

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期： 2021年1月12日**

**网络空间安全学院**

**目 录**

[1 表达式和标准输入与输出实验 3](#_Toc61357483)

[1.1 实验目的 3](#_Toc61357484)

[1.2 实验内容 3](#_Toc61357485)

[1.3 实验小结 16](#_Toc61357486)

[2 流程控制实验 17](#_Toc61357487)

[2.1 实验目的 17](#_Toc61357488)

[2.2 实验内容 17](#_Toc61357489)

[2.3 实验小结 35](#_Toc61357490)

[3 函数与程序结构实验 36](#_Toc61357491)

[3.1 实验目的 36](#_Toc61357492)

[3.2 实验内容 36](#_Toc61357493)

[3.3 实验小结 51](#_Toc61357494)

[4 编译预处理实验 53](#_Toc61357495)

[4.1 实验目的 53](#_Toc61357496)

[4.2 实验内容 53](#_Toc61357497)

[4.3 实验小结 66](#_Toc61357498)

[5 数组实验 67](#_Toc61357499)

[5.1 实验目的 67](#_Toc61357500)

[5.2 实验内容 67](#_Toc61357501)

[5.3 实验小结 82](#_Toc61357502)

[6 指针实验 83](#_Toc61357503)

[6.1 实验目的 83](#_Toc61357504)

[6.2 实验题目及要求 83](#_Toc61357505)

[6.3 实验小结 108](#_Toc61357506)

[7 结构与联合实验 109](#_Toc61357507)

[7.1 实验目的 109](#_Toc61357508)

[7.2 实验内容 109](#_Toc61357509)

[1.3 实验小结 128](#_Toc61357510)

[8 文件操作实验 129](#_Toc61357511)

[8.1 实验目的 129](#_Toc61357512)

[8.2 实验内容 129](#_Toc61357513)

[8.3 实验小结 137](#_Toc61357514)

[9 实验总结 139](#_Toc61357515)

[9.1 遇到的问题与收获 139](#_Toc61357516)

[9.1.1 C语言学习中遇到的问题 139](#_Toc61357517)

[9.1.2 C语言学习中的收获 139](#_Toc61357518)

[参考文献 140](#_Toc61357519)

# 1 表达式和标准输入与输出实验

## 实验目的

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

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

（3）掌握简单C程序的编写方法。

（4）熟悉C语言程序的开发环境，并学会调试程序的方法。

## 1.2 实验内容

**1.2.1 程序改错与跟踪调试**

下面的实验1-1程序用来完成以下任务：

（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）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，拼成一个新的整数后输出。

在这个程序中存在若干语法和逻辑错误，要求先编译程序改正语法错误，再采用单步执行的方式调试程序找出逻辑错误。在单步执行程序的过程中，观察以下变量值：

（1）执行完c = 5/9 \* (f-32)，c的值为多少？

（2）执行完scanf(“%f”, &r)，r的值为多少？

（3）执行完newint = p&0xff00|k>>8，newint的值是多少？表达式k>>8的值是多少？

根据观察结果分析代码并修改程序，使之能够正确完成指定任务。

/\*实验1-1程序改错与跟踪调试题源程序\*/

1 #include<stdio.h>

2 #define PI 3.14159;

3 int main( void )

4 {

5 int f ;

6 short p, k ;

7 double c, r, s ;

/\* 任务1 \*/

8 printf("Input Fahrenheit: " ) ;

9 scanf("%d", f ) ;

10 c = 5/9 \* (f-32) ;

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

/\* 任务2 \*/

12 printf("input the radius r:");

13 scanf("%f", &r);

14 s = PI \* r \* r;

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

/\* 任务3 \*/

16 k = 0xa1b2, p = 0x8432;

17 newint = p&0xff00|k>>8;

18 printf("newint = %#x\n\n",newint);

19 return 0;

20 }

**解答：**

（1）语法错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.14159

2) 第6行的short p, k ;，未声明变量newint，正确形式为：

short p, k, newint ;

3) 第11行的scanf(“&d”,f );，未指明变量的地址，正确形式为：

scanf(“%d”,&f)

4) 第17行的scanf(“%f”,&r) ;，r为double类型，在scanf()函数中占位符应为1f，正确形式为：

scanf(“%1f”,&r) ;

5) 第19行的printf("\nThe acreage is %.2f\n\n",&s);中，不应使用地址，正确形式为：

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

（2）修改完以上错误后，在单步执行过程中，观察以下变量值：

1. 执行完c = 5/9 \* (f-32)，c的值为：0
2. 执行完scanf(“%f”, &r)，r的值为：0
3. 执行完newint = p&0xff00|k>>8，newint的值是0xffffffa1。表达式k>>8的值是0xa1。

（3）基于以上的单步执行观察结果，说明还存在如下逻辑错误：

1）第5行的c = 5/9 \* (f-32) ;，整数直接相除会损失精度，需要改为浮点型，应该修改成：c = 5.0/9.0 \* (f-32) ;

2）第6行的short p, k ;变量类型错误，应为unsigned short，应该修改成：unsigned short p, k, newint ;

修改后，源程序清单如下：

#include<stdio.h>

#define PI 3.14159

int main( void )

{

int f ;

unsigned short p, k, newint ;

double c, r, s ;

/\* 任务1 \*/

printf("Input Fahrenheit: " ) ;

scanf("%d", &f ) ;

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

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

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

/\* 任务2 \*/

printf("input the radius r:") ;

scanf("%lf", &r) ;

s = PI \* r \* r ;

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

/\* 任务3 \*/

k = 0xa1b2, p = 0x8432 ;

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

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

return 0 ;

}

（4）错误修改后运行结果如图1-1所示。

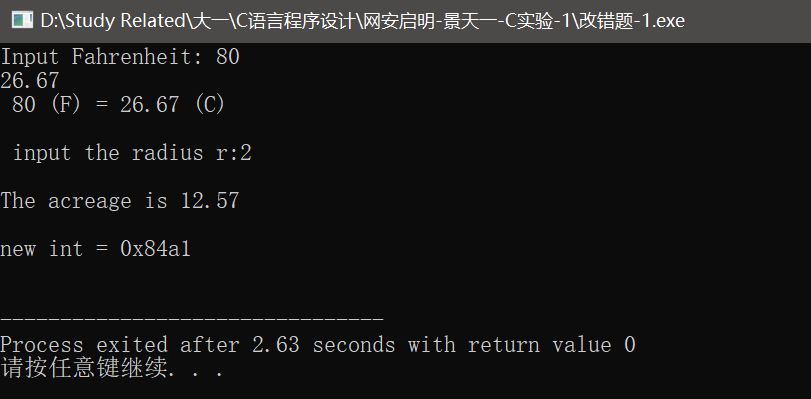
****

图1-1 实验1-1修改后运行结果截图

**1.2.2 源程序修改替换**

下面的实验1-2程序用“更相减损”法求m与n的最大公约数。

/\* 实验1-2程序分析与修改题源程序 \*/

#include <stdio.h>

int main( )

{

int m, n, k, p, i, d;

printf("input m, n \n");

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

if (m<n) /\* 交换m和n \*/

{

int t;

t = m;

m = n;

n = t;

}

k = 0;

while (m%2 == 0 && n%2 == 0) /\* m和n均为偶数 \*/

{

m /= 2; /\* 用2约简m和n \*/

n /= 2;

k++;

}

for (p = 1,i = 0; i<k; i++) p \*= 2; /\* 求p=2k \*/

while((d=m-n)!=n)

{

if(d>n) m = d;

else

{

m = n;

n = d;

}

}

d \*=p;

printf("the greatest common divisor : %d", d);

return 0;

}

1. 分析程序采用的“更相减损”法的算法步骤。

**解答：**

1. 程序采用的“更相减损”法的算法步骤如下：
   1. 输入被除数m和除数n
   2. 若m<n时，互换m, n的值
   3. 当m, n均可被2整除时，同时除以2直到其中一个数不能被2整除；
   4. 求数字p，使得p=2^k，k为上一步约简次数；
   5. 求m-n的值；
   6. 循环直至m-n的值等于除数n，计此时该数字为d；
   7. 最大公约数即为d\*p；
2. 按以下要求对源程序进行优化，提高程序的执行效率。

将交换m和n、判断一个整数是否是偶数、用2约简m和n、求p=2k等操作改成用位运算实现，并且m和n的交换不能够使用中间变量t，需要删除声明t的语句。

1. 将else后的复合语句改用一条表达式语句。
2. 按照要求优化后的程序分别如下：
3. 交换m和n的代码为：

m = m ^ n;

n = n ^ m;

m = m ^ n;

1. 判断一个整数是否是偶数的代码为：

(m&0x0001) == 0 && (n&0x0001) == 0

1. 用2约简m和n的代码为：

m = m >> 1; /\* 用2约简m和n \*/

n = n >> 1;

1. 求p=2k的代码为：

for (p = 1,i = 0; i<k; i++) p = p << 1; /\* 求p=2k \*/

1. else后的复合语句可以修改为：

m = n, m=d;

1. 优化和替换后程序的源代码如下：

#include <stdio.h>

int main(void )

{

int m, n, k, p, i, d;

printf("input m, n \n");

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

if (m<n) /\* 交换m和n \*/

{

m = m ^ n;

n = n ^ m;

m = m ^ n;

}

k = 0;

while ((m&0x0001) == 0 && (n&0x0001) == 0) /\* m和n均为偶数 \*/

{

m = m >> 1; /\* 用2约简m和n \*/

n = n >> 1;

k++;

}

for (p = 1,i = 0; i<k; i++) p = p << 1; /\* 求p=2^k \*/

while((d=m-n)!=n)

{

if(d>n)

m = d;

else

m = n, n = d;

}

d \*=p;

printf("the greatest common divisor : %d", d);

return 0;

}

优化和替换后运行结果如图1-2所示。

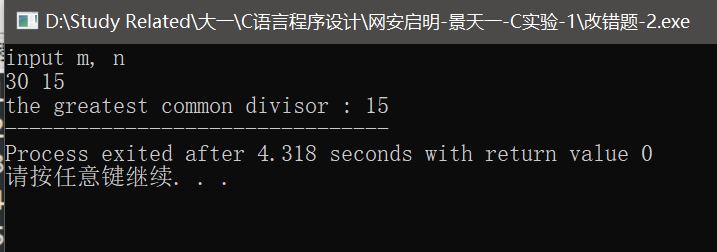


图1-2 实验1-2优化替换后运行结果截图

**1.2.3 程序设计**

（1）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，输入Ctrl+Z程序结束。要求：①用条件表达式；②字符的输入输出用getchar和putchar函数。程序应能循环接受用户的输入，直至输入Ctrl+Z程结束。例如：

A （键盘输入）

a

^Z （键盘输入）

**解答：**

1） 算法流程如图1-3所示。

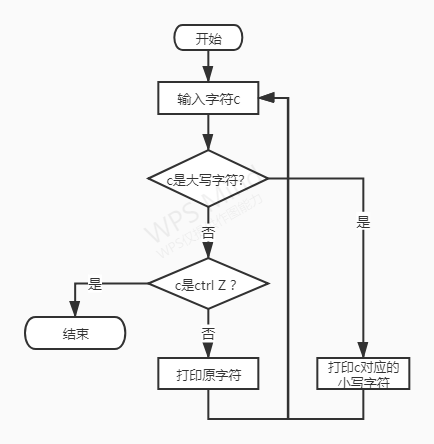


图1-3 程序设计题1的程序流程图

2）源程序清单

#include<stdio.h>

int main(void)

{

while(1)

{

char c;

c=getchar();

if (c>=65 && c<=90)

putchar(c+32);

else if(c==EOF)

break;

else

putchar(c);

}

return 0;

}

3）测试

（a）测试数据：

测试数据如表1-1所示：

表1-1程序题1的测试数据

|  |  |  |
| --- | --- | --- |
| 输入数据 | 预计输出 | 实际输出 |
| B | B | B |
| a | a | a |
| Ctrl Z | 结束 | 结束 |

（b） 对应测试数据的运行结果如图1-4所示。

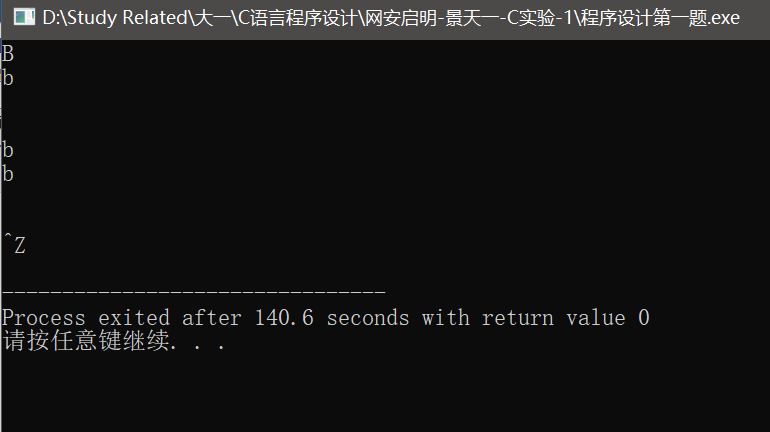


图1-4实验题1-3测试结果如图

输出结果符合预期，说明程序运行正确。

（2）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（x从右至左编号为0～15），并使其向左端（第15位）靠齐。要求：①检查m和n的范围；②x的值以十六进制输入，m和n以十进制输入；③结果以十六进制输出。

**解答：**

1. 解题思路如图1-5所示：

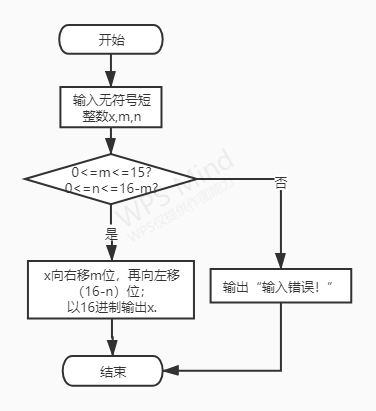


图1-5程序设计题2程序流程图

1. 程序清单

#include<stdio.h>

int main( void )

{

unsigned short x,m,n;

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

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

if (m >= 0 && m <= 15 && n >= 0 && n <= 16-m)

/\*判断m、n的值是否在合理范围内\*/

{

printf("ans=%hx\n",x);

}

else

{

printf("输入错误!\n");

}

return 0;

}

3）测试

（a） 测试数据：

测试数据如表1-3示。

表1-2程序题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 |
| x | m | n |
| 用例1 | 0100 0110 1000 0000（4680） | 7 | 4 | 计算结果1101 0000 0000 0000 即D000 |
| 用例2 | 1101 0101 1000 0011（D(583） | 16 | 1 | 输入错误（m值超范围） |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） |

