找函数进行查找

float \_score;

printf("Now, you can input a student's score:\n");

while (scanf("%f", &\_score) != EOF)

{

Stu one = scan(stu, 0, n - 1, \_score);

if (one.score == -1)

printf("Not Found!\n");

else printf("Found : %s %.2f\n", one.name, one.score);

}

return 0;

}

Stu scan(Stu stu[], int begin, int end, float score)

{

//当begin和end足够接近时，验证begin，end

if (begin >= end - 1)

{

if (stu[end].score == score)

return stu[end];

else if (stu[begin].score == score)

return stu[begin];

else return null;

}

//递归调用函数二分查找

int i = (begin + end) / 2;

if (stu[i].score == score)

return stu[i];

else if (stu[i].score < score)

return scan(stu, begin, i, score);

else if (stu[i].score > score)

return scan(stu, i, end, score);

}

**【测试数据及测试预测结果】**

此程序再第一次输入阶段尽量保证分数的区分度，而在第二次输入分数查找时，分为搜索中间、最大、最小进行检验，故设计以下6个学生样例：

A 79

B 88

C 72.5

D 90

E 84

F 81

在搜索阶段，输入72.5,90,84,81,79，88六个数进行测试，应分别输出Not Found！，C，D，E，F，A，B四个结果，实际输出如图5-12，完全符合预测结果。

**【测试结果】**

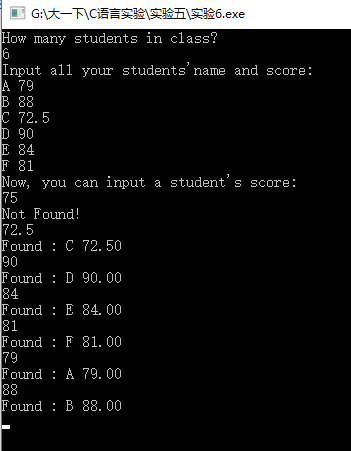


图5-12 实验五必做题5.2.6测试结果

## 5.3 选做题

**【题目】**

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。

**【算法流程图】**

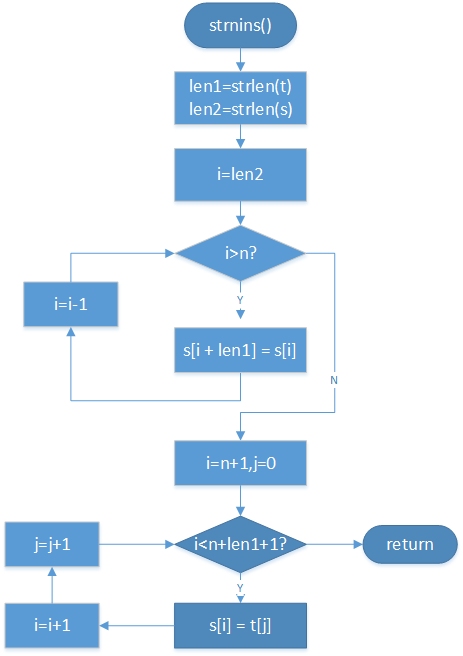


图5-13 实验五选做题算法流程图

**【程序清单】**

#include<stdio.h>

#include<string.h>

void strnins(char s[], char t[], int n);

int main()

{

char s[100], t[100];

int n;

printf("Input s:\n");

fgets(s, 100, stdin);

s[strlen(s) - 1] = s[strlen(s)];

printf("Input t:\n");

fgets(t, 100, stdin);

s[strlen(t) - 1] = s[strlen(t)];

printf("Input n:\n");

scanf("%d", &n);

strnins(s, t, n);

printf("%s", s);

return 0;

}

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

{

int len1 = strlen(t);

int len2 = strlen(s);

int i = 0, j = 0;

//移位空出位置

for (i = len2; i > n; i--)

s[i + len1] = s[i];

//插入字符串

for (i = n + 1, j = 0; i < n + len1 + 1; i++, j++)

s[i] = t[j];

}

## 5.4 自设题

**【题目】**

从键盘输入一个年月日，计算该日期在一年中是第几天。（用数组）

**【程序清单】**

#include<stdio.h>

int main()

{

int year,month,day,m,i;

m=0;

scanf("%d %d %d",&year,&month,&day);

int a[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};

int b[13]={0,31,29,31,30,31,30,31,31,30,31,30,31};

if((year%4==0&&year%100!=0||year%400==0))

{

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

m+=b[i];

m+=day;

}

else

{

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

m+=a[i];

m+=day;

}

printf("the day is the year of %d day",m);

return 0;

}

**【程序测试数据、测试理论结果及测试结果】**

表5-3 实验五 自设题测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 2000 3 22 | 82 | 如图5-14 |
| （2） | 1400 5 1 | 122 | 如图5-15 |
| （3） | 1456 7 9 | 191 | 如图5-16 |

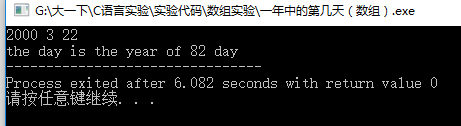


图5-14 实验五 自设题样例（1）测试结果



图5-15 实验五 自设题样例（2）测试结果



图5-14 实验五 自设题样例（3）测试结果

## 5.5小结

本次实验中收获最大的两个实验题就是N人报错和C语言成绩排序两个题目，前者考验了我们的思维，让我对三目判断语句有了深刻的理解，还学会了用标记法来代替一次次压缩数组的方法，提高了算法的效率。后者是一个综合性很强的程序，运用了二分法查找和递归算法这两个重要的算法，同时对数组的运用熟练程度有一定的要求。

# 6指针实验

## 6.1 实验目的

熟练掌握指针的说明、赋值、使用。通过掌握用指针引用数组的元素，熟悉指向数组的指针的使用。熟练掌握字符数组与字符串的使用，同时熟练运用指针数组及字符指针数组。掌握指针函数与函数指针的用法。运用带有参数的main函数。

## 6.2 必做题

### 6.2.1 源程序改错

**【题目】**

下面的源程序中是否存在错误？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源程序进行调试修改，使之能够正确执行。

#include "stdio.h"

int main(void)

{

float \*p;

scanf("%f",p);

printf("%f\n",\*p);

}

**【错误原因分析及改错方案】**

源程序中没有对悬挂指针进行赋值，所以需要定义一个变量，使指针p指向它，再对这个变量赋值，所以改为 float a，\*p=&a；

**【修改后代码】**

#include "stdio.h"

int main(void)

{

float a,\*p=&a;

scanf("%f",p);

printf("%f\n",\*p);

return 0;

**}**

**【测试数据、程序测试理论结果及测试结果】**

本程序非常简单，仅设计一组测试即可：输入5，预计输出5.000000，实际输出如图，完全符合预测。

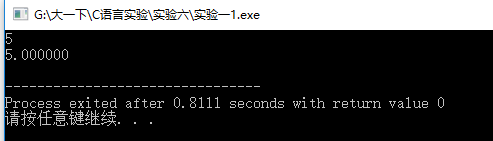


图6-1 实验六必做题6.2.1 测试结果

### 6.2.2 源程序修改替换

**【题目】**

（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。

（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.

#include "stdio.h"

#include "string.h"

int 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:

;

}

**【修改后代码】**

#include "stdio.h"

#include "string.h"

int main(void)

{

char \*(\*p)(char[],const char[]);

char a[80],b[80],c[160],\*result=c;

int choice,i;

do{

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

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

printf("\t\t3 exit.\n");

printf("\t\tinput a number (1-3) please!\n");

scanf("%d",&choice);

}

while(choice<1 || choice>5);

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

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

i=0;

gets(a);

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

i=0;

gets(b);

result=(\*p)(a,b);

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

down:

return 0;

}

**【测试数据、测试理论结果及测试结果】**

（1）本测试包含两种函数，每种函数设一组测试数据，同时包含输入异常的数据，最终测试退出，因此，共3组测试数据：

1、输入1，第一个字符hello，第二个字符world，将第二个字符复制到第一个字符并覆盖，结果为world；

2、输入2，第一个字符hello，第二个字符world!，将第二个字符连接到第一个字符后面，结果为helloworld!；

3、输入3，退出循环。

实际测试结果如图所示，完全符合要求。

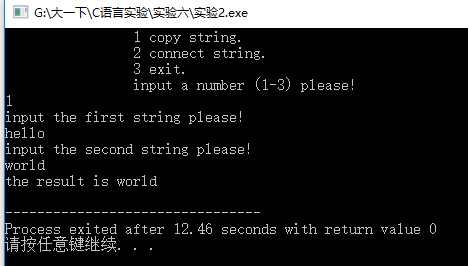


图6-2 实验六必做题6.2.2（1）样例1测试结果

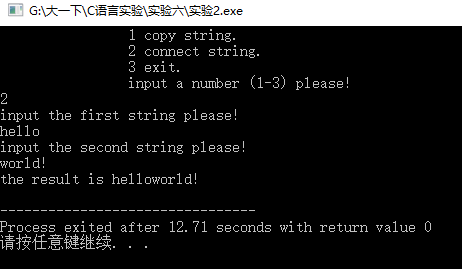


图6-3 实验六必做题6.2.2（1）样例2测试结果

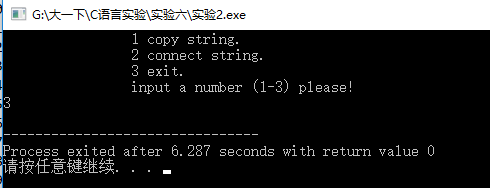


图6-3 实验六必做题6.2.2（1）样例3测试结果

（2）

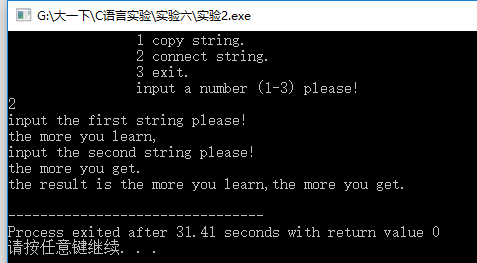


图6-4 实验六必做题6.2.2（2）运行结果

### 6.2.3 跟踪调试

**【题目】**

请按下面的要求对所给源程序进行操作，并回答问题和排除错误。

(1) 单步执行源程序。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) 排除源程序中的错误，使程序输出结果为： there is a boat on the lake.

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

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

}

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

{

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

;

return (s);

}

**【调试过程】**

调试过程共设置两个断点，分别在进入函数strcpy和返回main处。命中第一个断点时，读取局部变量窗口中的s的值，为0x012ffecc，指向的地址所储存的字符串无效,如图6-5所示；命中第二个断点时，读取局部变量窗口中的s的值，为0x012ffec9，指向的地址所储存的字符串无效,如图6-6所示。

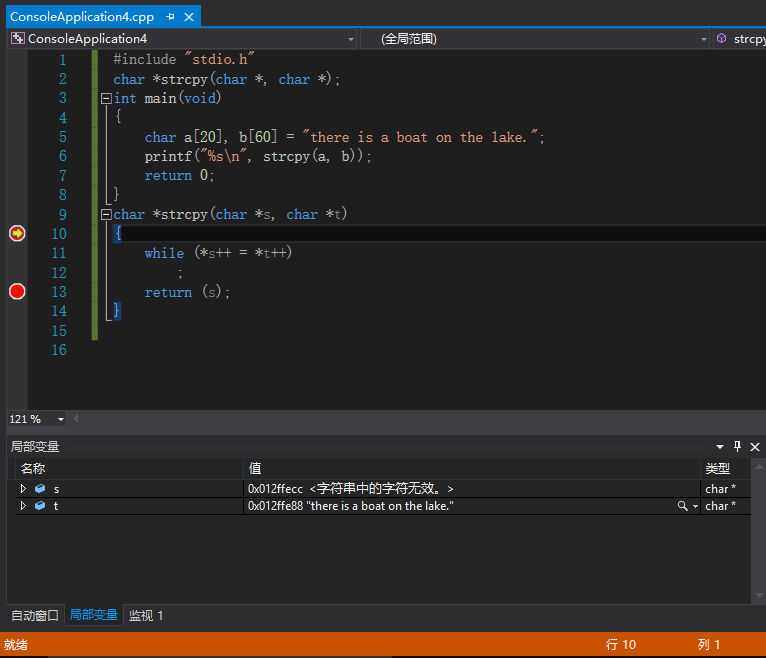


图6-5 实验六必做题6.2.3跟踪调试（1）

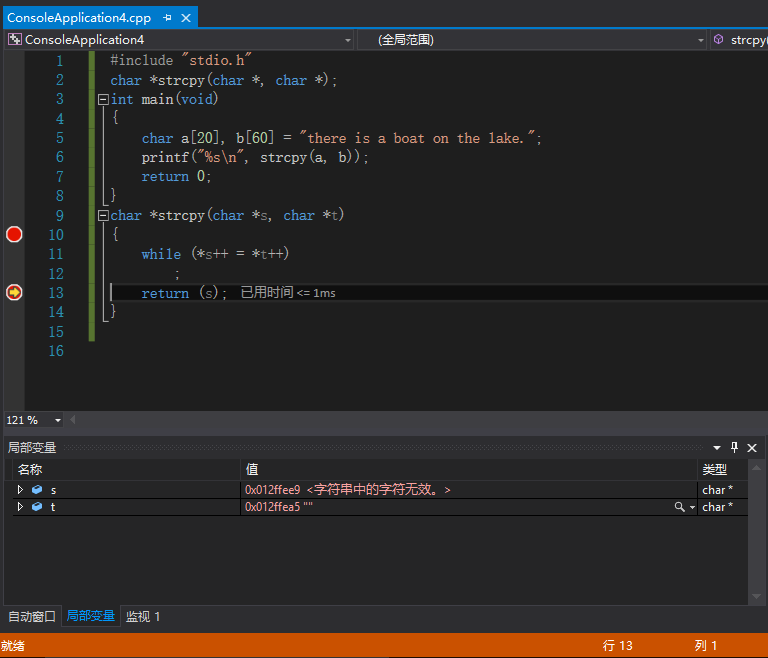


图6-6 实验六必做题6.2.3跟踪调试（2）

### 6.2.4 十六进制输出

**【题目】**

已知一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。试编写一个程序,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位并以十六进制数字字符的形式进行显示。

**【算法流程图】**

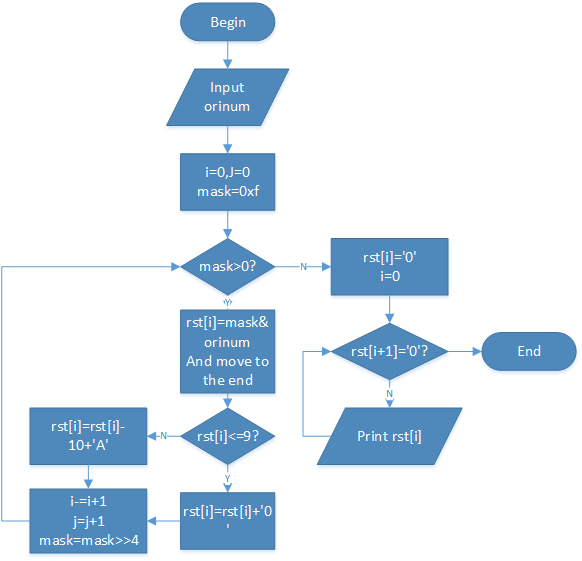


图6-7 实验六必做题6.2.4算法流程图

**【程序清单】**

#include<stdio.h>

#include<stdlib.h>

int main()

{

unsigned long mask = 0xflu << (8 \* sizeof(long) - 4);

char \*rst = (char\*)malloc(sizeof(char) \* 100);

long \*orinum = (long\*)malloc(sizeof(long) \* 100);

int n, i = 0, j, k;

printf("How many numbers are to be input:\n");

scanf("%d", &n);

printf("Please input the numbers:\n");

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

scanf("%ld", orinum + i);

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

{

mask = 0xflu << (8 \* sizeof(long) - 4);

for (i = 0, j = 0; mask > 0; mask >>= 4, j++, i++)

{

//依次取出每一位

\*(rst + i) = (char)((mask&(\*(orinum+k))) >> (sizeof(long) \* 8 - 4 - 4 \* j));

if (\*(rst + i) <= 9)

\*(rst + i) += '0';

else if (rst[i] > 9)

\*(rst + i) = \*(rst + i) - 10 + 'A';

}

\*(rst + i) = '\0';//最后赋值结束符\0

for (i = 0; \*(rst + i + 1) != '\0'; i++)

printf("%c ", \*(rst + i));

printf("%c\n", \*(rst + i));

}

return 0;

}

**【测试样例，测试理论结果及测试结果】**

为了方便测试，在代码中加入了一个循环结构，使得可以一次性进行多组数据的测试。

本次测试共有正数、负数、0三种情况，正负数中应测试正常数与大（小）于int类型的数，因此在32位编译器（long占四个字节）上设计以下5组测试。

表6-1 实验六必做题6.2.4测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试  结果 |
| （1） | 88888 | 00015B38 | 如图6-8 |
| （2） | 15 | 0000000F |
| （3） | 0 | 00000000 |
| （4） | -20 | FFFFFFEC |
| （5） | -77777 | FFFED02F |

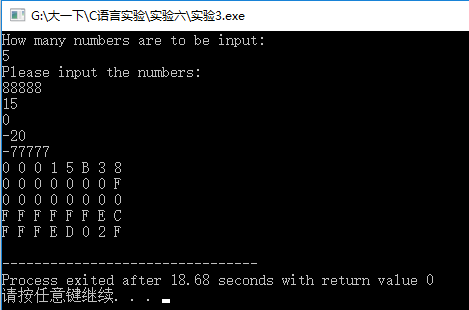


图6-8 实验六必做题6.2.4测试结果

### 6.2.5 去除空格空行

**【题目】**

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。

**【算法】**

首先确定你需要输入的行数，然后分别用gets函数输入字符串，并且定义一个指针数组以用来调用二维字符数组，然后检查每个字符串的空格，将多余空格合并之后，就按照顺序输出，空行不予输出。