（b） 对应测试测试用例1的运行结果如图1-4所示。

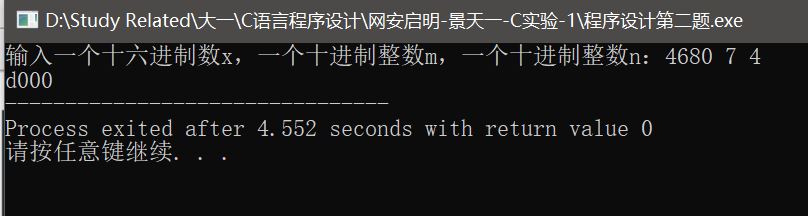


图1-6程序设计题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图1-5所示。

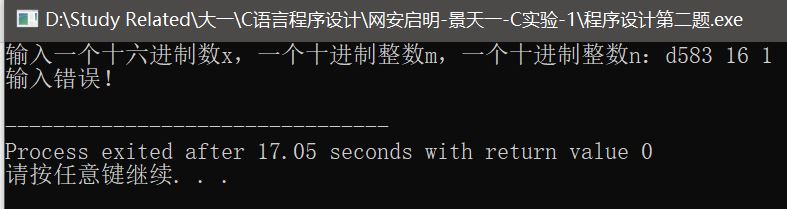


图1-7程序设计题2的测试用例二的运行结果

对应测试测试用例3的运行结果如图1-6所示。

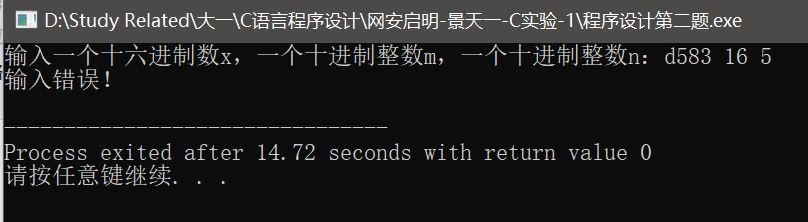


图1-8程序设计题2的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）IP地址通常是4个用句点分隔的小整数（即点分十进制），但这些地址在机器中是用一个无符号长整型数表示的。例如3232235876，其机内二进制表示就是11000000 10101000 00000001 01100100，按照8位一组用点分开，该IP地址就写成192.168.1.100。

读入无符号长整型数表示的互联网IP地址，对其译码，以常见的点分十进制形式输出。要求循环输入和输出，直至输入Ctrl+Z结束。

**解答：**

1. 解题思路：

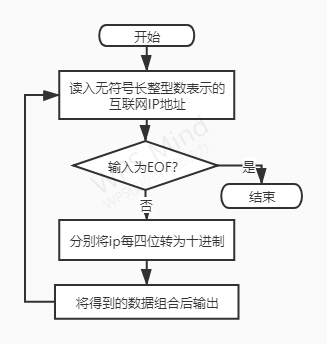
算法流程如图1-9所示。

图1-9程序设计题3程序流程图

1. 程序清单

#include<stdio.h>

int main(void)

{

unsigned long ip,d\_1,d\_2,d\_3,d\_4; //获取数据

while(scanf("%lu",&ip) != EOF) //循环条件

{

d\_1 = (ip & 0xff000000)>>24;

d\_2 = (ip & 0x00ff0000)>>16;

d\_3 = (ip & 0x0000ff00)>>8;

d\_4 = (ip & 0x000000ff);

printf("%d.%d.%d.%d\n",d\_1,d\_2,d\_3,d\_4); //ip地址输出

scanf("%lu",&ip);

}

return 0;

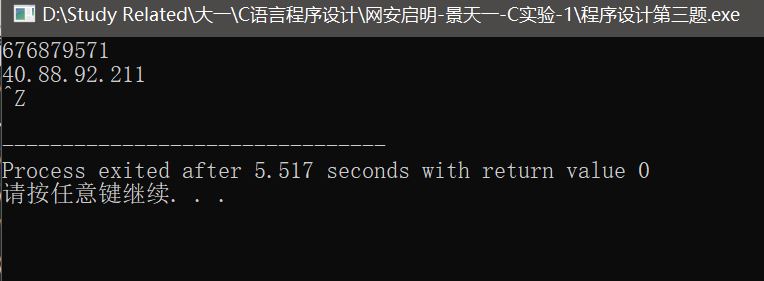
}

1. 测试
   1. 测试数据

测试数据如表1-3所示。

1. 表1-3程序题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 实际输出 |
| 用例1 | 676879571（00101000 01011000 01011100 11010011） | 211.92.88.40 |
| 用例2 | ^Z | 结束 |

 b) 对应测试测试用例1的运行结果如图1-10所示。

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 1.3 实验小结

本次实验主要练习了输入输出函数的用法，在具体编程实验中有重要的意义。遇到的第一个问题是ctrl z的输入，通过查询资料和询问同学解决；第二个问题是在输入ctrl z时终止，通过查阅课本了解到scanf函数的返回值在出错或终止时为EOF，因此设计判断scanf函数返回值的方案来实现。

本次实验最大的体会是，完整实现需求并将其有条理地记录下来是极为重要的，对每一次实验都需要提前预习，完成代码demo，以给后期调试，撰写报告，整理反思留出更多时间。

# 2 流程控制实验

## 2.1 实验目的

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

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 2.2 实验内容

**2.2.1 程序改错**

下面的实验2-1程序是合数判断器（合数指自然数中除了能被1和本身整除外，还能被其它数整除的数），在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够正确完成指定任务。

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF) {
7. for(i=2,k=x>>1;i<=k;i++)
8. if (!x%i) {
9. flag = 1;
10. break;
11. }
12. if(flag=1) printf("%d是合数", x);
13. else printf("%d不是合数", x);
14. }
15. return 0;
16. }

**解答：**

（1）语法错误修改：

1) 第8行的if (!x%i)，运算优先级错误，正确形式为：

if (!(x%i) )

2) 第12行的if(flag=1) printf("%d是合数", x);，判定等于写成了赋值符号，正确形式为：

if(flag==1) printf("%d是合数", x);

（2）逻辑错误修改：

1）一次循环后，没有给flag赋初值，应该在while循环开始时添加语句：

flag = 0 ;

（3）修改后，源程序清单如下：

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF)
7. {
8. flag = 0 ; // 逻辑错误
9. for(i=2,k=x>>1;i<=k;i++)
10. if (!(x%i) )
11. { // 疑似语法错误
12. flag = 1;
13. break;
14. }
15. if(flag==1) printf("%d是合数\n", x); // 第三处
16. else printf("%d不是合数\n", x);
17. }
18. return 0;
19. }

（4）测试

(a)测试用例

选择的测试用例如表2-1所示：

表2-1 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 2 | 2不是合数 |
| 用例2 | 3 | 3不是合数 |
| 用例3 | 10 | 10是合数 |
| 用例4 | ^Z | 结束 |

(b)对应表2-1测试用例的运行结果如图2-1所示：

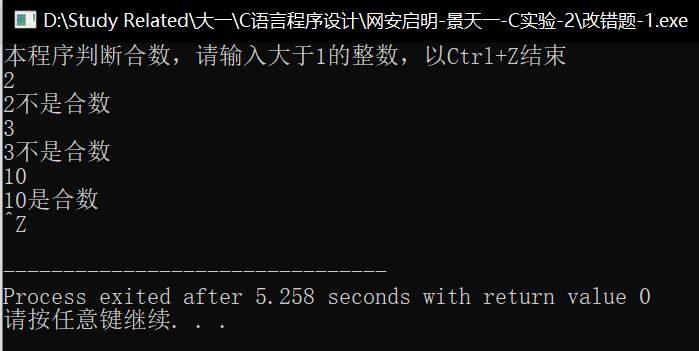


图2-1 实验2-1修改后运行结果截图

**2.2.2 程序修改替换**

（1）修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句。

（2）修改实验2-1程序，将for循环改用do-while循环。

（3）修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。

**解答：**

1. 修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句，按照要求修改后的程序如下：

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF)
7. {
8. flag = 0 ;
9. for(i=2,k=x>>1;(i<=k)&&(flag != 1);i++)
10. if (!(x%i) )
11. {
12. flag = 1;
13. }
14. if(flag==1) printf("%d是合数", x);
15. else printf("%d不是合数", x);
16. }
17. return 0;
18. }
19. 修改实验2-1程序，将for循环改用do-while循环。按照要求修改后的程序如下：

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF)
7. {
8. flag = 0 ;
9. i = 2 ;
10. k = x>>1 ;
11. do
12. {
13. if (!(x%i) && (x != 2 ) && (x!=0))
14. {
15. flag = 1;
16. }
17. i++ ;
18. }while((i<=k)&&(flag != 1));
20. if(flag==1) printf("%d是合数\n", x);
21. else printf("%d不是合数\n", x);
22. }
23. return 0;
24. }
25. 修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。
26. 解题思路：通过for 循环遍历所有3位数，调用前两题的素数判断函数进行判断，通过对输入数据除以10和100获取第二位，第三位，若判断后均为合数，则输出该数据；遍历所有3位数后结束循环。
27. 修改后程序的源代码如下：
28. #include<stdio.h>
29. int composite(int x)
30. {
31. int i, k, flag = 0;
32. for(i=2,k=x>>1;i<=k;i++)
33. if (!(x%i) )
34. {
35. flag = 1;
36. break;
37. }
38. return flag ;
39. }
40. int main(void)
41. {
42. for (int init=100;init < 1000;init++)
43. {
44. if ((composite(init)==1)&&(composite(init/10)==1)&&(composite(init/100)==1))printf("%d ",init);
45. }
46. return 0 ;
47. }
48. 优化和替换后运行结果如图2-2所示。

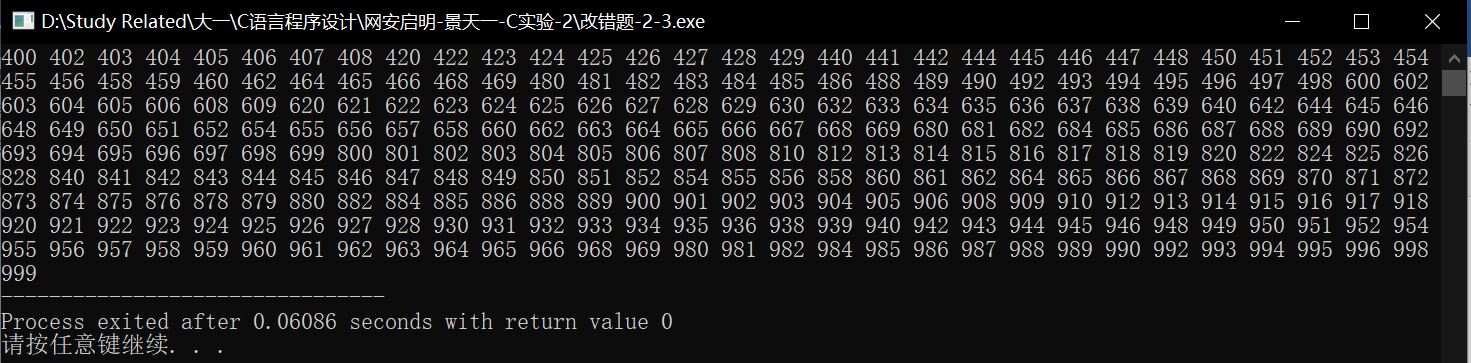
****

图2-2 实验2-2优化替换后运行结果截图

**2.2.3 程序设计**

（1）假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；x＞5000，收取25%的税金。（注意税金的计算按照阶梯计税法，比如，工资为4500，那么税金=1000\*5% + 1000\*10% + 1000\*15% + 501\*20%）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

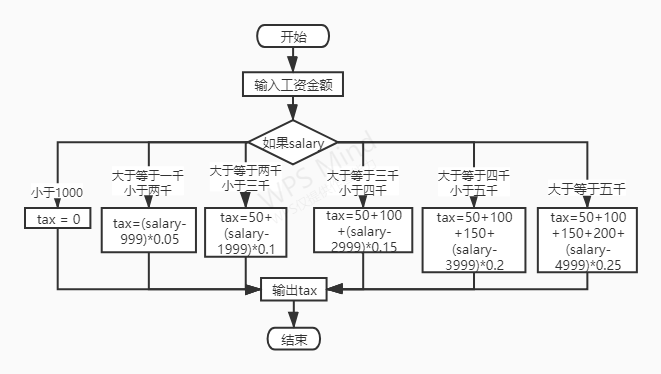
1. 算法流程
2. 程序设计题1 if语句版流程图如图2-3所示：

图2-3 程序设计题1程序流程图

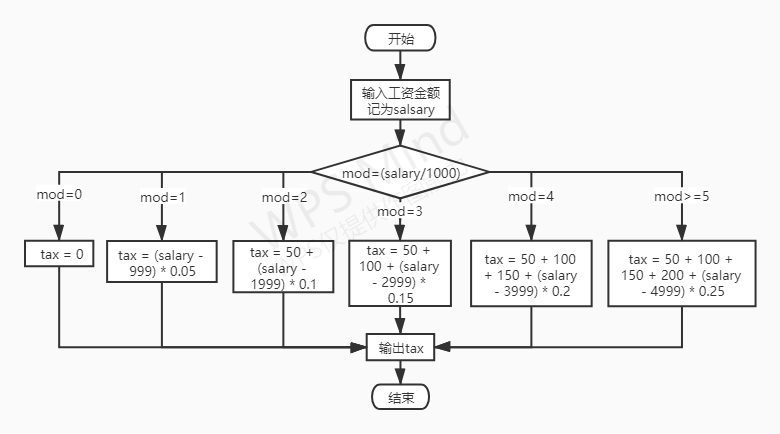
1. 程序设计题1 switch语句版流程图如图2-4所示：

图2-4 程序设计题1程序流程图

2) 源程序清单

1. 程序设计题1 if语句版源程序如下所示：
2. #include<stdio.h>
3. int main(void)
4. {
5. float salary = 0 ;
6. float tax = 0;
7. printf("请输入工资金额：") ;
8. scanf("%f",&salary) ;
10. if (salary < 1000)
11. {
12. tax = 0 ;
13. }
14. else if (salary >= 1000 && salary < 2000)
15. {
16. tax = (salary - 999) \* 0.05 ;
17. }
18. else if (salary >= 2000 && salary < 3000)
19. {
20. tax = 50 + (salary - 1999) \* 0.1 ;
21. }
22. else if (salary >= 3000 && salary < 4000)
23. {
24. tax = 50 + 100 + (salary - 2999) \* 0.15 ;
25. }
26. else if (salary >= 4000 && salary < 5000)
27. {
28. tax = 50 + 100 + 150 + (salary - 3999) \* 0.2 ;
29. }
30. else if (salary >= 5000)
31. {
32. tax = 50 + 100 + 150 + 200 + (salary - 4999) \* 0.25 ;
33. }
35. printf("税金额度为：%.2f\n",tax) ;
36. return 0 ;
37. }
38. 程序设计题1 switch语句版源程序如下所示：
39. #include<stdio.h>
40. int main(void)
41. {
42. float salary = 0 ; // 声明变量
43. float tax = 0;
44. printf("请输入工资金额：") ; // 输入工资金额
45. scanf("%f",&salary) ;
46. int mod=0 ;
47. mod = (salary / 1000) ; // 通过区分除数来判断大小
48. switch(mod)
49. {
50. case 0 : tax = 0 ;break ;
51. case 1 : tax = (salary - 999) \* 0.05 ;break ;
52. case 2 : tax = 50 + (salary - 1999) \* 0.1 ;break ;
53. case 3 : tax = 50 + 100 + (salary - 2999) \* 0.15 ;break ;
54. case 4 : tax = 50 + 100 + 150 + (salary - 3999) \* 0.2 ;break ;
55. default: tax = 50 + 100 + 150 + 200 + (salary - 4999) \* 0.25 ;
56. }
58. printf("税金额度为：%.2f",tax) ;
59. return 0 ;
60. }

3）测试

（a） 测试数据：

选择的测试用例如表2-2所示：

表2-2 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 900 | 0 |
| 用例2 | 4500 | 400.2 |
| 用例3 | 6000 | 750.25 |