**【程序清单】**

#include<stdio.h>

#include<string.h>

#define max 81

int check(char \*s);

int main()

{

int n,x,i,j=0;

printf("Please enter the n of your lines:");

scanf("%d",&n);

char a[n][max],\*p[n];

getchar();//吸收换行符

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

p[i]=a[i];//将二维数组的各个一级数组的首地址存入指针数组中

gets(p[i]);

}

printf("print:\n");

while(j<n)

{

x=check(p[j]);//检查数组并将其按要求将多余空格合并。

if(x){

printf("%s\n",p[j]);

j++;

}

else j++;//若此行为换行，则跳过。

}

printf("\nPress any key to quit...");

getch();

return 0;

}

int check(char \*s)

{

int i=0,k=0,y=0,flag=0;

if(\*s!='\0')//是否有输入字符。

{

y=1;

while(\*s!='\0')

{

if(\*s==' ')

{

if(flag!=1)//若为第一个空格则跳过压缩数组

flag=1;

else while(\*(s+i-1)!='\0')//压缩字符数组。使空格变为一个

{

\*(s+i-1)=\*(s+i);

i++,k=1;

}

if(!k) s++;//多余空格压缩完毕就跳到下个字符。

i=0,k=0;

}

else//不是空格。检查通过直接下个字符。

{

s++;

flag=0;

}

}

}

return y;

}

**【测试样例、测试理论结果】**

输入3表示三组数据，一组输有一个空格的a a a a a a a,7个a，一组数隔两个空格的，一组为空行，a a a a a a a,7个a。根据分析，则输出结果为两行有一个空格的a a a a a a a。

**【测试结果】**

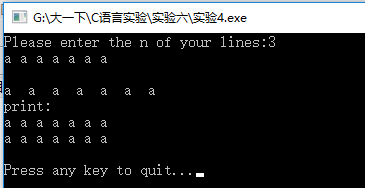


图6-9 实验六必做题6.2.5测试结果

### 6.2.6 命令行参数

**【题目】**

编写一个程序,输入n个整数，排序后输出。排序的原则由命令行可选参数-d决定，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。

**【程序代码】**

#include<stdio.h>

int main(int argc, int argv[])

{

int i, j, n;

int num[50];

scanf("%d", &n);

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

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

for (j = n - 1; j > 1; j--)

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

{

//升序

if (num[i - 1] > num[i])

{

int tmp = num[i - 1];

num[i - 1] = num[i];

num[i] = tmp;

}

//降序

if (num[i - 1] < num[i] && argc == 2)

{

int tmp = num[i - 1];

num[i - 1] = num[i];

num[i] = tmp;

}

}

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

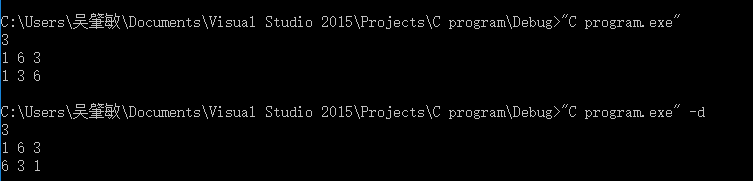
printf("%d ", num[i]);

printf("\n");

}

**【测试数据、测试理论结果及测试结果】**

测试数据分为两组，含-d与不含-d，在cmd中进入.exe所在文件夹，先输入"C program.exe"，然后测试输入3，数据为1、6、3，应按升序排序，即结果为1 3 6，然后输入相同的数据，应按降序排序，结果为6 3 1。实际输出结果如图6-10所示，符合预测。



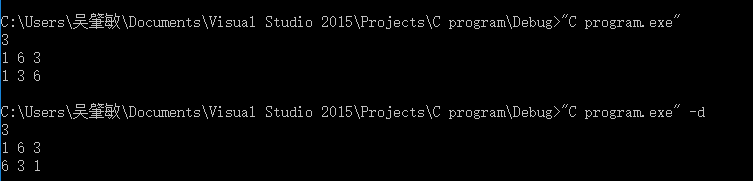


图6-10 实验六 6.2.6 测试结果

### 6.2.7 成绩处理

**【题目】**

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M门课程的成绩并且都存放到相应的数组中。试编写下列函数：

(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。）

**【算法流程图】**

本程序主要分为1、输入数据，2、计算每个人平均分，3、计算每个课程平均分，4、统计每个课程低于平均分的人数，5、统计每个课程不及格的人数，统计高于90分人数。本程序算法为简单的查找算法，重点考察指针表示数组。且由于3、4、5算法基本相同，故以3为例。具体程序框图如图~图。

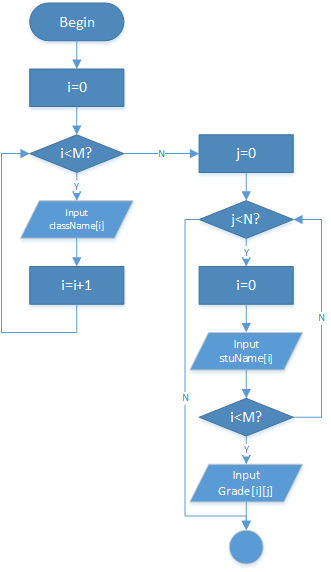


图6-11 实验六必做题6.2.7 算法流程图（1）

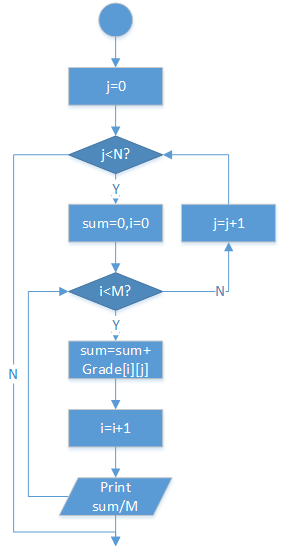


图6-12 实验六必做题6.2.7 算法流程图（2）

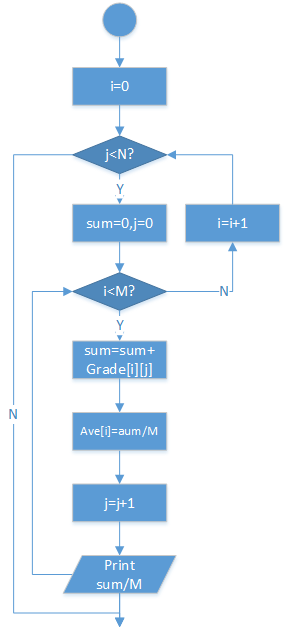


图6-13 实验六必做题6.2.7 算法流程图（3）

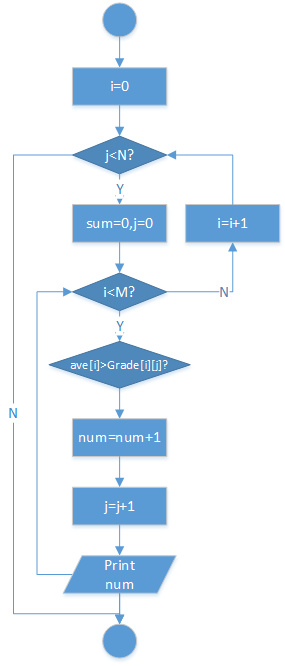


图6-14 实验六必做题6.2.7 算法流程图（4）

**【代码清单】**

#include<stdio.h>

#include<stdlib.h>

#define M 5 //number of classes

#define N 5 //number of students

int main()

{

char \*className = (char\*)malloc(8\*sizeof(char)\*M);

char \*\*studentName = (char\*\*)malloc(8\*sizeof(char\*)\*N);

float \*\*studentGrade = (float\*\*)malloc(8\*sizeof(float\*)\*N);

int i, j;

//输入数据

printf("Input 5 classes' name (single letters only):\n");

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

{

scanf("%c", className + i);

getchar();

}

printf("Please input each student's name followed by \nhis (or her) scores of classes:\n");

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

{

\*(studentName + j) = (char\*)malloc(8 \* sizeof(char) \* 100);

\*(studentGrade + j) = (float\*)malloc(8 \* sizeof(float)\*M);

scanf("%s", \*(studentName + j));

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

scanf("%f", \*(studentGrade + j) + i);

}

//计算每个学生平均数并输出

float sum;

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

{

sum = 0;

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

sum += \*(\*(studentGrade + j) + i);

printf("Average score of %s is %.2f\n", \*(studentName + j), sum / M);

}

//计算每个课程的平均数并输出

float \*ave = (float\*)malloc(8 \* sizeof(float)\*M);

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

{

sum = 0;

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

sum += \*(\*(studentGrade + j) + i);

\*(ave + i) = sum / M;

printf("Average score of %c is %.2f\n", \*(className + i), sum / M);

}

//统计低于全班各课程成绩平均的人数

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

{

int num = 0;

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

if (\*(ave + i) > \*(\*(studentGrade + j) + i))

num++;

printf("Number of students lower than avg of %c is %d\n", \*(className + i), num);

}

//统计全班不及格人数

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

{

int num = 0;

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

if (\*(\*(studentGrade + j) + i) < 60.0)

num++;

printf("Number of students fail %c is %d\n", \*(className + i), num);

}

//统计全班90分以上人数

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

{

int num = 0;

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

if (\*(\*(studentGrade + j) + i) > 90.0)

num++;

printf("Number of students %c perfect is %d\n", \*(className + i), num);

}

return 0;

}

**【测试数据及测试理论结果】**

本实验情况较为单一，仅设计一组数据即可，设计师尽量加大数据的混乱度。因此，设计以下输入：

1、首先输入课程名：A B C D E；

2、然后输入每个学生的姓名（为一个单词，不含空格）和成绩：

Qw

78 94 58.5 74.8 91.4

Er

59.9 88.3 66 76 92.3

Ty

77 56.9 74 88.1 81.3

Ui

65 85.4 70 79.4 88.7

Op

70 89.4 85.5 79.3 87.5

预测输出没人平均数如下：

Average score of Qw is 79.34

Average score of Er is 76.50

Average score of Ty is 75.46

Average score of Ui is 77.70

Average score of Op is 82.34

预测输出课程平均分如下：

Average score of A is 69.98

Average score of B is 82.80

Average score of C is 70.80

Average score of D is 79.52

Average score of E is 88.24

预测输出统计低于平均分人数如下：

Number of students lower than avg of A is 2

Number of students lower than avg of B is 1

Number of students lower than avg of C is 3

Number of students lower than avg of D is 4

Number of students lower than avg of E is 2

预测输出不及格人数如下：

Number of students fail A is 1

Number of students fail B is 1

Number of students fail C is 1

Number of students fail D is 0

Number of students fail E is 0

预测输出90分以上人数如下：

Number of students A perfect is 0

Number of students B perfect is 1

Number of students C perfect is 0

Number of students D perfect is 0

Number of students E perfect is 2

**实际输出如图6-15、图6-16所示，完全符合预测。**

**【测试结果】**

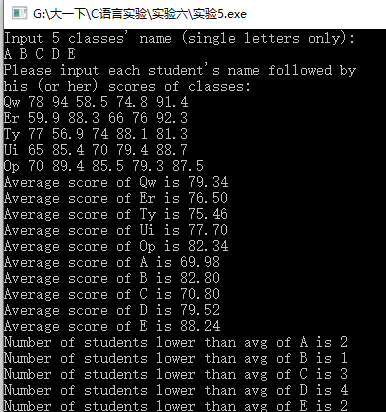


图6-15 实验六6.2.7测试结果（1）

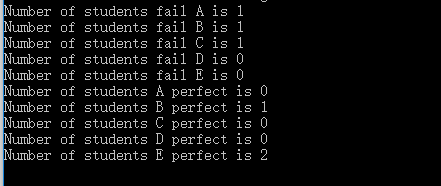


图6-16 实验六6.2.7测试结果（2）

## 6.3 选做题

### 6.3.1 小数相加

**【题目】**

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。

**【算法流程图】**

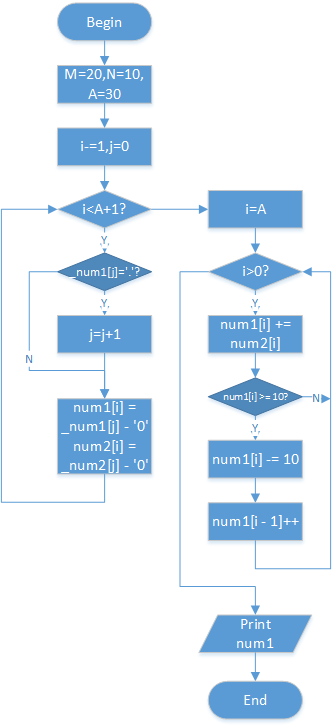


图6-17 实验六 选做题6.3.1算法流程图

**【程序清单】**

#include<stdio.h>

#include<string.h>

#define M 20

#define N 10

#define A (M+N)

int main()

{

char \_num1[A + 2], \_num2[A + 2];

printf("Please input two numbers with (20nums).(10nums):\n");

scanf("%s", \_num1);

scanf("%s", \_num2);

int num1[A + 1] = { 0 }, num2[A + 1] = { 0 };

int i, j, flag = 0;

for (i = 1, j = 0; i < A + 1; i++, j++)

{

if (\_num1[j] == '.') j++;

num1[i] = \_num1[j] - '0';

num2[i] = \_num2[j] - '0';

}

for (i = A; i > 0; i--)

{

num1[i] += num2[i];

if (num1[i] >= 10)

{

num1[i] -= 10;

num1[i - 1]++;

}

}

if (num1[0] != 0)

printf("%d", num1[0]);

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

{

if (i == M + 1)

printf(".");

printf("%d", num1[i]);

}

printf("\n");

return 0;

}

**【测试数据、测试理论结果及测试结果】**

根据首位为0、末位为0以及引发进位设计如下三组数据。

表6-2 实验六 选做题6.3.1测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 011111111101111111111.0111111111  +  01111111110111111111.0111111111 | 0222222222  0222222222  .0222222222 | 如图6-18 |
| （2） | 12345678901234567890.1234567890  +  12345678901234567890.1234567890 | 2469135780  2469135780  .2469135780 | 如图6-19 |
| （3） | 65874654712647846587.2168745612  +  84658723547565358465.5622148754 | 15053337826  0213205052  .7790894366 | 如图6-20 |

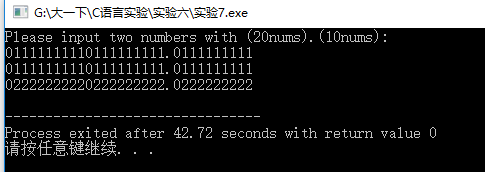


图6-18 实验六 选做题6.3.1样例（1）测试结果

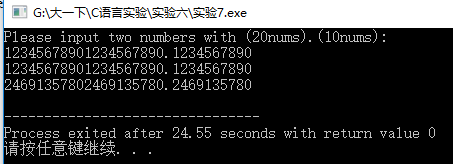


图6-19 实验六 选做题6.3.1样例（2）测试结果

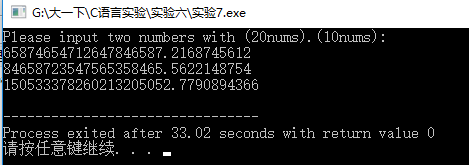


图6-20 实验六 选做题6.3.1样例（3）测试结果

### 6.3.2 复杂声明

**【题目】**

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。

**【算法】**

其思想是跟源程序修改替换中程序修改替换的思想一样，不过就是这个是用的是一个复杂声明，声明了一个存放函数指针的数组，并且将两个字符函数的指针赋给它，相当于给这个函数起了一个别的名称来执行功能，输入两个字符串，按照输入条件来判断用其中一个函数指针，即执行被指函数的功能。

**【程序清单】**

#include<stdio.h>

#include<string.h>

char \*\_strcat(const char \*, const char \*);

char \*\_strcpy(const char \*, const char \*);

char c[100];

int main()

{

int x=1;

char s[162],a[81],\*q=s;

char \*(\*p[2])(const char\*,const char \*);

p[0] = \_strcat;

p[1] = \_strcpy;

printf("输入两行字符串:\n");

gets(s);

gets(a);

printf("输入你想要的指令:\n");

printf("1-将两个字符串连接并输出\n");

printf("2-用第二行字符串覆盖第一行字符串并输出\n0-退出\n");

while(x!=0)

{

scanf("%d",&x);

if(x==1){

q=p[0](s,a);

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

}

else if(x==2){

q=p[1](s,a);

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

}

}

printf("\nPress any key to quit...");

getchar();

return 0;

}

char \*\_strcat(const char \*a, const char \*b)

{

strcpy(c, a);

strcat(c, b);

return c;

}

char \*\_strcpy(const char \*a, const char \*b)

{

strcpy(c, a);

return strcpy(c, b);

}

**【测试数据、测试理论结果及测试结果】**

本程序测试时，设计了如下三组样例，

第一组，普通样例，Hello world；！； 输出结果预测，选择命令1的结果为Hello world！选择命令2的输出结果为！；实际输出结果如图6-21。

第二组，第一行为Hello world，第二个字符串选择空行；输出结果预测，选择命令1的结果为Hello world，选择命令2的结果为空行；实际输出结果如图6-22。

第三组，第一行为 helloworld，第二行为！ ！ ！，在串头加了空格；输出结果预测，选择命令1的结果为 helloworld！ ！ ！；选择命令2的结果为！ ！ ！；实际输出结果如图6-23。