（b） 对应测试数据的运行结果截图

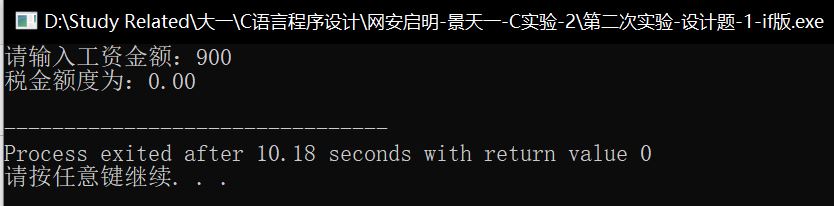
i) 对应测试测试用例1的运行结果如图2-5所示。

图2-5 程序设计题1的测试用例一的运行结果

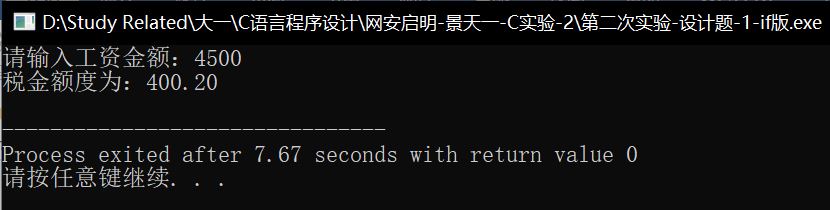
对应测试测试用例2的运行结果如图2-6所示。

图2-6 程序设计题1的测试用例二的运行结果

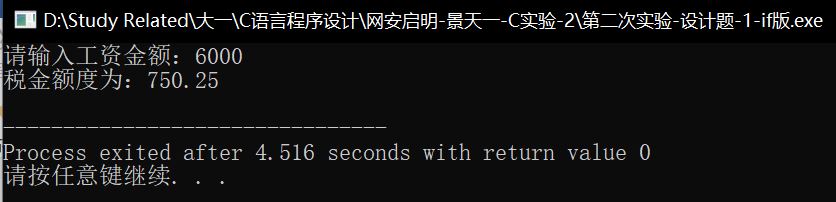
对应测试测试用例3的运行结果如图2-7所示。

图2-7 程序设计题1的测试用例三的运行结果

（2）将输入的正文复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替。

**解答：**

1) 解答思路：

程序设计题1 switch语句版流程图如图2-8所示：

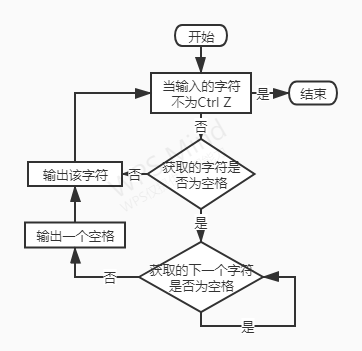


图2-8程序设计题2程序流程图

2)程序清单

1. #include<stdio.h>
2. int main(void)
3. {
4. char ch ; // 声明字符ch；
5. while (scanf("%c",&ch) != EOF) // 设定停止条件：输入ctrl z;
6. {
7. if (ch == ' ') // 若获得的字符为空格，则继续读取直至获取的字符部位空格为止；
8. {
9. while(ch == ' ')
10. {
11. scanf("%c",&ch);
12. }
13. putchar(' '); // 输出一个空格；
14. putchar(ch) ;
15. }
16. else
17. {
18. putchar(ch); // 若获得的字符不为空格，则直接打印字符；
19. }
20. }
21. return 0 ;
22. }

3）测试

（a） 测试数据：

测试数据如表2-2所示。

表2-2 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | □□□abc□□□12 | □abc□12 |
| 用例2 | □□□□□□□a | □a |
| 用例3 | 1□□□2□□□3 | 1□2□3 |
| 用例4 | ^Z | 结束 |

（b） 对应测试测试用例1，2，3，4的运行结果如图2-9所示。

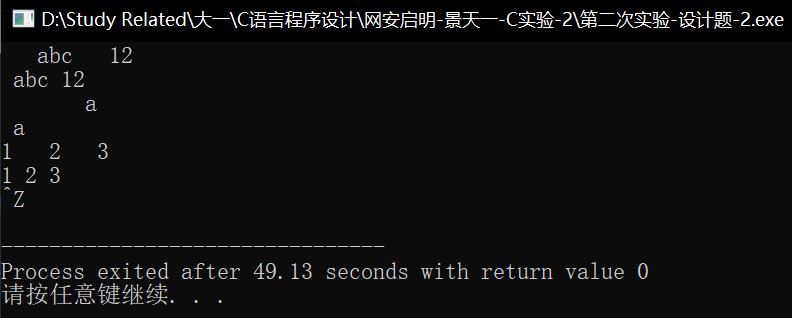


图2-9 程序设计题2的测试用例一的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）打印如下的杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

第i行第j列位置的数据值可以由组合表示，而的计算如下：

|  |  |
| --- | --- |
|  | (i=0,1,2,…) |
|  | (j=0,1,2,3,…,i) |

根据以上公式，采用顺推法编程，输出金字塔效果的杨辉三角形。特别要注意空格的数目，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。

**解答：**

1) 解答思路：

程序设计题3流程图如图2-10所示：

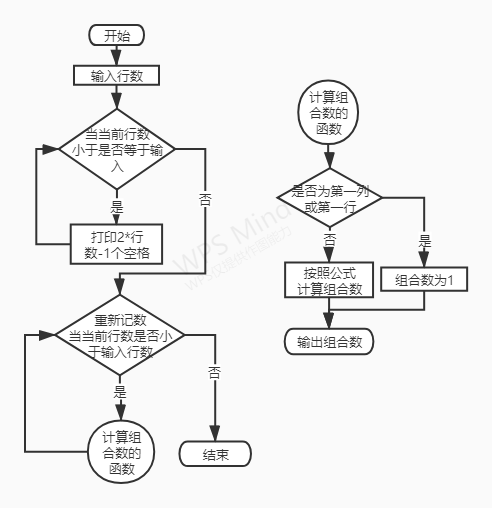


图2-10 程序设计题3程序流程图

1. 源程序清单
2. #include<stdio.h>
3. int factorial(int num) //计算阶乘的函数
4. {
5. int sum =1 ;
6. while(num !=1) // 封装来节省行数
7. {
8. sum = sum \* num ;
9. num-- ;
10. }
11. return sum ;
12. }
13. int number(int i, int j) // 输出第j行第i列的数字
14. {
15. int sum = 1 ;
16. i-- ;
17. j-- ;
18. if (i == 0 || i == j) //若计算第一列或最后一列的数字，则直接输出1
19. sum = 1 ;
20. else
21. {
22. sum = (factorial(j)/(factorial(i)\*factorial(j-i))) ; //调用阶乘函数
23. }
24. return sum ;
25. }
26. int main(void)
27. {
28. int line=0, j=1 ; // line表示最大行数，j表示当前行数
29. printf("请输入行数：") ;
30. scanf("%d",&line) ;
31. line++ ;
32. while (j <= line) // 当前行数达到输出最大行数时停止
33. {
34. // 首先打印左侧空格
35. int count=1 ;
36. while (count < (2\*(line - j + 1) - 1)) // 每行前空格为行数乘2减1
37. {
38. printf(" ") ;
39. count++ ;
40. }
41. // 依次打印数字
42. count = 1 ; // count表示当前列数 j表示当前列数
43. while (count <= j)
44. {
45. int num ;
46. num = number(count,j) ;
47. printf("%-4d",num) ;
48. if (count == j)
49. printf("\n") ;
50. count++ ;
51. }
53. j++ ;
54. }
56. return 0 ;
57. }
58. 测试
59. 测试数据

测试数据如表2-3所示：

表2-3 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试数据 | 预计输出 |
| 用例1 | 5 |  |
| 用例2 | 9 |  |
| 用例3 | 10 |  |

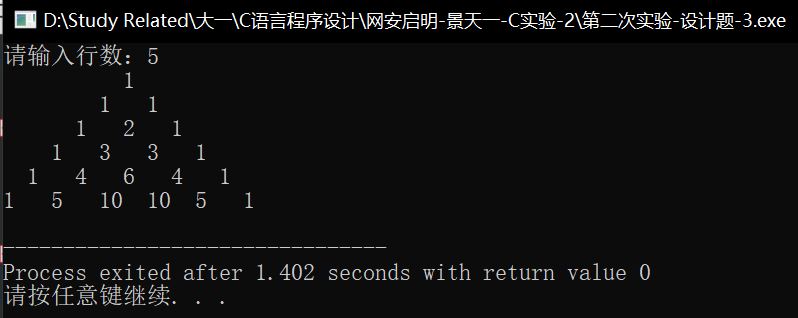
1. 对应测试测试用例1的运行结果如图2-11所示。

图2-11 程序设计题3的测试用例1的运行结果

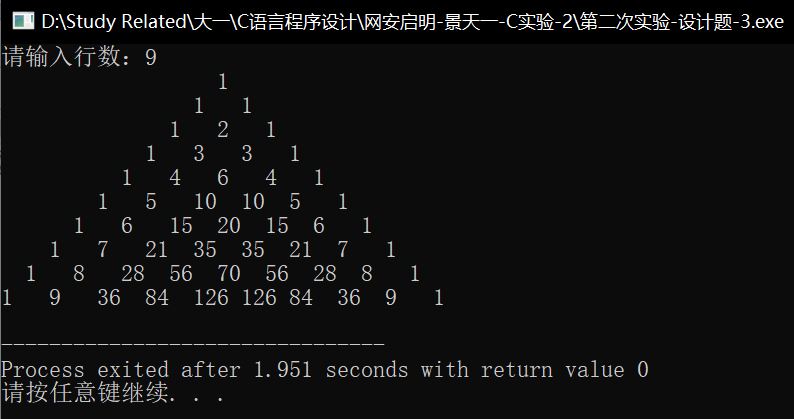
对应测试测试用例2的运行结果如图2-12所示。

图2-12 程序设计题3的测试用例2的运行结果

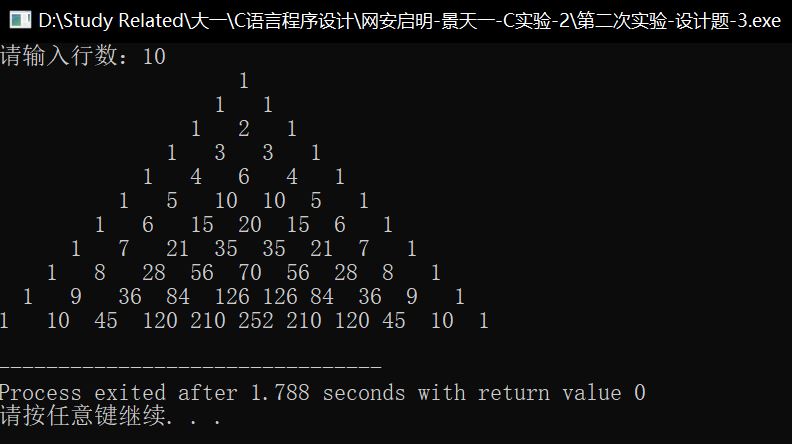
对应测试测试用例3的运行结果如图2-13所示。

图2-13 程序设计题3的测试用例3的运行结果

1. 625这个数很特别，625的平方等于390625，其末3位也是625。请编程输出所有这样的3位数：它的平方的末3位是这个数本身。

**解答：**

* + 1. 解答思路：

程序设计题4流程图如图2-14所示：

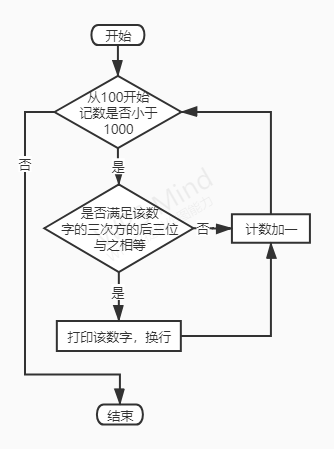


图2-14 程序设计题4程序流程图

1. 源程序清单
2. #include <stdio.h>
3. int main(void)
4. {
5. for (int num = 100;num < 999;num++) // 遍历所有三位数
6. {
7. if ((num == ((num \* num)%1000)) && (num <= 999)) // 题目条件
8. {
9. printf("%d\n", num) ;
10. }
11. }
12. return 0 ;
13. }

3) 测试

预计输出结果：376 625

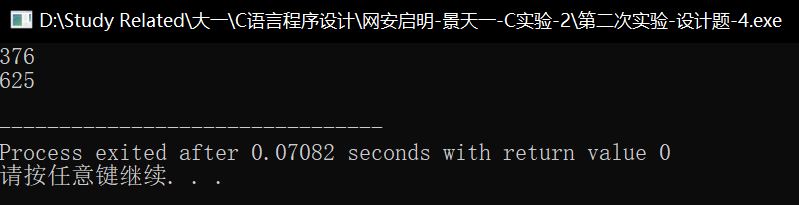
 程序设计题4的运行结果如图2-15所示：

图2-15 程序设计题4的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 2.3 实验小结

在编程设计题1中最初没有理解分段收取税金的含义，导致输出结果与预计不一致，说明每次在写程序时，最好是确保已经完全弄懂题目的目的和含义再进行操作。在编程设计题3中，因为不了解double型占位符的作用导致程序设计卡壳；在查阅资料了解了占位符和左对齐后解决了问题。

另外，在此次实验中，我切实感受到了程序设计时所要注意的一些问题，尤其是程序的可修改性，如果将主要重复的模块封装为函数，将大大节省调试步骤和编写时间。

# 3 函数与程序结构实验

## 3.1 实验目的

（1）熟悉和掌握函数的定义、声明；函数调用与参数传递，函数返回值类型的定义和返回值使用。

（2）熟悉和掌握不同存储类型变量的使用。

（3）练习使用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。

## 3.2 实验内容

**3.2.1 程序改错题**

下面是计算s=1!+2!+3!+…+n!的源程序(n<20)。在这个源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够输出如下结果：

k=1 the sum is 1

k=2 the sum is 3

k=3 the sum is 9

……

k=20 the sum is 2561327494111820313

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

1. #include <stdio.h>
2. int main(void)
3. {
4. int k;
5. for(k=1;k<=20;k++)
6. printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));
7. return 0;
8. }
9. long sum\_fac(int n)
10. {
11. long s=0;
12. int i,fac;
13. for(i=1;i<=n;i++)
14. fac\*=i;
15. s+=fac;
16. return s;
17. }

**解答：**

1. 语法错误修改：

1) 没有在文件头声明自定义函数，正确形式为：

long long sum\_fac(int n) ;

2) 第6行的printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));没有使用正确的占位符，正确形式为：

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

1. 逻辑错误修改：
2. 第9行long sum\_fac(int n)函数定义的返回值类型有误，正确形式为：

long long sum\_fac(int n)

1. 第11，12行变量s和变量fac都应为long long型，正确形式为：

long long s=0,fac=1;

int i;

1. 第15行s+=fac;应该包含在for循环内，正确形式为：

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

{

fac\*=i;

s+=fac; // 该语句应该在for循环内

}

修改后，源程序清单如下：

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

1. #include <stdio.h> // 没有声明计算阶乘和的函数，故在main()中无法进行调用
2. long long sum\_fac(int n) ; // 如上
3. int main(void)
4. {
5. int k;
6. for(k=1;k<=20;k++)
7. printf("k=%d\tthe sum is %lld\n",k,sum\_fac(k)); // 占位符改为lld
8. return 0;
9. }
10. long long sum\_fac(int n) // long long
11. {
12. long long s=0,fac=1;
13. int i; // fac,s 也应为long long类型 ，并应该赋初始值
14. for(i=1;i<=n;i++)
15. {
16. fac\*=i;
17. s+=fac; // 该语句应该在for循环内
18. }
19. return s;
20. }
21. 错误修改后运行结果如图3-1所示。

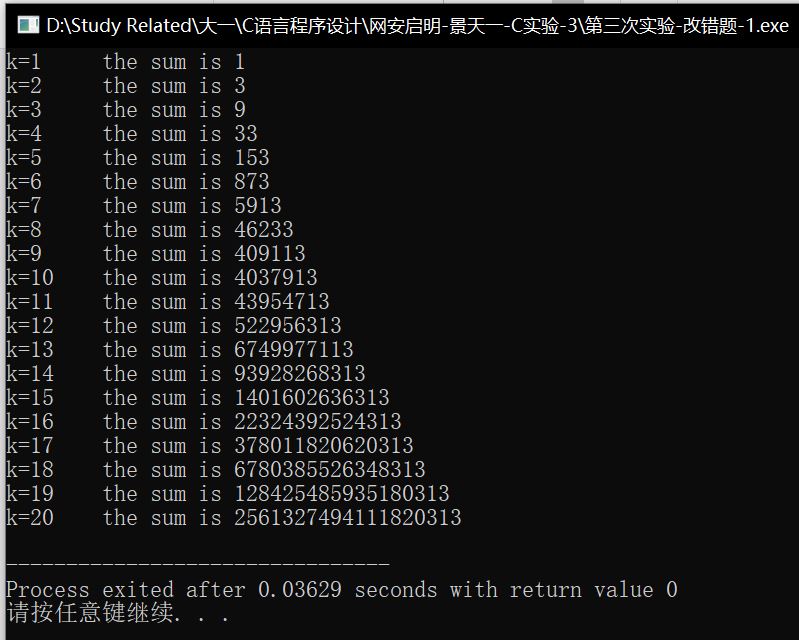
****

图3-1 实验3-1修改后运行结果截图

**3.2.2 源程序修改替换**

1. 根据将实验3-1改错题程序中sum\_fac函数修改为一个递归函数，用递归的方式计算。

**解答：**

按照要求修改后的sum\_fac()函数如下：

1. long long sum\_fac(int n)
2. {
3. if (n==1)
4. return 1 ;
5. long long fac=1, sum=0 ;
6. for (int count = 1;count<=n;count++)
7. fac \*= count ;
8. sum = fac + sum\_fac(n-1) ;
9. return sum ;
10. }

优化和替换后运行结果如图3-2所示。

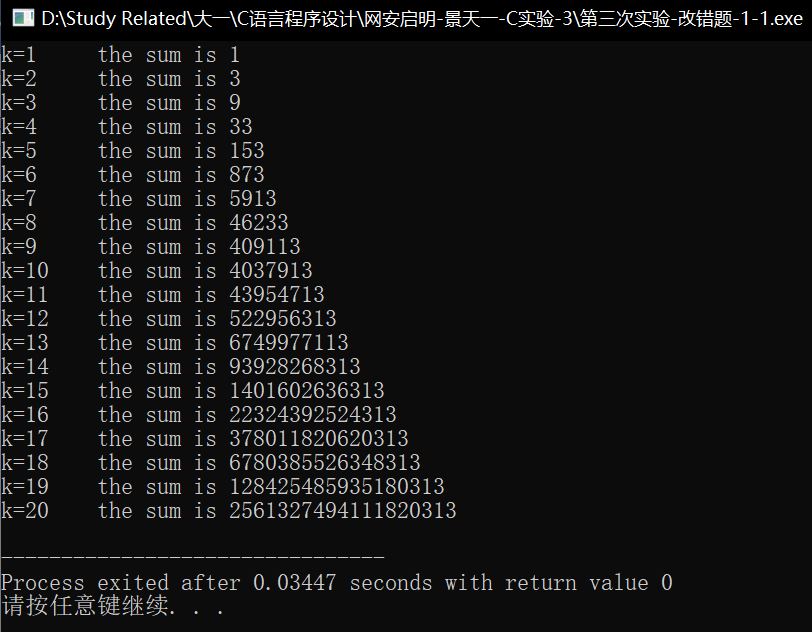


图3-2 实验3-2优化替换后运行结果截图

1. 下面是计算的源程序，其中x是浮点数，n是整数。从键盘输入x和n，然后计算s的值。修改该程序中的sum和fac函数，使之计算量最小。

/\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

1. #include<stdio.h>
2. double mulx(double x,int n);
3. long fac(int n);
4. double sum(double x,int n)
5. {
6. int i;
7. double z=1.0;
8. for(i=1;i<=n;i++)
9. {
10. z=z+mulx(x,i)/fac(i);
11. }
12. return z;
13. }
14. double mulx(double x,int n)
15. {
16. int i;
17. double z=1.0;
18. for(i=0;i<n;i++)
19. {
20. z=z\*x;
21. }
22. return z;
23. }
24. long fac(int n)
25. {
26. int i;
27. long h=1;
28. for(i=2;i<=n;i++)
29. {
30. h=h\*i;
31. }
32. return h;
33. }
34. int main()
35. {
36. double x;
37. int n;
38. printf("Input x and n:");
39. scanf("%lf%d",&x,&n);
40. printf("The result is %lf:",sum(x,n));
41. return 0;
42. }

**解答：**

1. 解答思路：

观察发现，在mulx()函数和fac()函数中存在重复计算的现象，可以通过将每次计算得到的值以静态变量的形式存储起来减少下次运算的计算步骤，当该函数下一次被调用时，仍可利用上一次计算得到的结果。

1. 按照要求优化后的程序源代码如下：
2. #include<stdio.h>
3. double mulx(double x) ;
4. long fac(int n);
5. double sum(double x,int n)
6. {
7. int i;
8. double z=1.0;
9. for(i=1;i<=n;i++)
10. {
11. z=z+mulx(x)/fac(i); // 去掉了一个参数
12. }
13. return z;
14. }
16. double mulx(double x)
17. {
18. static double z=1.0 ; // 利用静态变量减少多次计算的计算量
19. z = z \* x;
20. return z ;
21. }
22. long fac(int n)
23. {
24. static long h = 1 ; // 利用静态变量减少多次计算的计算量
25. h \*= n ;
26. return h ;
27. }
28. int main()
29. {
30. double x;
31. int n;
32. printf("Input x and n:");
33. scanf("%lf%d",&x,&n);
34. printf("The result is %lf:",sum(x,n));
35. return 0;
36. }
37. 测试
38. 测试数据

测试数据如表3-1所示：

表3-1 改错与替换题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | | 预计输出 |
| x | n |
| 用例1 | 3 | 3 | 13.000000 |
| 用例2 | 4 | 5 | 42.866667 |

1. 对应测试测试用例1的运行结果如图3-3所示。

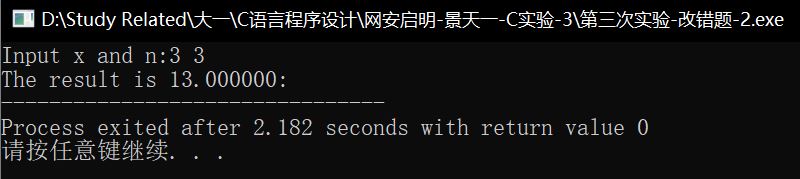


图3-3 改错与替换题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图3-4所示。

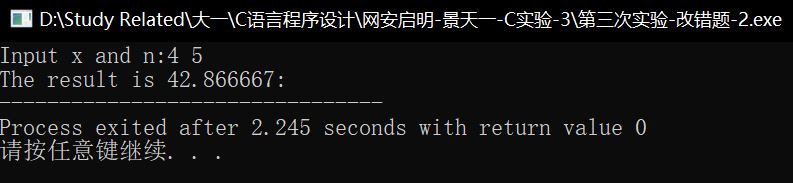


图3-4 改错与替换题2的测试用例二的运行结果

实际输出与预计输出相符，说明程序运行正确。

**3.2.3 跟踪调试题**

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，在watch窗口中观察Ik,sum,n值。具体操作如下：

（1）设输入5，观察刚执行完“scanf("%d",&k);”语句时，sum、k的值是多少？

（2）在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值是多少？返回后光条停留在哪个语句上？

（3）在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了哪里？

（4）在fabonacci函数内部单步执行，观察函数的递归执行过程。体会递归方式实现的计算过程是如何完成数计算的，并特别注意什么时刻结束递归，然后直接从第一个return语句返回到了哪里？

（5）在fabonacci函数递归执行过程中观察参数n的变化情况，并回答为什么k、sum在fabonacci函数内部不可见？

**/\*实验3-3跟踪调试题程序：**计算fabonacci数列前n项和**\*/**

#include<stdio.h>

int main(void)

{

int i,k;

long sum=0,fabonacci(int n);

printf("Inut n:");

scanf("%d",&k);

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

sum+=fabonacci(i);

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

}

return 0;

}

long fabonacci(int n)

{

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

return 1;

else

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

}

**解答：**

1. 设输入5，观察刚执行完“scanf("%d",&k);”语句时，sum、k的值是多少？

答：sum的值为0，k的值为5；

1. 在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值是多少？返回后光条停留在哪个语句上？

答：sum的值为0，k的值为0，i的值为0；返回后光条停留在printf("i=%d\tthe sum is %ld\n",i,sum);语句上

1. 在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了哪里？

答：调到了if(n==1 || n==2)

1. 在fabonacci函数内部单步执行，观察函数的递归执行过程。体会递归方式实现的计算过程是如何完成数计算的，并特别注意什么时刻结束递归，然后直接从第一个return语句返回到了哪里？

答：当n等于1或2时结束递归；从第一个return语句返回到printf("i=%d\tthe sum is %ld\n",i,sum);

1. 在fabonacci函数递归执行过程中观察参数n的变化情况，并回答为什么k、sum在fabonacci函数内部不可见？

答：sum和k是在main函数内定义的，其作用域只在main函数内，在fabonacci函数中未作外部变量引用声明，故在fabonacci函数内部不可见。

**3.2.4 程序设计题**

1. 编程验证歌德巴赫猜想：一个大于等于4的偶数都是两个素数之和。要求设计一个函数对其形参n验证哥德巴赫猜想，并以“n=n1+n2”的形式输出结果。例如：n=6，输出“6=3+3”。main函数循环接收从键盘输入的整数n，如果n是大于或等于4的偶数，调用上述函数进行验证。

**解答：**

1. 解题思路：

程序设计题1的解题思路如图3-5所示：

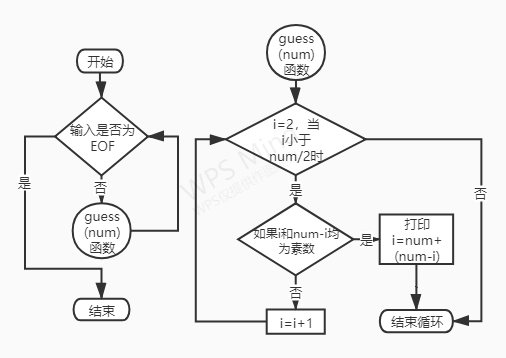


图3-5 程序设计题1的程序流程图

1. 程序清单
2. #include<stdio.h>
3. int composite(int x)
4. {
5. int i, k, flag = 0;
6. for(i=2,k=x>>1;i<=k;i++)
7. if (!(x%i) )
8. {
9. flag = 1;
10. break;
11. }
12. return flag ;
13. }
14. int guess(int num )
15. {
16. for (int i=2;i<=(num>>1);i++)
17. {
18. if((composite(i)==0)&&(composite(num-i)==0))
19. {
20. printf("%d=%d+%d\n",num,i,num-i) ;
21. break ;
22. }
23. }
24. return 0 ;
25. }
26. int main(void)
27. {
28. int num ;
29. printf("输入一个大于4的偶数：") ;
30. while(scanf("%d",&num)!=EOF)
31. {
32. guess(num) ;
33. }
34. return 0 ;
35. }
36. 测试

（a） 测试数据：

编程设计题1的测试数据如表3-2所示。

表3-2 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 预计输出 |
| 用例1 | 6 | 6=3+3 |
| 用例2 | 8 | 8=3+5 |
| 用例3 | ^Z | 结束 |

（b） 对应测试测试用例1，2，3的运行结果如图3-6所示。

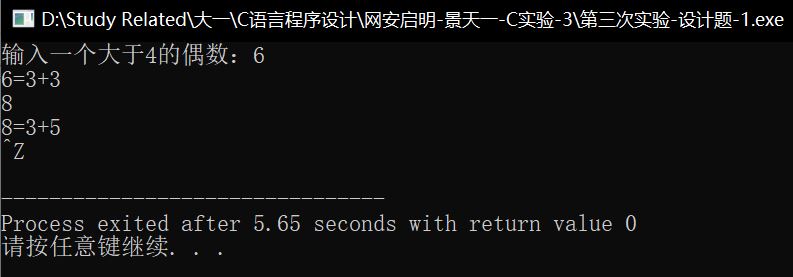


图3-6 程序设计题1的测试用例1，2，3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 完全数（Perfect number），又称完美数或完备数，特点是它的所有真因子（即除了自身以外的约数，包括1）之和恰好等一它本身。例如6=1+2+3，28=1+2+4+7+14等。编程寻找104以内的所有完全数。要求设计一个函数，判定形参n是否为完全数，如果是，则返回1，否则返回0。在main函数中调用该函数求104以内的所有完全数，并以完全数的真因子之和的形式输出结果，例如“6=1+2+3”；。

**解答：**

1. 解题思路：

程序设计题2的解题思路如图3-7所示：

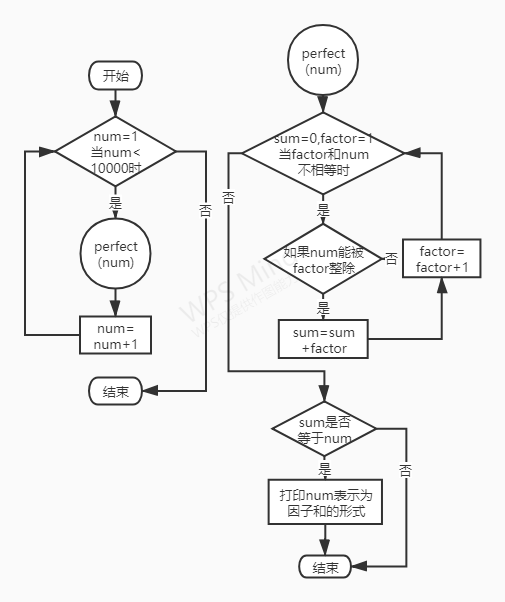


图3-7 程序设计题2的程序流程图

1. 源程序清单
2. int perfect(int num) // 定义一个输出完美数的函数
3. {
4. int sum = 0 ;
5. for (int factor=1;factor!=num;factor++)
6. {
7. if ((num%factor)==0)sum+=factor ;
8. }
9. if (sum==num)
10. {
11. printf("%d=1",num) ;
12. for (int factor=2 ;factor!=num ;factor++)
13. {
14. if ((num%factor)==0)printf("+%d",factor) ;
15. }
16. printf("\n") ;
17. }
18. return 0 ;
19. }
20. int main(void)
21. {
22. int top = 10000 ;
23. for (int num=1;num < top;num++)
24. {
25. perfect(num) ;
26. }
27. return 0 ;
28. }
29. 测试