实际输出如图，完全符合预测结果。

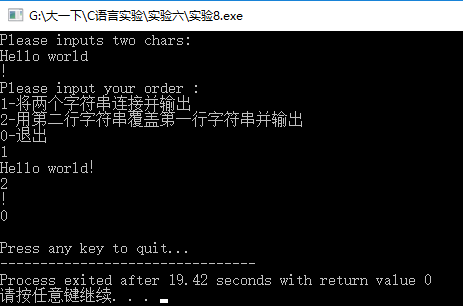


图6-21 实验六选做题6.3.2第一组测试结果

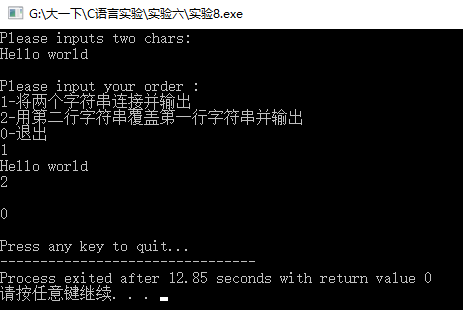


图6-22 实验六选做题6.3.2第二组测试结果

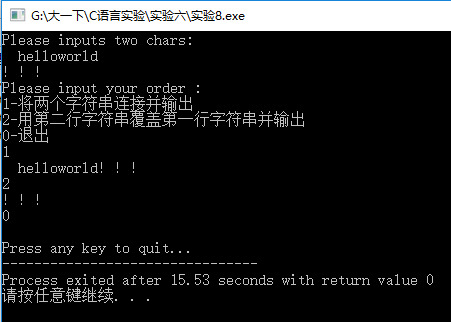


图6-23 实验六选做题6.3.2第三组测试结果

## 6.4 自设题

**6.4.1 验证数组**

**【题目】**

编写一个程序来验证p是不是数组。

**【原理】**

如果p是数组，则p的值和&p的值是相同的；

p++的值会因为p是数组而发生改变。

**【程序清单】**

#include<stdio.h>

void sub(int p[])

{

printf("the size of space p:%d\n",sizeof(p));

printf("the result of p:%p\n",p);

printf("the address of p:%p\n",&p);

p++;

printf("the result of p++:%p\n",p);

}

int main()

{

int a[10]={1,3,5,2,7,9,6,8,0,4};

printf("the size of space a:%d\n",sizeof(a));

printf("the result of a:%p\n",a);

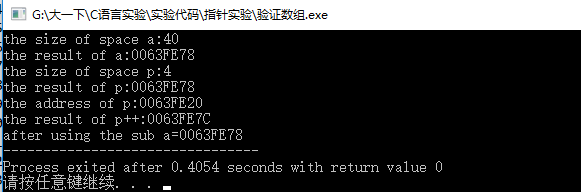
sub(a);

printf("after using the sub a=%p",a);

return 0;

}

**【输出结果】**



**【自设小结】**

通过完成这个自设题，对数组和指针有了深刻的理解，包括数组的首地址与数组名的关系。

## 6.5 小结

源程序改错中，依赖于平时良好的输入习惯，在输入过程中已经可以检查出大多数语法错误，或者拼写错误。接下来再检查一下逻辑是否有误，确认最终无误即可。

源程序修改与替换题锻炼我们将充分利用指针的能力，同时要注意指针的含义，指针指向对象的含义，充分理解防止出现问题，也为以后多重引用奠定基础。

程序设计题则是自主的设计实践，根据题目要求自主设计指针，增强对于所学知识的熟练度和理解程度，锻炼编写代码的能力以及并通过问题的进一步引申锻炼思维的灵活性。

# 7结构与联合实验

## 7.1 实验目的

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。在实验过程中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。了解字段结构和联合的用法。

## 7.2 必做题

### 7.2.1 表达式求值

**【题目】**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

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

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 表达式 | 计算值 | 验证值 |
| 1 | (++p)->x |  |  |
| 2 | p++,p->c |  |  |
| 3 | \*p++->t,\*p->t |  |  |
| 4 | \*(++p)->t |  |  |
| 5 | \*++p->t |  |  |
| 6 | ++\*p->t |  |  |

**【计算值】**

表7-1 实验七必做题7.2.1计算值表

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

**【程序代码】**

#include<stdio.h>

int main()

{

char u[] = "UVWXYZ";

char v[] = "xyz";

struct T {

int x;

char c;

char \*t;

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

printf("%d\n", ((++p)->x)); p = a;

printf("%c\n", (p++, p->c)); p = a;

printf("%c\n", (\*p++->t, \*p->t)); p = a;

printf("%c\n", (\*(++p)->t)); p = a;

printf("%c\n", (\*++p->t)); p = a; p->t--;

printf("%c\n", (++\*p->t));

return 0;

}

**【验证值】**

表7-2 实验七必做题7.2.1验证值表

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

**程序运行图如下图。**

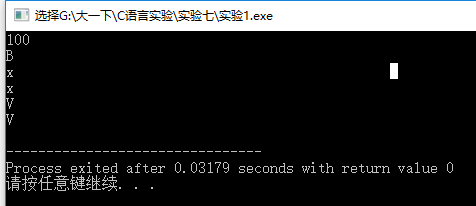


图7-1 实验七必做题7.2.1程序运行图

### 7.2.2 源程序修改替换

**【题目】**

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

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

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

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

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

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

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

void create\_list(struct s\_list \*headp,int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

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

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

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

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

（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。

**【错误分析及修正】**

由于C语言使用值传递的方式，故在函数中对形参headp赋值并不能改变其实参的大小，因此该参数没有必要，以返回值的形式返回地址。所以参数类型不应该是struct s\_list \*headp而是struct s\_list \*\*headp，同样的在引用和赋值的时候也需要相应的改变类型。

**【修改后程序】**

(1)

#include<stdio.h>

#include<stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

void creat\_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};

creat\_list(&head, s);

p = head;

while(p) {

printf("%d\t", p -> data);

p = p -> next; }

printf("\n");

return 0;

}

void creat\_list(struct s\_list \*\*headp, int \*p)

{

struct s\_list \*loc\_head = NULL, \*tail;

if(p[0] == 0)

;

else{

loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head -> data = \*p++;

tail = loc\_head;

while(\*p) {

tail -> next = (struct s\_list \*)malloc(sizeof(struct s\_list));

tail = tail -> next;

tail -> data = \*p++; }

tail -> next = NULL; }

\*headp = loc\_head;

}

(2)

#include <stdio.h>

#include <stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

void creat\_list(struct s\_list \*\*headp,int \*p);

int main()

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0};

creat\_list(&head,s);

p=head;

while(p){

printf("%d\t",p->data);

p=p->next; }

printf("\n");

return 0;

}

void creat\_list(struct s\_list \*\*headp,int \*p)

{

struct s\_list\*loc\_head=NULL,\*tail;

struct s\_list\*temp;

if(p[0]==0 ;

else {

loc\_head=(struct s\_list\*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++;

tail=loc\_head;

while(\*p){

temp=(struct s\_list\*)malloc(sizeof(struct s\_list));

temp->next=loc\_head;

loc\_head=temp;

loc\_head->data = \*p++; }

tail->next=NULL; }

\*headp=loc\_head;

}

**【运行结果】**

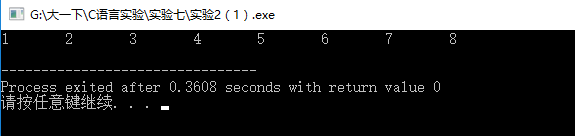


图7-2 实验七必做题7.2.2（1）运行结果

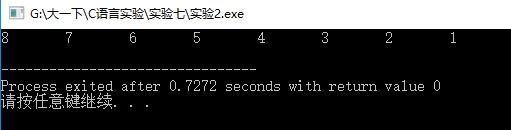


图7-3 实验七必做题7.2.2（2）运行结果

### 7.2.3 字段函数

**【题目】**

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,…,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);

}

**【算法流程图】**

本程序为简单的顺序执行。依照题意设置字段结构和函数，优先级可以通过函数出现位置来实现。其他不变，照常输入。

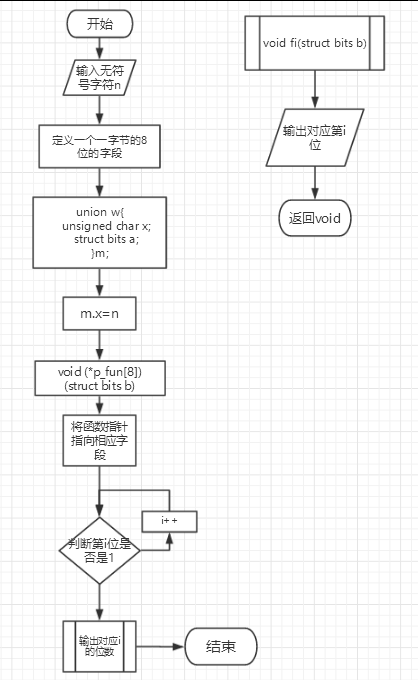


图7-4 实验七必做题7.2.3算法流程图

**【程序清单】**

#include<stdio.h>

struct bits{

unsigned int bit0:1,bit1:1, bit2:1, bit3:1, bit4:1, bit5:1, bit6:1, bit7:1;