程序设计题2运行结果如图3-8所示：

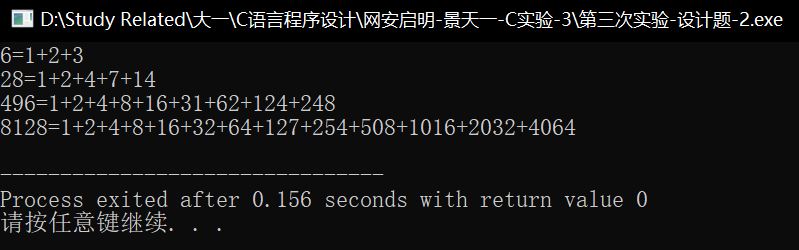


图3-8 程序设计题2的运行结果

1. 自幂数是指一个n位数，它的每个位上的数字的n次幂之和等于它本身。水仙花数是3位的自幂数，除此之外，还有4位的四叶玫瑰数、5位的五角星数、6位的六合数、7位的北斗星数、8位的八仙数等。编写一个函数，判断其参数n是否为自幂数，如果是，则返回1；否则，返回0。main函数能反复接收从键盘输入的整数k，k代表位数，然后调用上述函数求k位的自幂数，输出所有k位自幂数，并输出相应的信息，例如“3位的水仙花数共有4个153，370，371，407”。当k=0时程序结束执行。

**解答：**

1. 解题思路：

程序设计题3的解题思路如图3-9所示：

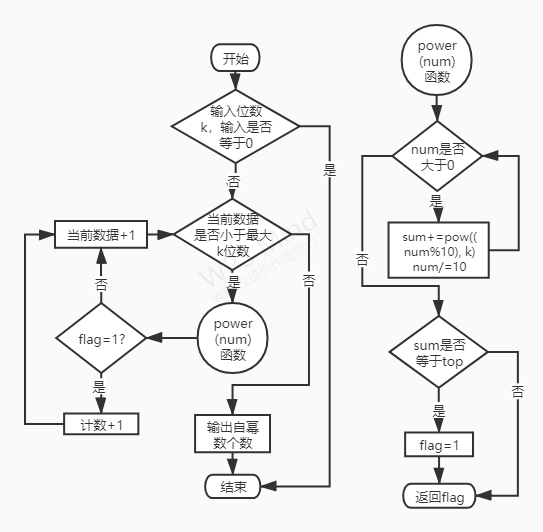


图3-9 程序设计题3的程序流程图

1. 源程序清单
2. #include<stdio.h>
3. #include<math.h>
4. int lenth(int x) // 定义了一个用于计算长度的函数
5. {
6. int n ;
7. for (n = 0; x > 0; n++) // 每次对输入数据除以10，当n等于0时终止
8. {
9. x /= 10;
10. }
11. return n ;
12. }
13. int power(int num) // 判断是否为自幂数，是输出1，否输出0
14. {
15. int flag=0, sum=0, k=0, top =0 ;
16. top = num ; // 用于存储输入数据，设定循环界限
17. k = lenth(num) ;
18. for (;num > 0 ;num /= 10) // num每次除以10，在取10的模，得到最后一位
19. {
20. sum += pow((num%10), k) ;
21. }
22. if (sum==top) // 若各位幂之和等于输入数据
23. flag = 1 ;
25. return flag ;
26. }
27. int main(void)
28. {
29. int k=1, top=0, min=0 ;
30. printf("输入所求自幂数的位数:") ;
31. for(;;)
32. {
33. scanf("%d",&k) ;
34. if (k==0) // 输入0时结束
35. break ;
36. min = pow(10,(k-1)) ; // 确定 k 位数边界
37. top = pow(10,(k)) ;
38. int count=0 ;
39. for(;min<top;min++)
40. {
41. if (power(min)==1) // 每出现一个自幂数，count加 1
42. count++ ;
43. }
44. printf("%d位的自幂数共有%d个:",k,count) ;
45. for(min = pow(10,(k-1));min<top;min++) // 再次遍历，输出所有 k 位自幂数
46. {
47. if (power(min)==1)
48. printf("%d ",min) ;
49. }
50. printf("\n") ;
51. }
52. return 0 ;
53. }
54. 测试
55. 测试数据

程序设计题3的测试用例如表3-3所示：

表3-3 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 预计输出 |
| 用例1 | 3 | 3位的自幂数共有4个:153 370 371 407 |
| 用例2 | 4 | 4位的自幂数共有3个:1634 8208 9474 |
| 用例3 | 0 | 结束 |

1. 对应测试测试用例1，2，3的运行结果如图3-10所示。

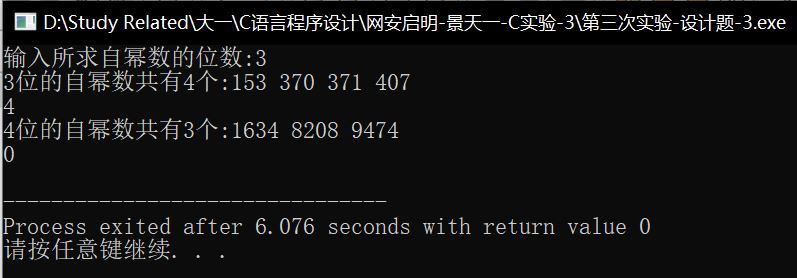


图3-8 程序设计题2的运行结果

实际输出结果与预计结果一致，说明程序运行正确。

## 3.3 实验小结

在减少计算量时，可以考虑使用static静态变量减少重复计算，节省内存空间；通过自定义函数，可以减少代码编写的时间，将不同功能模块进行封装，在进行测试时也可以更好地判断错误的所在。

本次实验里的一些题目，我感到还有修改优化的余地，在进行某个功能的实现和开发时应尽可能采用最优解法。另外，编辑器调试功能的使用可以方便进行改错，需要多加利用。

# 4 编译预处理实验

## 4.1 实验目的

（1）掌握文件包含、宏定义、条件编译和assert宏的使用；

（2）练习使用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。

（3）熟悉多文件编译技术

## 4.2 实验内容

**4.2.1 程序改错**

下面是用宏来计算平方差、交换两数的源程序.在这个源程序中存在若干错误，要求对该程序进行调试修改，使之能够正确完成指定任务。

/\*实验4-1改错与跟踪调试题程序：计算平方差、将换两数\*/

1. #include<stdio.h>
2. #define SUM a+b
3. #define DIF a-b
4. #define SWAP(a,b) a=b,b=a
5. int main()
6. {
7. int a,b;
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. return 0;
14. }

**解答：**

1. 语法错误修改：
2. 第2，3行的宏定义应该添加括号，正确形式为：

#define SUM (a+b)  // 加上括号

#define DIF (a-b)

2) 第4行的a=b,b=a，无法实现交换功能，正确形式为：

#define SWAP(a,b) a=SUM, b=DIF, a=DIF; // 更改为替换法

1. 错误修改后运行结果如图4-1所示。

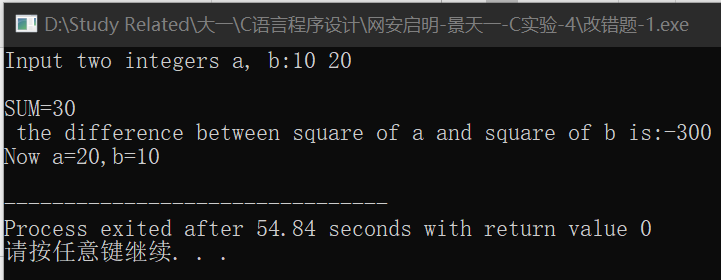


图4-1 实验1-1修改后运行结果截图

**4.2.2 程序修改替换**

下面是用函数实现求三个数中最大数、计算两浮点数之和的程序。在这个源程序中存在若干语法和逻辑错误。

要求：

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. 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。

/\*实验4-2程序修改替换题程序\*/

1. #include<stdio.h>
2. int main(void)
3. {
4. int a, b, c;
5. float d, e;
6. printf("Input three integers:");
7. scanf("%d %d %d",&a,&b,&c);
8. printf("\nThe maximum of them is %d\n",max(a,b,c));
9. printf("Input two floating point numbers:");
10. scanf("%f %f",&d,&e);
11. printf("\nThe sum of them is %f\n",sum(d,e));
12. return 0;
13. }
14. int max(int x, int y, int z)
15. {
16. int m=z;
17. if (x>y)
18. if(x>z) m=x;
19. else
20. if(y>z) m=y;
21. return m;
22. }
23. float sum(float x, float y)
24. {
25. return x+y;
26. }

**解答：**

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；

在文件开始处声明需要定义的函数，正确形式如下：

int max(int x, int y, int z) ; // 未声明函数

1. 用带参数的宏替换函数max，来实现求最大数的功能。

实现形式如下：

#define MAX(x,y,z) ((x>y)?((x>z)?x:z):((y>z)?y:z)) // 输出最大值

1. 优化和替换后程序的源代码如下：
2. #include<stdio.h>
3. #define MAX(x,y,z) ((x>y)?((x>z)?x:z):((y>z)?y:z)) // 输出最大值
4. int max(int x, int y, int z) ; // 声明函数
5. float sum(float x, float y) ;
6. int main(void)
7. {
8. int a, b, c;
9. float d, e;
10. printf("Input three integers:");
11. scanf("%d %d %d",&a,&b,&c);
12. // printf("\nThe maximum of them is %d\n",max(a,b,c));
13. printf("\nThe maximum of them is %d\n",MAX(a,b,c));
14. printf("Input two floating point numbers:");
15. scanf("%f %f",&d,&e);
16. printf("\nThe sum of them is %f\n",sum(d,e));
17. return 0;
18. }
20. int max(int x, int y, int z)
21. {
22. int m=z;
23. if (x>y)
24. if(x>z) m=x;
25. else
26. if(y>z) m=y;
27. return m;
28. }
29. float sum(float x, float y)
30. {
31. return x+y;
32. }

优化和替换后运行结果如图4-2所示。

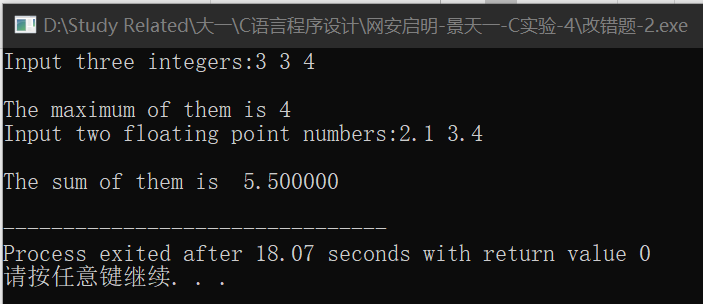


图4-2 实验4-2优化替换后运行结果截图

**4.2.3 跟踪调试**

下面程序利用R计算圆的面积s，以及面积s的整数部分。现要求：

（1）修改程序，使程序编译通过且能运行；

（2）单步执行。进入函数integerl\_fraction时，watch窗口中x为何值？在返回main时, watch窗口中i为何值？

（3）修改程序，使程序能输出面积s值的整数部分（要求四舍五入），不会输出错误信息assertion failed。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

1. #define R
2. int main(void)
3. {
4. float r, s;
5. int s\_integer=0;
6. printf ("Input a number: ");
7. scanf("%f",&r);
8. #ifdef R
9. s=3.14159\*r\*r;
10. printf("Area of round is: %f\n",s);
11. s\_integer=integer\_fraction(s);
12. assert((s-s\_integer)<0.5);
13. printf("The integer fraction of area is %d\n", s\_integer);
14. #endif
15. return 0;
16. }
17. int integer\_fraction(float x)
18. {
19. int i=x;
20. return i;
21. }

**解答：**

1. 程序改错：
2. 在文件开始处添加头文件声明#include<stdio.h>
3. 在文件开始处补充函数声明；
4. 单步执行：

输入r为2，进入函数integerl\_fraction时，watch窗口中x为12.5663605，在返回main时, watch窗口中i为12。

1. 修改程序：

第十二行，源程序无法实现四舍五入，修改如下：

if (s-s\_integer>0.5) // 四舍五入

s\_integer++;

1. 源程序清单如下：
2. #include<stdio.h> // 改1：头文件
3. #define R
4. int integer\_fraction(float x); // 补充函数声明
5. int main(void)
6. {
7. float r, s;
8. int s\_integer=0;
9. printf ("Input a number: ");
10. scanf("%f",&r);
11. #ifdef R
12. s=3.14159\*r\*r;
13. printf("Area of round is: %f\n",s);
14. s\_integer=integer\_fraction(s);
15. // assert((s-s\_integer)<0.5); 这样无法实现四舍五入
16. if (s-s\_integer>0.5) // 四舍五入
17. s\_integer++;
18. printf("The integer fraction of area is %d\n", s\_integer);
19. #endif
20. return 0;
21. }
22. int integer\_fraction(float x)
23. {
24. int i=x;
25. return i;
26. }

运行结果如图4-3所示：

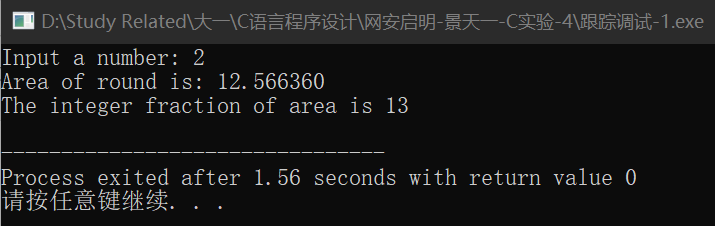


图4-3 跟踪调试题的运行结果图

符合预期结果，说明程序运行正确。

**4.2.4 程序设计**

1. 三角形的面积是，其中，a,b,c为三角形的三边，要求编写程序用带参数的宏来计算三角形的面积。定义两个带参数的宏，一个用来求s，另一个用来求area。

**解答：**

1. 解题思路：

利用math.h头文件中的pow()函数进行开方计算；为保证精确度，计算半周长s时应采用浮点数计算以保证精确度。

1. 程序清单：
2. #include<stdio.h>
3. #include<math.h>
4. #define S(a,b,c) (a+b+c)/2.0
5. #define AREA(a,b,c) pow((S(a,b,c)\*(S(a,b,c)-a)\*(S(a,b,c)-b)\*(S(a,b,c)-c)),0.5)
6. int main(void) {
7. int a=0,b=0,c=0 ;
8. printf("请输入三角形的三边：") ;
9. while(scanf("%d %d %d",&a,&b,&c)!=EOF) {
10. if ((a+b>c)&&(a+c>b)&&(b+c>a)) {
11. printf("Area is %f\n",AREA(a,b,c)) ;
12. }
13. else {
14. printf("%d,%d,%d三边无法组成三角形！\n",a,b,c);
15. }
16. printf("请输入三角形的三边：") ;
17. a=0;
18. b=0;
19. c=0;
20. }
21. return 0 ;
22. }

3）测试

（a） 测试数据：

程序设计题1的测试数据如表4-1所示。

表4-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 3，4，5 | 6.000000 |
| 用例2 | 2，2，2 | 2，2，2三边无法组成三角形 |
| 用例3 | ^Z | 结束 |

（b） 对应测试测试用例1，2，3的运行结果如图4-4所示。

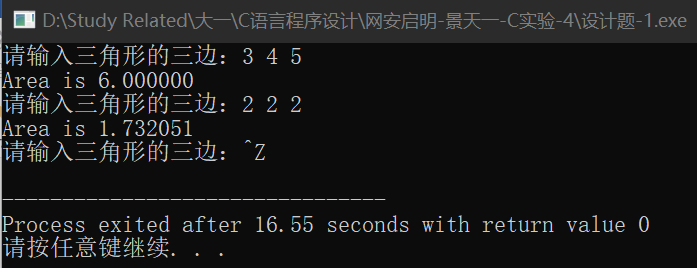


图4-4 程序设计题1的测试用例1，2，3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 用条件编译方法来编写程序。输入一行英文字符序列，可以任选两种方式之一输出：一为原文输出；二为变换字母的大小写后输出。例如小写‘a’变成大写‘A’，大写‘D’变成小写‘d’，其他字符不变。用#define命令控制是否变换字母的大小写。例如，#define CHANGE 1 则输出变换后的文字，若#define CHANGE 0则原文输出。

**解答：**

1. 解题思路

解题思路如图4-5所示。

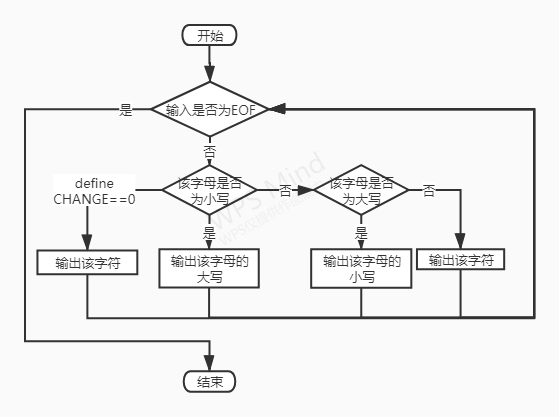


图4-5 程序设计题2算法流程图

1. 程序清单
2. #include<stdio.h>
3. #define CHANGE 0
4. char ch=0 ;
5. int main(void) {
6. while(scanf("%c",&ch)!=EOF) {
7. #if CHANGE==1
8. if (ch>=65 && ch<=90)
9. putchar(ch + 32) ;
10. else if (ch>=97 && ch<=122)
11. putchar(ch - 32) ;
12. else
13. putchar(ch) ;
14. #elif CHANGE==0
15. putchar(ch) ;
16. #endif
17. }
18. return 0 ;
19. }
20. 测试
21. 测试数据

程序设计题2测试数据如表4-2所示：

表4-2 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | aSdFgH  (CHANGE=1) | AsDfGh |
| 用例2 | ASdFgH  (CHANGE=0) | aSdFgH |
| 用例3 | ^Z | 结束 |

1. 对应测试测试用例1的运行结果如图4-6所示。

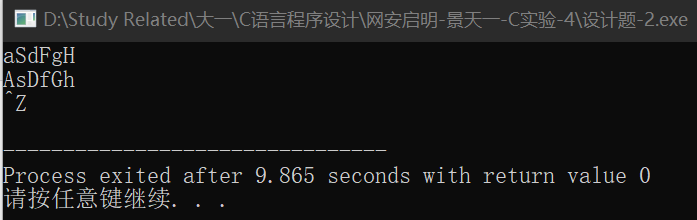


图4-6 程序设计题2的测试用例1的运行结果

1. 对应测试测试用例2, 3的运行结果如图4-7所示。

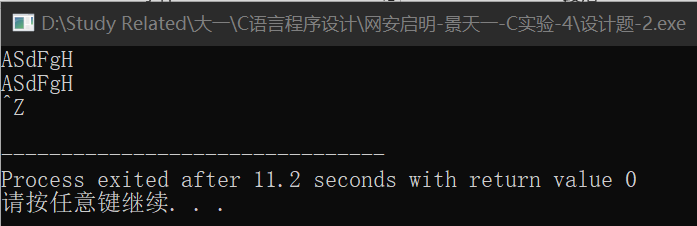


图4-7 程序设计题2的测试用例2, 3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**4.2.5 选做题**

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

**解答：**

1. 解题思路：

在Dev C++中生成一个项目文件，通过头文件链接的方式链接外部参数和函数。

1. 程序清单
2. /\*头文件file.h的内容\*/
3. #include<stdio.h>
4. extern int x;
5. extern int y;
6. extern char ch;
7. void func1(void);
8. /\*源文件file1.c的内容\*/
9. #include "file.h"
10. int x,y;
11. char ch;
12. int main(void) {
13. x=10;
14. y=20;
15. ch=getchar();
16. printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);
17. func1();
18. return 0;
19. }
20. /\*源文件file2.c的内容为：\*/
21. #include"file.h"
22. void func1(void) {
23. x++;
24. y++;
25. ch++;
26. printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);
27. }
28. 测试

输入字符a，预计输出：

in file1 x=10,y=20,ch is a

in file2 x=11,y=21,ch is b

编译运行后file1.c运行结果如图4-8所示。

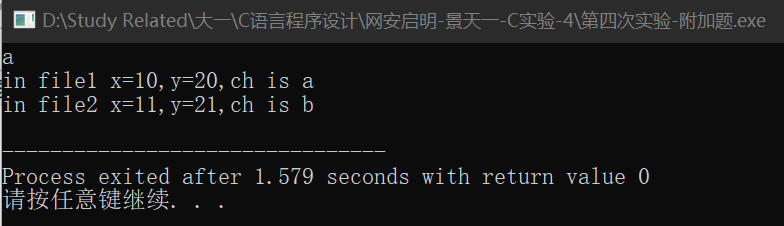


图4-8 选做题的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 4.3 实验小结

本次实验主要练习了编译预处理的相关内容，包括头文件的声明，自定义文件的链接，宏定义，带参数的宏定义。头文件声明是C/C++程序的重要组成部分，许多库函数为程序编写提供了很大的方便；自定义文件链接为开发者自行编写库，对大型项目进行模块化设计和开发管理提供了可能；宏定义在一定程度上简化了重复出现参数，同时可以更加清楚的表示其含义。

在编写选做题的过程中，我在链接文件时遇到了困难，询问助教后通过使用搭建DevC++项目的形式来解决文件链接问题。但这也引出了我的疑问：如何在不适用该项目，不依赖IDE的情况下在多平台实现多文件链接？

# 5 数组实验

## 5.1 实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

## 5.2 实验内容

**5.2.1 源程序改错与跟踪调试**

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgrmingLnguge

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。
2. 跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。

/\*实验5-1程序改错与跟踪调试题程序\*/

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[]="Programming";
7. printf("%s %s\n", b,a);
8. strcate(b,a); printf("%s %s\n",b,a);
9. strdelc(b, 'a'); printf("%s\n",b);
10. return 0;
11. }
12. void strcate(char t[],char s[])
13. {
14. int i = 0, j = 0;
15. while(t[i++]) ;
16. while((t[i++] = s[j++] )!= '\0');
17. }
18. void strdelc(char s[], char c)
19. {
20. int j,k;
21. for(j=k=0; s[j] != '\0'; j++)
22. if(s[j] != c) s[k++] = s[j];
23. }

**解答：**

1. 程序改错：
2. 第五行的b[]="Programming"，需要给b字符串增长空间至可以容纳a字符串，正确形式为：

b[30]="Programming"

1. 第16行需要让计数变量i退回到’\0’的位置，正确形式为：

i--;

1. 第25行，需要为数组的最后一个元素增加终止符，正确形式为：

s[k] = ‘\0’;

1. 修改完上述错误后单步执行源程序：
2. 当单步执行光条刚落在第二个while语句所在行时，i为11，t[i]为0，与预期相符；当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为ProgrammingLanguage和Language，实现了字符串连接。
3. 跟踪进入函数strdelc时，字符数组s中的内容为Language，字符c的值为a，结果正确；字符数组s, j和k值的变化不存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为ProgrmmingLnguge，实现了所要求的删除操作。
4. 源代码清单：
5. #include<stdio.h>
6. void strcate(char [],char []);
7. void strdelc(char [],char );
8. int main(void) {
9. char a[]="Language", b[30]="Programming"; // 为连接的字符串留出足够的空间
10. printf("%s %s\n", b,a);
11. strcate(b,a);
12. printf("%s %s\n",b,a);
13. strdelc(b, 'a');
14. printf("%s\n",b);
15. return 0;
16. }
17. void strcate(char t[],char s[]) {
18. int i = 0, j = 0;
19. while(t[i++]) ;
20. i--; // 退回到'\0'的位置
21. while((t[i++] = s[j++] )!= '\0');
22. }
23. void strdelc(char s[], char c) {
24. int j,k;
25. for(j=k=0; s[j] != '\0'; j++) {
26. if(s[j] != c)
27. s[k++] = s[j];
28. }
29. s[k] = '\0'; // 为最后一位添加终止符
30. }
31. 错误修改后运行结果如图5-1所示。

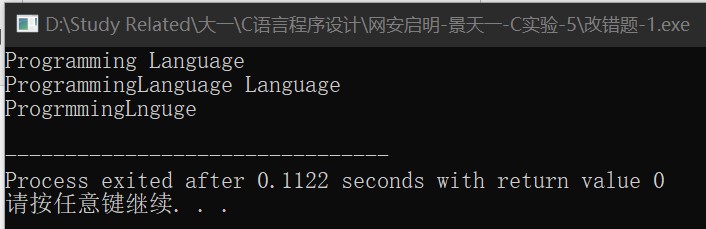


图5-1 实验5-1修改后运行结果截图

与预期结果相同，说明源代码正确。

**5.2.2 源程序完善和修改替换**

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。①请在源程序中的下划线处填写合适的代码来完善该程序。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。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

1. #include<stdio.h>
2. #define M 10
3. #define N 3
4. int main(void)
5. {
6. 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 \*/
7. int i, j, k;
8. for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/
9. a[i] = i + 1;
10. for(i = M, j = 0; i > 1; i--){
11. /\* 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 \*/
12. for(k = 1; k <= N; k++) /\* 1至N报数 \*/
13. 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 \*/
14. b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/
15. if(j)
16. for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/
17. \_\_\_\_\_\_\_\_\_\_\_\_\_\_;
18. }
19. for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/
20. printf(“%6d”, b[i]);
21. printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/
22. return 0;
23. }

**解答：**

1. 在下划线处填写合适的语句：
2. 第一处：a[j-1]:a[i-1]
3. 第二处：a[k] = a[k+1]
4. 修改后的运行结果如图5-2所示：

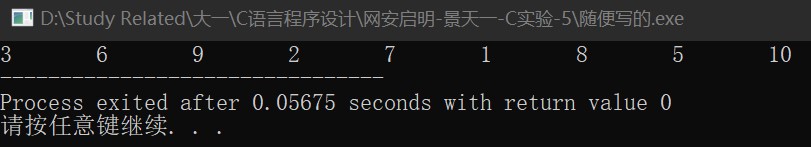


图5-2 实验5-2修改后运行结果截图

1. 按照要求优化程序：
2. 设计思路：

在一个数组中以0或1储存出圈或非出圈的状态，用数组中元素的位置表示编号；每次循环时，将出圈者的状态由1改为0进行标记；若循环位置状态为0，则跳过该位置。

1. 源程序清单：
2. #include<stdio.h>
3. #define M 10
4. #define N 3
5. int main(void) {
6. int lst[M]= {0}, return\_list[M-1]={}, count = 1, out\_num = 0;
7. for (int i=0; (i<M && out\_num<M); i++) {
8. if (!lst[i]) {
9. if (count==N) {
10. lst[i] = 1;
11. return\_list[out\_num] = i + 1;
12. out\_num++;
13. count = 1;
14. } else {
15. if (i == M-1)
16. i=-1;
17. count++;
18. }
19. }
20. }
21. for (int i=0; i<M-1; i++)
22. printf("%d\t",return\_list[i]);
23. return 0;
24. }
25. 优化和替换后运行结果如图5-3所示。

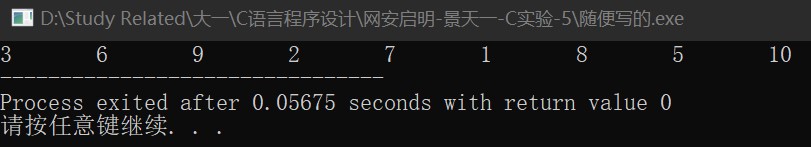


图5-3实验5-2优化替换后运行结果截图

**5.2.3 程序设计**

1. 输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

**解答：**

1. 算法流程如图5-4所示。

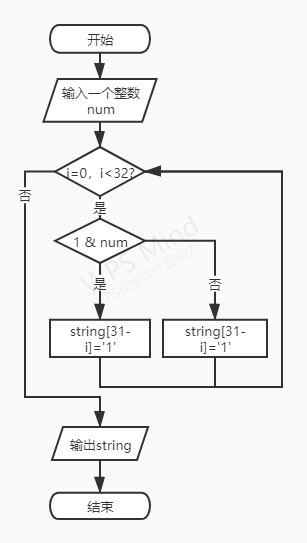


图5-4 程序设计题1的程序流程图

1. 源程序清单
2. #include <stdio.h>
3. int main(void)
4. {
5. int num;
6. char string[33];
7. int m, i;
8. printf("请输入要计算的数字：\n");
9. scanf("%d", &num);
10. for (i = 0, m = 1; i < 32; i++) {
11. string[31-i] = m & num ? '1' : '0'; //将结果存入一个字符数组中
12. m <<= 1; //对每一位用逻辑尺得到结果
13. }
14. string[32] = '\0';//将字符数组转变为字符串
15. printf("%s", string);//打印结果
16. return 0;
17. }
18. 测试

（a） 测试数据：

编程设计题1的测试数据如表5-1所示：

表5-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 1 | 00000000000000000000000000000001 |
| 用例2 | -1 | 11111111111111111111111111111111 |
| 用例3 | -2 | 11111111111111111111111111111110 |

（b） 对应测试测试用例1的运行结果如图5-5所示。



图5-5 程序设计题1的测试用例一的运行结果

对应测试测试用例2的运行结果如图5-6所示。

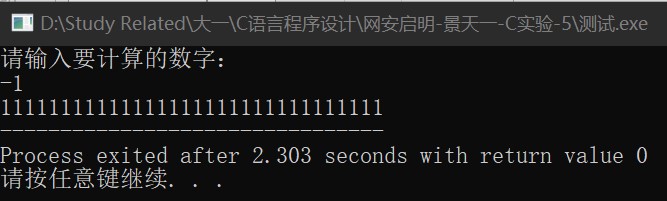


图5-6 程序设计题1的测试用例二的运行结果

对应测试测试用例3的运行结果如图5-7所示。

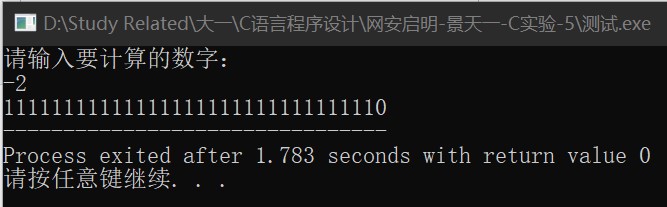


图5-7 程序设计题1的测试用例三的运行结果

1. 编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：

①输入n个学生的姓名和C语言课程的成绩。

②将成绩按从高到低的次序排序，姓名同时进行相应调整。

③输出所有学生的姓名和C语言课程的成绩。

1. 对程序设计第（2）题的程序增加查找功能：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，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