};

union w{

unsigned char x;

struct bits a;

}m;

void f0(struct bits b)

{

printf("the function 1 is called! %d\n",b.bit0);

}

void f1(struct bits b)

{

printf("the function 2 is called! %d\n",b.bit1);

}

void f2(struct bits b)

{

printf("the function 3 is called! %d\n",b.bit2);

}

void f3(struct bits b)

{

printf("the function 4 is called! %d\n",b.bit3);

}

void f4(struct bits b)

{

printf("the function 5 is called! %d\n",b.bit4);

}

void f5(struct bits b)

{

printf("the function 6 is called! %d\n",b.bit5);

}

void f6(struct bits b)

{

printf("the function 7 is called! %d\n",b.bit6);

}

void f7(struct bits b)

{

printf("the function 8 is called! %d\n",b.bit7);

}

int main()

{

void (\*p\_fun[8])(struct bits b);

unsigned char n;

printf("Pleas enter a char:");

scanf("%c",&n);

m.x=n;//将其存进联合中

p\_fun[0]=f0;//分别使函数指针指向不同字段

p\_fun[1]=f1;

p\_fun[2]=f2;

p\_fun[3]=f3;

p\_fun[4]=f4;

p\_fun[5]=f5;

p\_fun[6]=f6;

p\_fun[7]=f7;

if(m.a.bit0)

p\_fun[0](m.a);

if(m.a.bit1)

p\_fun[1](m.a);

if(m.a.bit2)

p\_fun[2](m.a);

if(m.a.bit3)

p\_fun[3](m.a);

if(m.a.bit4)

p\_fun[4](m.a);

if(m.a.bit5)

p\_fun[5](m.a);

if(m.a.bit6)

p\_fun[6](m.a);

if(m.a.bit7)

p\_fun[7](m.a);

printf("\nPress any key to quit...");

getch();

return 0;

}

**【测试数据、测试理论结果及测试结果】**

根据输入的字符不同种类设计如下三组样例，理论结果与实际结果一致。

1.输入a；

预测输出为

the function 1 is called! 1

the function 6 is called! 1

the function 7 is called! 1

2.输入4；

预测输出为

the function 3 is called! 1

the function 5 is called! 1

the function 6 is called! 1

3.输入？；

预测输出为

the function 1 is called! 1

the function 2 is called! 1

the function 6 is called! 1

the function 8 is called! 1

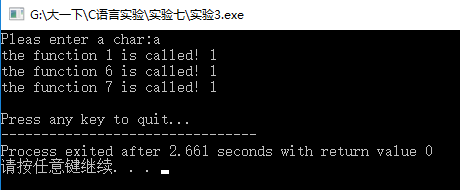


图7-5 实验七必做题7.2.3样例1测试结果

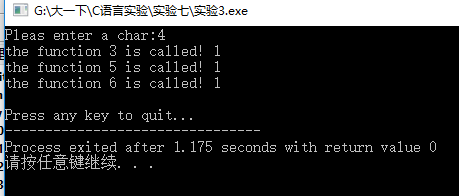


图7-6 实验七必做题7.2.3样例2测试结果

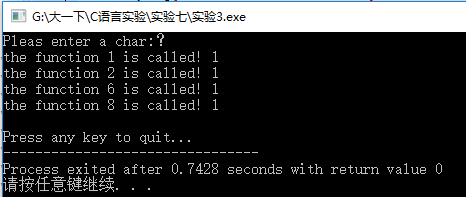


图7-7 实验七必做题7.2.3样例3测试结果

### 7.2.4 成绩统计

**【题目】**

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。

**【算法流程图】**

首先创建一个链表，并将数据输入链表，然后遍历链表输出。之后输入做的更改，遍历链表寻找对应数据，并修改它的值。之后遍历链表算出和，与之前所记录下的总数据数相比得到平均数。最后通过选择排序进行排序后输出。如图所示。

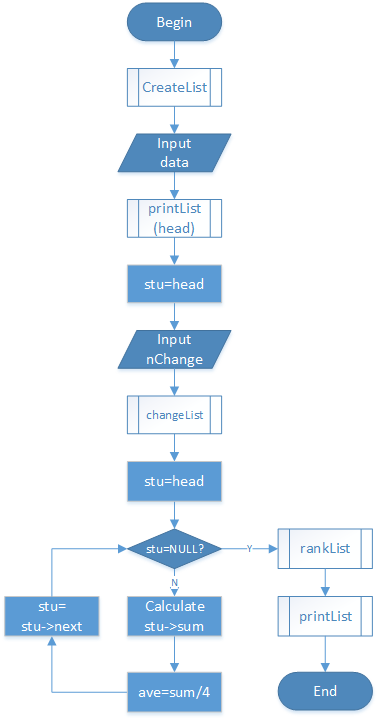


图7-8 实验七 必做题7.2.4算法流程图

**【程序清单】**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int flag = 0;

typedef struct \_Stu {

char stuNum[20];

char name[30];

float engGrade;

float mathGrade;

float phyGrade;

float CGrade;

float ave;

float sum;

struct \_Stu \*next;

}Stu;

Stu \*createList(int);

void printList(Stu \*stu);

int main()

{

int i, n, nChange;

scanf("%d", &n);

getchar();

Stu \*stu = createList(n);

Stu \*head = stu;

char waste[4];

//input the list

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

{

scanf("%s%s", stu->stuNum, stu->name);

scanf("%f", &stu->engGrade);

scanf("%f", &stu->mathGrade);

scanf("%f", &stu->phyGrade);

scanf("%f", &stu->CGrade);

fgets(waste, 4, stdin);

stu = stu->next;

}

//print out the list

printList(head);

//change the partiicular contents. Take change the C grade of Harry for example

stu = head;//initiate the pointer stu

scanf("%d", &nChange);

char \*lesson[4] = { "英语","数学","物理","C" };

char changeLesson[15], changeNum[20];

float changeScore;

while (nChange-- > 0)

{

stu = head;

scanf("%s %s %f", changeNum, changeLesson, &changeScore);

while (stu)

{

if (strcmp(stu->stuNum, changeNum) == 0)

{

if (strcmp(changeLesson, "英语") == 0)

stu->engGrade = changeScore;

if (strcmp(changeLesson, "数学") == 0)

stu->mathGrade = changeScore;

if (strcmp(changeLesson, "物理") == 0)

stu->phyGrade = changeScore;

if (strcmp(changeLesson, "C") == 0)

stu->CGrade = changeScore;

break;

}

stu = stu->next;

}

}

printf("Alter:\n");

printList(head);

//calciulate the average scores of each student

float sum[4] = { 0 };

for (stu = head; stu != NULL; stu = stu->next)

{

stu->sum = stu->engGrade + stu->CGrade + stu->mathGrade + stu->phyGrade;

stu->ave = stu->sum / 4;

}

//print out the sum and average

printf("总分和平均分:\n");

printf("%-15s%-20s%-10s%-10s\n", "ID", "名字", "总分", "平均分");

stu = head;

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->sum, stu->ave);

stu = stu->next;

}

//sort the number

Stu \*begin, \*end;

for (begin = head; begin != NULL; begin = begin->next)

for (end = begin; end != NULL; end = end->next)

if (end->ave < begin->ave)

{

char tmpNum[20], tmpName[20];

strcpy(tmpNum, end->stuNum);

strcpy(end->stuNum,begin->stuNum);

strcpy(begin->stuNum, tmpNum);

strcpy(tmpName,end->name);

strcpy(end->name,begin->name);

strcpy(begin->name,tmpName);

float tmpScore = end->ave;

end->ave = begin->ave;

begin->ave=tmpScore;

}

printf("\nSort:\n");

printf("%-15s%-20s%-10s\n", "ID", "名字", "平均分");

stu = head;

while (stu)

{

printf("%-15s%-20s%-10.2f\n",

stu->stuNum, stu->name, stu->ave);

stu = stu->next;

}

printf("\n");

return 0;

}

void printList(Stu \*stu)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "名字", "英语", "数学", "物理", "C");

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->engGrade, stu->mathGrade, stu->phyGrade, stu->CGrade);

stu = stu->next;

}

printf("\n");

}

Stu \*createList(int stuNum)

{

static int num;

static Stu \*head;

static Stu \*tail;

if (flag == 0)

{

num = stuNum;

head = (Stu\*)malloc(sizeof(Stu));

tail = head;

flag = 1;

}

if (num-- < 2)

{

tail->next = NULL;

return head;

}

else {

tail->next = (Stu\*)malloc(sizeof(Stu));

tail = tail->next;

return createList(num);