**解答：**

1. 解题思路：
2. 分别输入学生成绩和姓名；
3. 按照成绩从高到低进行选择排序；
4. 输出指令菜单：输入指令以选择输出姓名成绩或按成绩查询姓名；
5. 若按成绩查询姓名：利用二分查找查询某一成绩对应的学生姓名（之一），再分别向前向后查找同成绩的学生，打印对应姓名，若成绩不相同则停止查找；
6. 程序清单
7. #include<stdio.h>
8. #include<stdlib.h>
9. #define SWAP(a,b) a=(a-b), b=(a+b), a=(b-a);
10. // 函数模块声明
11. int main(void); // 主函数
12. void menu(int [], char (\*)[10]); // 目录函数
13. void input(int [], char (\*)[10]); // 输入姓名，成绩
14. void output(int [], char (\*)[10]); // 输出成绩，姓名
15. void range(int [], char (\*)[10]); // 排序函数
16. void search(int [], char (\*)[10]); // 按照姓名搜索成绩的函数
17. // 实用函数声明
18. void repeatPrint(int num, char ch, char end); // 用于重复打印的函数
19. // 外部变量声明
20. int listlen=0; // 用于定义数组长度的变量
21. // 主函数
22. int main(void) {
23. repeatPrint(80,'=','\n');
24. printf("\n\t\t\t# 学生成绩排名与查询系统\n\n\t\t\t# 输入 1 进行成绩录入\n\t\t\t# 输入 其他字符 退出\n\n");
25. repeatPrint(80,'=','\n');
26. int order;
27. printf("$ >");
28. scanf("%d",&order);
29. if (order==1) {
30. printf("#请输入学生人数：");
31. scanf("%d",&listlen);
32. int scores[listlen];
33. char names[listlen][10];
34. input(scores, names);
35. range(scores, names);
36. while(1)
37. menu(scores, names);
38. } else
39. exit(1);
40. return 0;
41. }
42. // 目录函数
43. void menu(int scores[], char (\*names)[10]) {
44. repeatPrint(80,'=','\n');
45. printf("\n\t\t\t# 学生成绩排名与查询系统\n\n\t\t\t# 输入 2 进入成绩查询\n\t\t\t# 输入 3 输出成绩排名\n\t\t\t# 输入 其他字符 退出\n\n");
46. repeatPrint(80,'=','\n'); // #Step1：输入学生人数\n\t#Step2：输入“[学生姓名 学生成绩]
47. int order;
48. printf("$ >");
49. scanf("%d",&order);
50. system("cls");
51. if (order==2)
52. search(scores, names);
53. else if (order==3)
54. output(scores, names);
55. else
56. exit(1);
57. }
58. // 输入姓名成绩的函数
59. void input(int scores[], char (\*names)[10]) {
60. char ch=0;
61. for (int i=0; i<listlen; i++) {
62. printf("\n学生姓名：");
63. scanf("%s",names[i]);
64. printf("学生成绩：");
65. scanf("%d",&scores[i]);
66. }
67. }
68. // 排序函数(选择排序)
69. void range(int lst[], char (\*names)[10]) {
70. int n=listlen;
71. int count=0, count2=1;
72. for (count=0; count<n; count++) {
73. for (count2=count+1; count2<n; count2++) {
74. if (lst[count]>lst[count2]) {
75. SWAP(lst[count], lst[count2]);
76. for (int i=0; i<9; i++) {
77. SWAP(names[count][i], names[count2][i]);
78. }
79. }
80. }
81. }
82. }
83. // 用于输出的函数
84. void output(int scores[], char (\*names)[10]) {
85. for (int i=0; i < listlen; i++)
86. printf("姓名：%s\t成绩：%d\n",names[i], scores[i]);
87. }
88. // 用于重复打印的函数
89. void repeatPrint(int num, char ch, char end) {
90. for (int n=1; n<num; n++)
91. printf("%c",ch) ;
92. printf("%c",end);
93. }
94. // 用于搜索的函数
95. void search(int scores[], char (\*names)[10]) {
96. int high = listlen-1, low = 0, mid, num = 0, target;
97. printf("输入要查询的成绩：");
98. scanf("%d",&target);
99. for(mid = (low + high)/2;; mid = (low + high)/2) {
100. if (scores[high]==target) {
101. mid = high;
102. break;
103. } else if (scores[mid]==target) {
104. break;
105. } else if (low >= high || mid == high || mid == low) {
106. num = 1;
107. break;
108. } else if (scores[mid]>target) {
109. high = mid;
110. } else if(scores[mid]<target) {
111. low = mid;
112. }
113. }
114. if (num) {
115. printf("未找到相关信息！\n");
116. } else {
117. while ((mid>=low)&&(scores[mid-num]==target)) {
118. printf("姓名：%s\t成绩：%d\n",names[mid-num],target);
119. num++;
120. }
121. num=1;
122. while ((mid<=high)&&(scores[mid+num]==target)) {
123. printf("姓名：%s\t成绩：%d\n",names[mid+num],target);
124. num++;
125. }
126. }
127. }

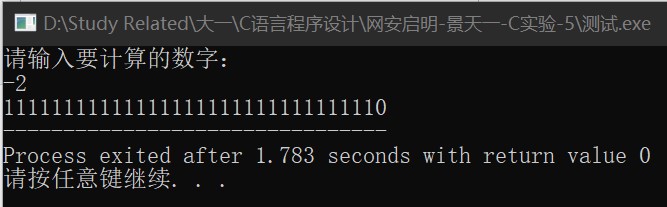
3）测试

（a） 测试数据如表5-2所示：

表5-2 编程题2，3的测试数据

|  |  |
| --- | --- |
| 测试用例 | 输入数据 |
| 用例1 | 啊啊啊 90  不不不 89  冲冲冲 94 |
| 用例2 | 啊啊啊 90  蛤蛤蛤 90  大大大 87  冲冲冲 96 |

（b） 对应测试测试用例1的运行结果如图5-8，5-9，5-10所示。



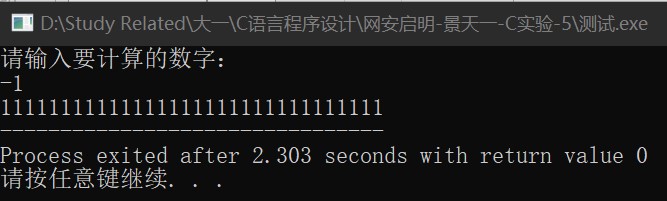


图5-8，5-9，5-10 程序设计题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图5-11，5-12，5-13所示。

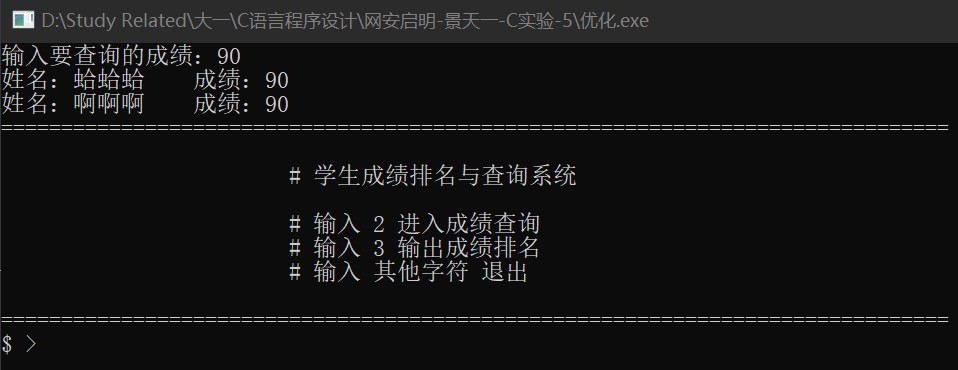
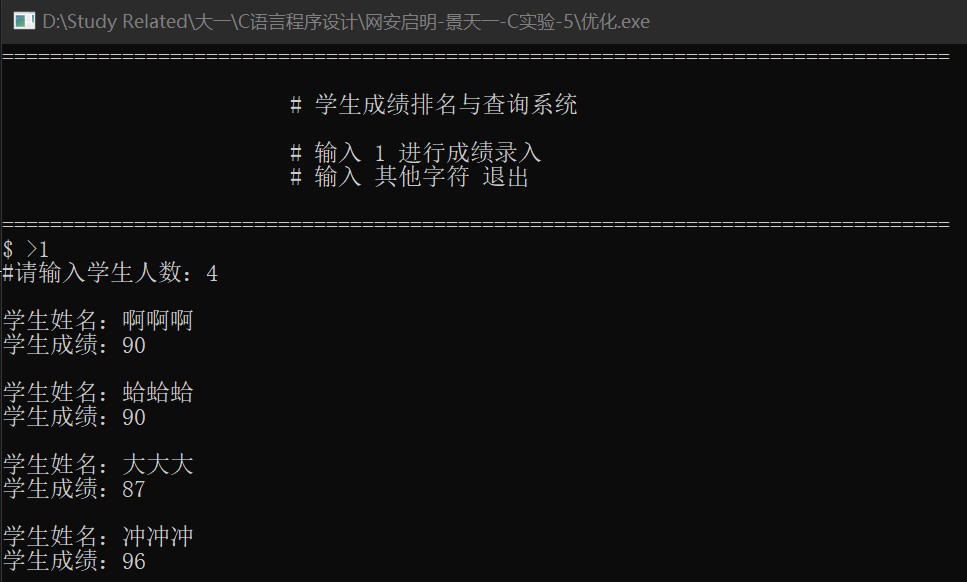


图5-11，5-12 程序设计题2的测试用例二的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**5.2.4 程序设计选做题**

编写并上机调试运行能实现以下功能的函数和程序。

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

**解答：**

1. 设计思路：
2. 在strins函数中新建一个长度为字符串s和t字符长度加1的数组用于返回数据；
3. 每次读取s字符数组的位置，当位置i小等于n时，依次输入s数组的字符；
4. 当位置i等于n小于n+ t 数组长度时，依次输入t数组字符；
5. 当位置i大于n + t数组长度时，继续输入s数组字符；
6. 为最后一位添加终止符’\0’；
7. 源代码清单：
8. #include<stdio.h>
9. #include<string.h>
10. void strnins(char [], char [], int );
11. int main(void) {
12. char s[]={'1','2','3','7','8','9'}, t[]={'4','5','6'};
13. int n=3;
14. strnins(s, t, n);
15. return 0;
16. }
17. void strnins(char s[], char t[], int n) {
18. char return\_list[strlen(s) + strlen(t) + 1];
19. for (int i = 0; i <= strlen(return\_list); i++) {
20. if (i < n) {
21. return\_list[i] = s[i];
22. } else if (i >= n && i < n + strlen(t)) {
23. return\_list[i] = t[i-n];
24. } else if (i >= n + strlen(t)) {
25. return\_list[i] = s[i-strlen(t)];
26. }
27. }
28. printf("%s\n", return\_list);
29. }
30. 测试

选做题1的运行结果如图5-13所示：



图5-13 选做题1的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 5.3 实验小结

本次实验主要练习了数组的使用，涉及数组的初始化，数组元素的赋值，作为函数参数的数组，字符数组的使用，作为函数参数的多维数组等。

其中，我对作为函数参数的多维数组的使用体会最深。在C语言中，传入函数的多维数组要求有确定的非第一唯长度，因此对多维数组的使用存在很多限制。在上网查询并询问老师后，了解到可以用一维指针或指向多维数组的指针的方式传入，引发了我对指针使用的好奇。同时，C语言对数组使用的诸多限制也要求我们寻找实现动态数组的方法。

# 6 指针实验

## 6.1 实验目的

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

## 6.2 实验题目及要求

**6.2.1 源程序改错题**

在下面所给的源程序中，函数strcopy(t, s)的功能是将字符串s复制给字符串t，并且返回串t的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:

programming↙ （键盘输入）

programming

Input a string again:

language↙ （键盘输入）

language

1. #include<stdio.h>
2. char \*strcopy(char \*, const char \*);
3. int main(void)
4. {
5. char \*s1, \*s2, \*s3;
6. printf("Input a string:\n", s2);
7. scanf("%s", s2);
8. strcopy(s1, s2);
9. printf("%s\n", s1);
10. printf("Input a string again:\n", s2);
11. scanf("%s", s2);
12. s3 = strcopy(s1, s2);
13. printf("%s\n", s3);
14. return 0;
15. }
16. /\*将字符串s复制给字符串t，并且返回串t的首地址\*/
17. char \* strcopy(char \*t, const char \*s)
18. {
19. while(\*t++ = \*s++);
20. return (t);
21. }

**解答：**

1. 逻辑错误修改：
2. 第4行声明指针变量前，需要为指针提供具体指向的地址，防止出现野指针，正确形式为：

char ch\_1[100], ch\_2[100], ch\_3[100];

1. 第20行返回的指针t指向的是复制出的字符串的末地址，需要返回首地址，故添加一个字符指针变量指向首地址作为返回值：

char \*p=t;

while(\*p++ = \*s++);

return (t); // 返回字符串首地址

1. 修改后，源程序清单如下：
2. #include<stdio.h>
3. char \*strcopy(char \*, const char \*);
4. int main(void) {
5. char ch\_1[100], ch\_2[100], ch\_3[100]; // 为指针提供具体指向的变量，防止出现野指针情况
6. char \*s1=ch\_1, \*s2=ch\_1, \*s3=ch\_3;
7. printf("Input a string:\n", s2);
8. scanf("%s", s2);
9. strcopy(s1, s2);
10. printf("%s\n", s1);
11. printf("Input a string again:\n", s2);
12. scanf("%s", s2);
13. s3 = strcopy(s1, s2);
14. printf("%s\n", s3);
15. return 0;
16. }
17. /\*将字符串s复制给字符串t，并且返回串t的首地址\*/
18. char \* strcopy(char \*t, const char \*s) {
19. char \*p=t;
20. while(\*p++ = \*s++);
21. return (t); // 返回字符串首地址
22. }
23. 错误修改后运行结果如图6-1所示。

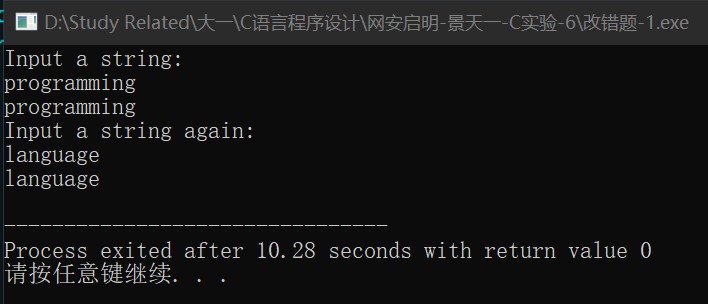


图6-1 实验6-1修改后运行结果截图

**6.2.2 源程序完善、修改替换题**

1. 下面程序中函数strsort用于对字符串进行升序排序，在主函数中输入N个字符串（字符串长度不超过49）存入通过malloc动态分配的存储空间，然后调用strsort对这N个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