}

}

**【测试数据、测试理论结果及测试结果】**

5

U201614547 Qw 85 86 87 88

U201614848 Er 89 90 91 92

U201615049 Ty 56 85 89 59

U201615252 Ui 84 89 64 100

U201616260 Op 89.5 90 85 74

使用3次测试修改，分别为：

3

U201614547 数学 95.6

U201615049 C 100

U201616260 英语 100

第一次输入后，预计输出为 ：

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 86.00 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 59.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 89.50 90.00 85.00 74.00第二次输入后，预计输出为：

Alter:

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 95.60 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 100.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 100.00 90.00 85.00 74.00

总分和平均分

ID 名字 总分 平均分

U201614547 Qw 355.60 88.90

U201614848 Er 362.00 90.50

U201615049 Ty 330.00 82.50

U201615252 Ui 337.00 84.25

U201616260 Op 349.00 87.25

Sort:

ID 名字 平均分

U201615049 Ty 82.50

U201615252 Ui 84.25

U201616260 Op 87.25

U201614547 Qw 88.90

U201614848 Er 90.50

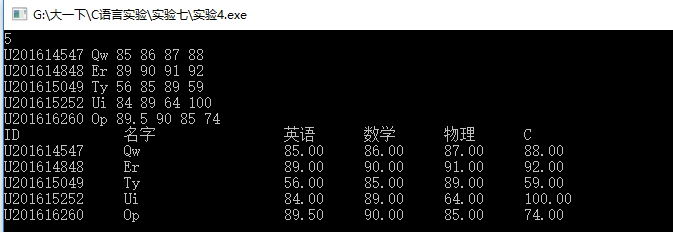


图7-9 实验七必做题7.2.4测试结果（1）

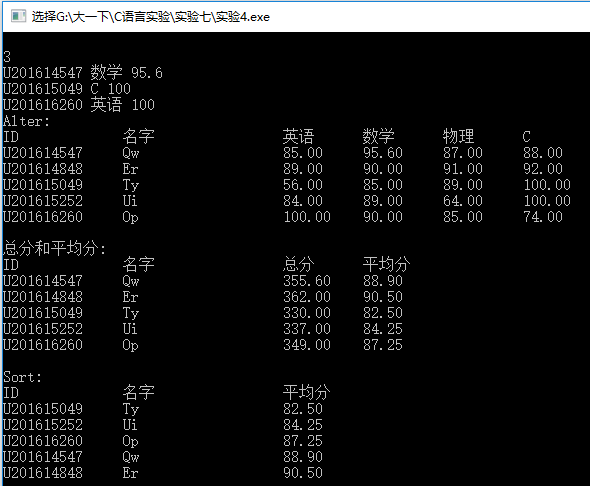


图7-10 实验七必做题7.2.4测试结果（2）

## 7.3 选做题

### 7.3.1 双向链表

**【题目】**

采用双向链表重做编程设计题中的第（2）题。

**【算法流程图】**

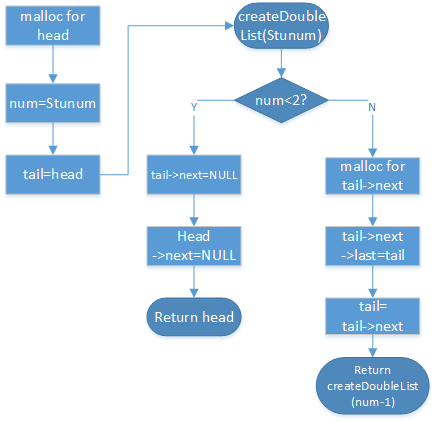


图7-11 实验七选做题双向链表算法流程图

**【程序清单】**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct \_Stu {

char stuNum[20];

char name[30];

float engGrade;

float mathGrade;

float phyGrade;

float CGrade;

float ave;

float sum;

struct \_Stu \*next;

struct \_Stu \*last;

}Stu;

Stu \*createDoubleList(int);

void printList(Stu \*stu);

int main()

{

int i, n, nChange;

scanf("%d", &n);

getchar();

Stu \*stu = createDoubleList(n);

Stu \*head = stu, \*rst;

char waste[4];

//input the list

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

{

scanf("%s%s", stu->stuNum, stu->name);

scanf("%f", &stu->engGrade);

scanf("%f", &stu->mathGrade);

scanf("%f", &stu->phyGrade);

scanf("%f", &stu->CGrade);

fgets(waste, 4, stdin);

stu = stu->next;

}

printList(head);

//change the partiicular contents. Take change the C grade of Harry for example

stu = head;//initiate the pointer stu

scanf("%d", &nChange);

char \*lesson[4] = { "English","Math","Physics","C" };

char changeLesson[15], changeNum[20];

float changeScore;

while (nChange-- > 0)

{

stu = head;

scanf("%s %s %f", changeNum, changeLesson, &changeScore);

while (stu)

{

if (strcmp(stu->stuNum, changeNum) == 0)

{

if (strcmp(changeLesson, "English") == 0)

stu->engGrade = changeScore;

if (strcmp(changeLesson, "Math") == 0)

stu->mathGrade = changeScore;

if (strcmp(changeLesson, "Physics") == 0)

stu->phyGrade = changeScore;

if (strcmp(changeLesson, "C") == 0)

stu->CGrade = changeScore;

break;

}

stu = stu->next;

}

}

printf("Alter:\n");

printList(head);

//calculate the average scores of each student

float sum[4] = { 0 };

for (stu = head; stu != NULL; stu = stu->next)

{

stu->sum = stu->engGrade + stu->CGrade + stu->mathGrade + stu->phyGrade;

stu->ave = stu->sum / 4;

}

//print out the sum and average

printf("SumAndAvg:\n");

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

stu = head;

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->sum, stu->ave);

stu = stu->next;

}

return 0;

}

//to print out the list

void printList(Stu \*stu)

{

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

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->engGrade, stu->mathGrade, stu->phyGrade, stu->CGrade);

stu = stu->next;

}

printf("\n");

}

Stu \*createDoubleList(int stuNum)

{

static Stu \*head= (Stu\*)malloc(sizeof(Stu));

static int num = stuNum;

static Stu \*tail = head;

if (num-- < 2)

{

tail->next = NULL;

head->last = NULL;

return head;

}

else

{

tail->next = (Stu\*)malloc(sizeof(Stu));

tail->next->last = tail;

tail = tail->next;

return createDoubleList(num);

}

}

**【测试数据、测试理论结果及测试结果】**

5

U201614547 Qw 85 86 87 88

U201614848 Er 89 90 91 92

U201615049 Ty 56 85 89 59

U201615252 Ui 84 89 64 100

U201616260 Op 89.5 90 85 74

使用3次测试修改，分别为：

3

U201614547 数学 95.6

U201615049 C 100

U201616260 英语 100

第一次输入后，预计输出为 ：

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 86.00 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 59.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 89.50 90.00 85.00 74.00第二次输入后，预计输出为：

Alter:

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 95.60 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 100.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 100.00 90.00 85.00 74.00

总分和平均分

ID 名字 总分 平均分

U201614547 Qw 355.60 88.90

U201614848 Er 362.00 90.50

U201615049 Ty 330.00 82.50

U201615252 Ui 337.00 84.25

U201616260 Op 349.00 87.25

Sort:

ID 名字 平均分

U201615049 Ty 82.50

U201615252 Ui 84.25

U201616260 Op 87.25

U201614547 Qw 88.90

U201614848 Er 90.50

实际输出如图7-12，完全符合预测。

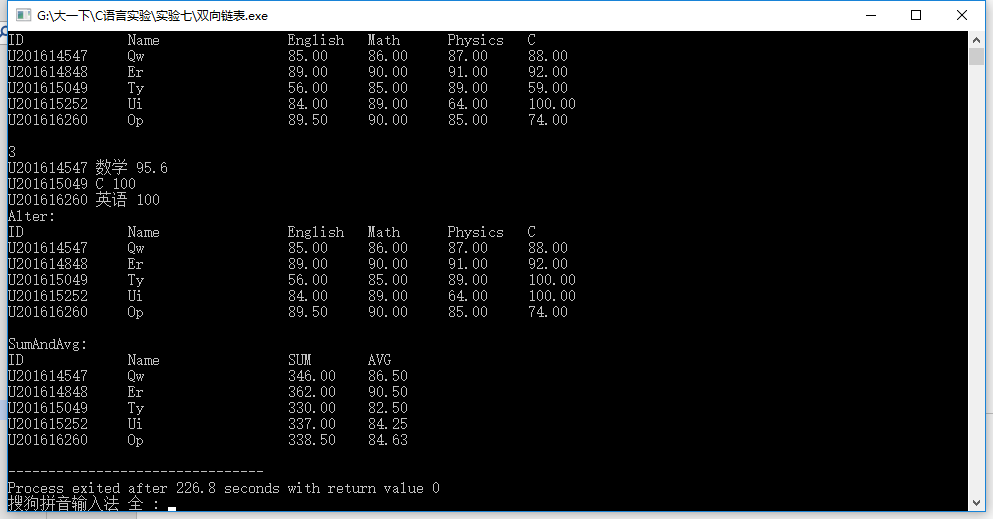


图7-12 实验七选做题双向链表测试结果

## 7.4 小结

通过表达式求值进一步加深了对指针与结构体的理解，可以更熟练地计算指针与结构体的表达式；通过源程序修改替换进一步理解了性参与实参的区别；程序设计第二题，则是学生信息录入表，并用链表来记录。而链表就是结构的一种比较普遍的应用，通过这个题目我对创建链表更为熟悉，也对链表的输入输出有了更深的印象，所以通过成绩统计以及之后的构建单项链表、构建双向链表进一步熟悉了关于链表数据结构的有关操作；通过查阅资料学习了宏定义中#与##运算符的作用，并尝试用其简化代码，取得了一定效果。

# 8文件实验

## 8.1实验目的

熟悉文本文件和二进制文件在磁盘中的存储方式；掌握流式文件的读写方法。

## 8.2 必做题

### 8.2.1 文件程序验证

**【题目】**

设有程序：

#include <stdio.h>

int main(void)

{

short a=0x253f,b=0x7b7d;

char ch;

FILE \*fp1,\*fp2;

fp1=fopen("d:\\abc1.bin","wb+");

fp2=fopen("d:\\abc2.txt","w+");

fwrite(&a,sizeof(short),1,fp1);

fwrite(&b,sizeof(short),1,fp1);

fprintf(fp2,"%hx %hx",a,b);

rewind(fp1); rewind(fp2);

while((ch = fgetc(fp1)) != EOF)

putchar(ch);

putchar('\n');

while((ch = fgetc(fp2)) != EOF)

putchar(ch);

putchar('\n');

fclose(fp1);

fclose(fp2);

return 0;

}

（1）请思考程序的输出结果，然后通过上机运行来加以验证。

（2）将两处sizeof(short)均改为sizeof(char)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)结果有什么不同。

**【答案】**

（1）预测程序运行输出结果-分别为？%} { 和253f 7b7d。

实际输出结果如下图8-1、图8-2和图8-3所示。与预测结果一致。

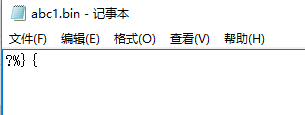


图8-1文件输出结果

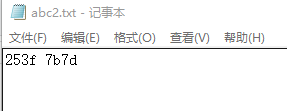


图8-1文件输出结果

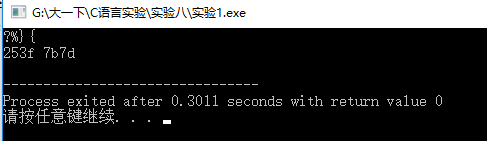


图8-3 控制台输出结果

（2）第一个更改为sizeof(char)后，读入时只读入一个大小为char的数据，从而导致输出的长度减半，即为?}；而第二个更改为sizeof(char)后，因为后面会被fprintf覆盖，所以不会造成影响。输出结果如下图8-4、图8-5所示。

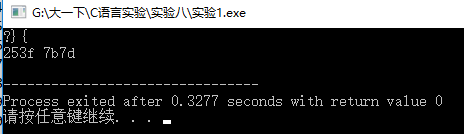


图8-4 实验八必做题8.2.1（2）修改第一个后输出结果

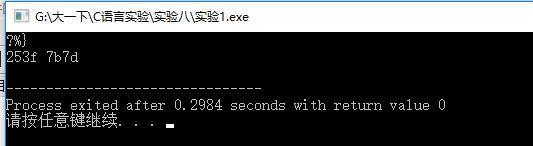


图8-5 实验八必做题8.2.1（2）修改第二个后输出结果

（3）改为%d后，将以十进制形式输出a，b，预测结果为9535 31613。实际结果如下图8-4.

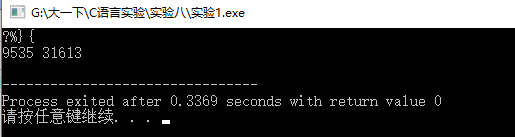


图8-6 实验八必做题8.2.1（3）输出结果

### 8.2.2源程序修改替换

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：type filename

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

1. #include<stdio.h>
2. #include<stdlib.h>
3. int main(int argc, char\* argv[])
4. {
5. char ch;
6. FILE \*fp;
7. if(argc!=2){
8. printf("Arguments error!\n");
9. exit(-1);
10. }
11. if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/
12. printf("Can't open %s file!\n",argv[1]);
13. exit(-1);
14. }
15. while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/
16. putchar(ch); /\* 向显示器中写字符 \*/
17. fclose(fp); /\* 关闭filename \*/
18. return 0;
19. }

（2）用输入输出重定向freopen改写上述源程序中的main函数。

**【错误分析及修改】**

1.命令行参数中的argv[0]被文件名占用，所以输入的type应该为argv[1]，并且总参数应该是3个。应该将if(argc!=2)改为if (argc != 3)，同时将判断filename均改为argv[2]。

2.读入字符时没有考虑到=与!=的优先级。应在ch=fgetc(fp)外加（）来保证运算优先级。

3、源程序存在一个缺陷。由于仅仅判断是否有两个参数，故输入t filename的时候也会正常执行。修改这个缺陷只要加入判断参数argv[1]是否为type的代码。即为：

if (strcmp(argv[1], "type") != 0)

{

printf("%s is not a right command", argv[1]);

exit(-1);

}

**【修改后代码】**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

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

{

char ch;

FILE \*fp;

if (argc != 3) {

printf("Arguments error!\n");

exit(-1);

}

if (strcmp(argv[1], "type") != 0)

{

printf("%s is not a right command", argv[1]);

exit(-1);

}

if ((fp = fopen(argv[2], "r")) == NULL) { /\* fp 指向 filename \*/

printf("Can't open %s file!\n", argv[2]);

exit(-1);

}

while ((ch = fgetc(fp)) != EOF) /\* 从filename中读字符 \*/

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

### 8.2.3 大小写转化

**【题目】**

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。

**【程序清单】**

#include<stdio.h>

enum \_state { SPACE, COPY };

void change(char str[]);

int main()

{

FILE \*fp1;

char str[100];

fp1 = fopen("D:\\m.txt", "w+");

fgets(str, 100, stdin);

change(str);

fprintf(fp1, "%s", str);

fclose(fp1);

return 0;

}

void change(char str[])

{

int i = 0;

enum \_state state;

if(str[0]==' ') state = SPACE;

else {

str[0] ^= 32;

state = COPY;

}

for (i = 1; str[i] != '\0'; i++)

{

switch (state)

{

case SPACE:

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

str[i] ^= 32;

if (str[i] == ' ')

state = SPACE;

else state = COPY;

break;

case COPY:

if (str[i] == ' ')

state = SPACE;

else state = COPY;

break;

}

}

}

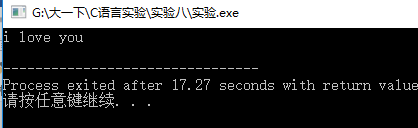
**【测试样例、测试理论结果及测试结果】**

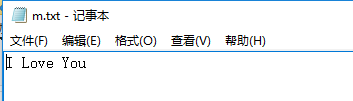
1、在控制台输入“i love you.”，预计在文件中输出“I Love You.”；

2、在控制台输入“ you love me, aren’t you?”，预计在文件中输出“You Love Me, Aren’t You?”；

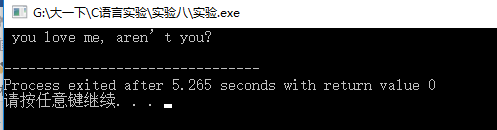
3、在控制台输入“ i love you and the world. Haha!”，空格保留，大写保留，预计在文件中输出“ I Iove You And The World. Haha!”；

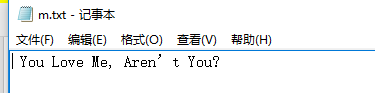
实际输出分别如图8-，图8.5-2，图8.5-3所示，完全符合预测**。**



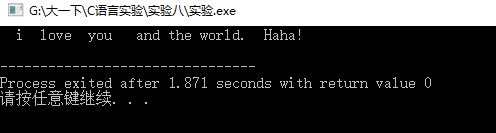


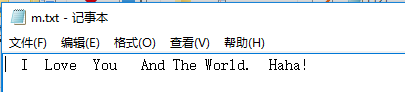
实验八必做题8.2.3样例1控制台输入及文件输出





实验八必做题8.2.3样例2控制台输入及文件输出





实验八必做题8.2.3样例3控制台输入及文件输出

## 8.3小结

通过完成这几个文件实验，我对文件的一些基础函数有了较好的掌握，同时真正理解了文件的读与写。

# 参考文献

[1]曹计昌,卢萍,李开. C语言程序设计,北京：科学出版社,2013

[2]李开,卢萍,曹计昌. C语言实验与课程设计,北京：科学出版社,2013

[3]百度文库 华中科技大学C语言实验与课程设计上机实验报告汇总(1-8全部) <http://wenku.baidu.com/view/22fd62d6312b3169a451a4b.0html>

[4]王金鹏.C语言可以这样学，北京：清华大学出版社，2016