1. #include<stdio.h>
2. #include<\_\_\_\_\_\_\_\_\_>
3. #include<string.h>
4. #define N 4
5. /\*对指针数组s指向的size个字符串进行升序排序\*/
6. void strsort(char \*s[], int size)
7. {
8. \_\_\_\_\_\_\_temp;
9. int i, j;
10. for(i=0; i<size-1; i++)
11. for (j=0; j<size-i-1; j++)
12. if (\_\_\_\_\_\_\_\_\_\_\_)
13. {
14. temp = s[j];
15. \_\_\_\_\_\_\_\_\_\_\_;
16. s[j+1] = temp;
17. }
18. }
19. int main()
20. {
21. int i;
22. char \*s[N], t[50];
23. for (i=0; i<N; i++)
24. {
25. gets(t);
26. s[i] = (char \*)malloc(strlen(t)+1);
27. strcpy(\_\_\_\_\_\_\_);
28. }
29. strsort(\_\_\_\_\_\_\_\_);
30. for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}
31. return 0;
32. }

②数组作为函数参数其本质类型是指针。例如，对于形参char \*s[]，编译器将其解释为char \*\*s，两种写法完全等价。请用二级指针形参重写strsort函数，并且在该函数体的任何位置都不允许使用下标引用。

**解答：**

1. 下划线处填空：
2. 第2行：#include<malloc.h>
3. 第7行：char \*temp;
4. 第11行：strcmp(s[j],s[j+1])==1
5. 第25行：strcpy(s[i], t);
6. 按要求修改后的源程序代码清单如下：
7. #include<stdio.h>
8. #include<malloc.h> //
9. #include<string.h>
10. #define N 4
11. /\*对指针数组s指向的size个字符串进行升序排序\*/
12. void strsort(char \*\*s, int size) {
13. char \*temp; //
14. int i, j;
15. for(i=0; i<size-1; i++) {
16. for (j=0; j<size-i-1; j++) {
17. if (strcmp(\*(s+j),\*(s+j+1))==1) { //
18. temp = \*(s+j);
19. \*(s+j) = \*(s+j+1);
20. \*(s+j+1) = temp;
21. }
22. }
23. }
24. }
25. int main() {
26. int i;
27. char \*s[N], t[50];
28. for (i=0; i<N; i++) {
29. gets(t);
30. s[i] = (char \*)malloc(strlen(t)+1);
31. strcpy(s[i], t); //
32. }
33. strsort(s, N);
34. for (i=0; i<N; i++) {
35. puts(s[i]);
36. free(s[i]);
37. }
38. return 0;
39. }
40. 测试用例如表6-1所示：

表6-1改错题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 测试用例1 | acdb  abcd  acbd  bcda | abcd  acbd  acdb  bcda |

填空后运行结果如图6-2所示。

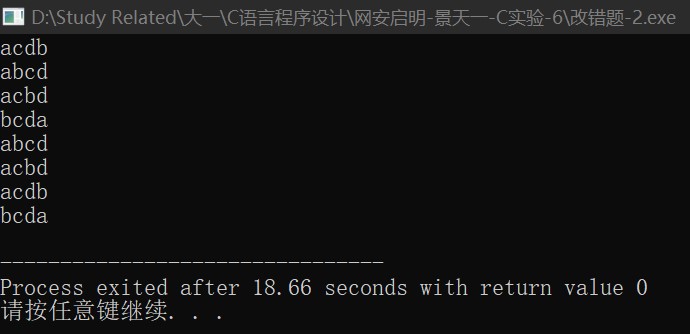


图6-2 实验6-2填空后运行结果截图

1. 下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

①请在源程序中的下划线处填写合适的代码来完善该程序，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

1. # include<stdio.h>
2. # include<string.h>
3. int main (void)
4. {
5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
6. char a[80], b[80], \*result;
7. int choice;
8. while(1)
9. {
10. do
11. {
12. printf("\t\t1 copy string.\n");
13. printf("\t\t2 connect string.\n");
14. printf("\t\t3 parse string.\n");
15. printf("\t\t4 exit.\n");
16. printf("\t\tinput a number (1-4) please.\n");
17. scanf("%d", &choice);
18. }while(choice<1 || choice>4);
19. switch(choice)
20. {
21. case 1: p = strcpy; break;
22. case 2: p = strcat; break;
23. case 3: p = strok; break;
24. case 4: goto down;
25. }
26. getchar();
27. printf("input the first string please!\n");
28. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
29. printf("input the second string please!\n");
30. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
31. result = \_\_\_\_\_\_\_\_\_\_\_(a, b);
32. printf("the result is %s\n", result);
33. }
34. down:
35. return 0;
36. }

②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

**解答：**

1. 下划线处填空：
2. 第4行：char \*(\*p)(char\*, const char \*);
3. 第31行：gets(a);
4. 第33行：gets(b);
5. 第34行：result = p(a, b);
6. 重写源程序后的代码清单如下：
7. # include<stdio.h>
8. # include<string.h>
9. int main (void) {
10. char \*(\*p)(char\*, const char \*); //
11. char a[80], b[80], \*result;
12. int choice;
13. while(1) {
14. do {
15. printf("\t\t1 copy string.\n");
16. printf("\t\t2 connect string.\n");
17. printf("\t\t3 parse string.\n");
18. printf("\t\t4 exit.\n");
19. printf("\t\tinput a number (1-4) please.\n");
20. scanf("%d", &choice);
21. } while(choice<1 || choice>4);
22. switch(choice) {
23. case 1:
24. p = strcpy;
25. break;
26. case 2:
27. p = strcat;
28. break;
29. case 3:
30. p = strtok;
31. break;
32. case 4:
33. goto down;
34. }
35. getchar();
36. printf("input the first string please!\n");
37. gets(a);
38. printf("input the second string please!\n");
39. gets(b);
40. result = p(a, b);
41. printf("the result is %s\n", result);
42. }
43. down:
44. return 0;
45. }
46. 用函数指针作为转移表改写程序后的源代码如下：
47. # include<stdio.h>
48. # include<string.h>
49. int main (void) {
50. char \*(\*p[3])(char\*, const char \*)={
51. strcpy,
52. strcat,
53. strtok,
54. }; //
55. char a[80], b[80], \*result;
56. int choice;
57. while(1) {
58. do {
59. printf("\t\t1 copy string.\n");
60. printf("\t\t2 connect string.\n");
61. printf("\t\t3 parse string.\n");
62. printf("\t\t4 exit.\n");
63. printf("\t\tinput a number (1-4) please.\n");
64. scanf("%d", &choice);
65. } while(choice<1 || choice>4);
66. if (choice == 4)
67. goto down;
68. getchar();
69. printf("input the first string please!\n");
70. gets(a);
71. printf("input the second string please!\n");
72. gets(b);
73. result = p[choice-1](a, b);
74. printf("the result is %s\n", result);
75. }
76. down:
77. return 0;
78. }
79. 测试：

源程序按照题目要求运行后的结果如图6-3所示：

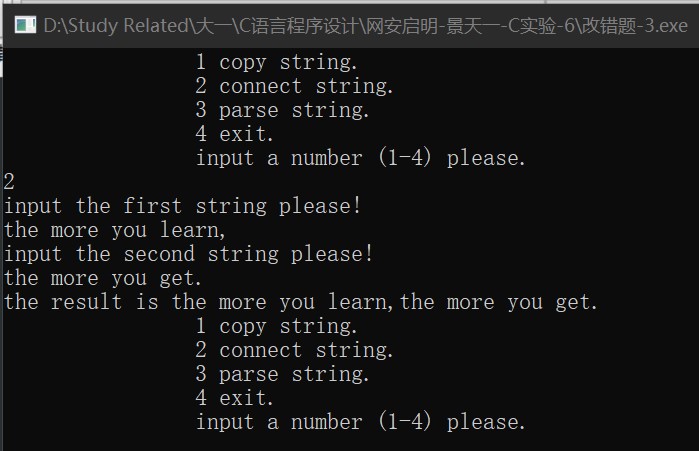


图6-3 实验6-3填空并修改后运行结果截图

**6.2.3 跟踪调试题**

请按下面的要求对源程序进行操作，并回答问题和排除错误。

（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（2）排除错误，使程序输出结果为：there is a boat on the lake.

1. #include "stdio.h"
2. char \*strcpy(char \*,char \*);
3. void main(void)
4. {
5. char a[20],b[60]="there is a boat on the lake.";
6. printf("%s\n",strcpy(a,b));
7. }
8. char \*strcpy(char \*s,char \*t)
9. {
10. while(\*s++=\*t++)
11. ;
12. return (s);
13. }

**解答：**

1. 单步执行： 进入strcpy时watch窗口中s为ox62fe00””；返回main时, watch窗口中s为0x62fe1d””；
2. 程序改错：第9行添加char \*p=s; 第10行正确形式为：while(\*p++=\*t++); 最后返回的即为s的首地址；
3. 源程序清单：
4. #include "stdio.h"
5. char \*strcpy(char \*,char \*);
6. int main(void) {
7. char a[20],b[60]="there is a boat on the lake.";
8. printf("%s\n",strcpy(a,b));
9. return 0;
10. }
11. char \*strcpy(char \*s,char \*t) {
12. char \*p=s;
13. while(\*p++=\*t++);
14. return (s);
15. }
16. 测试：

跟踪调试题1的运行结果如图6-4所示：

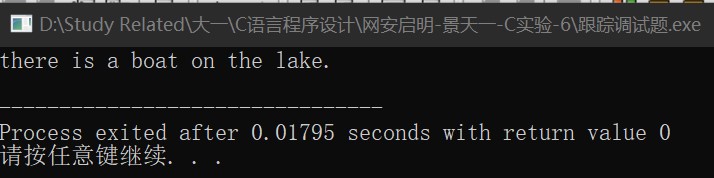


图6-4 跟踪调试题修改后运行结果截图

**6.2.4 程序设计**

1. 一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。试从该长整型变量的高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示，要求通过指针取出每字节。

**解答：**

1. 解题思路：

程序设计题-1的解题思路如图6-5所示：

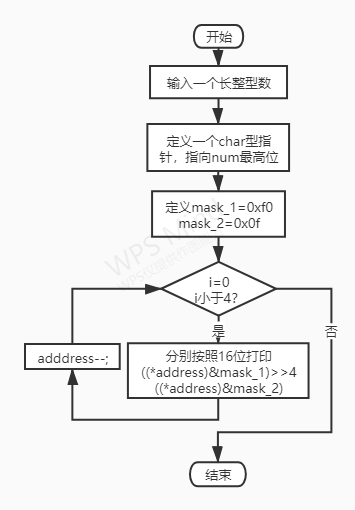


图6-5 程序设计题1的程序流程图

1. 源程序清单
2. #include<stdio.h>
3. int main(void) {
4. long int num=0x1234ABCD;
5. scanf("%ld", &num);
6. char \*address = (char\*)&num;
7. address += 3;
8. int mask\_1=0xf0, mask\_2=0x0f;
9. for (int i = 0; i < 4; i++) {
10. printf("0x%x\t", ((\*address)&mask\_1)>>4);
11. printf("0x%x\n", ((\*address)&mask\_2));
12. address--;
13. }
14. return 0;
15. }
16. 运行结果：

设计题-1的运行结果如图6-6所示：

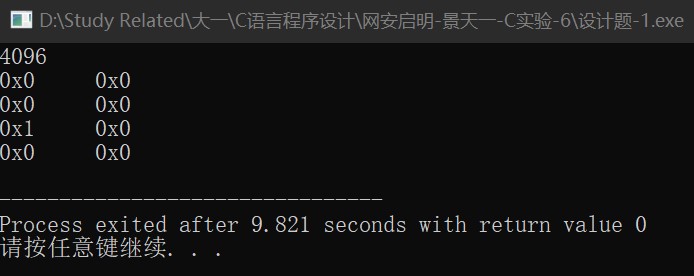


图6-6 设计题1运行结果截图

1. 旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转90°。提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像。输出原始矩阵逆时针旋转90°后的矩阵。例如，输入：

2 3

1 5 3

3 2 4

则输出：

3 4

5 2

1 3

**解答：**

1. 解题思路：

设计题-2的解题思路如图6-7所示：

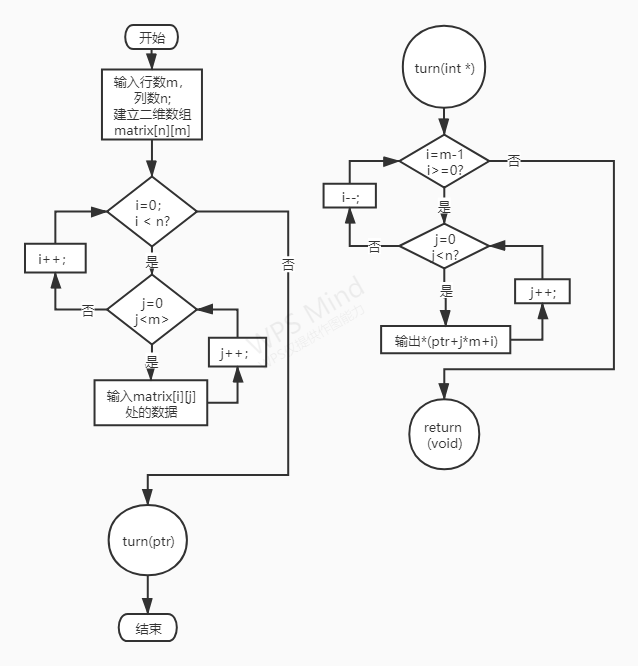


图6-7 程序设计题2的程序流程图

1. 源程序清单：
2. #include<stdio.h>
3. void turn(int \*);
4. int m, n;
5. int main(void) {
6. scanf("%d %d", &n, &m);
7. int matrix[n][m];
8. for (int i = 0; i < n; i++) {
9. for (int j = 0; j < m; j++) {
10. scanf("%d",&matrix[i][j]);
11. }
12. }
13. printf("\nInput complete!\n");
14. int \*ptr=(int\*)&matrix[0];
15. turn(ptr);
16. return 0;
17. }
18. void turn(int \*ptr) {
19. for (int i = m-1; i >= 0; i--) {
20. for (int j = 0; j < n; j++) {
21. // 希望打印matrix[j][i]
22. printf("%d ", \*(ptr + j\*m + i));
23. }
24. printf("\n");
25. }
26. }

3）测试

（a） 测试数据：

测试数据如表6-2所示。

表6-2 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 2 3  1 5 3  3 2 4 | 3 4  5 2  1 3 |
| 用例2 | 5 4  3 0 9 7  8 9 6 3  0 4 2 7  2 9 5 6  9 6 5 3 | 7 3 7 6 3  9 6 2 5 5  0 9 4 9 6  3 8 0 2 9 |
| 用例3 | 1 4  11 76 88 45 | 45  88  76  11 |

（b） 对应测试测试用例1的运行结果如图6-8所示。

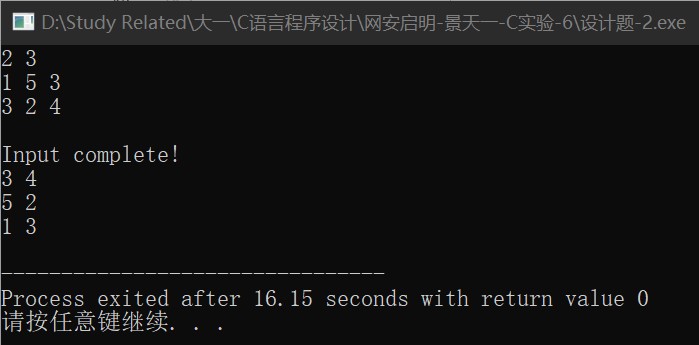


图6-8 程序设计题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图6-9所示。



图6-9 程序设计题2的测试用例二的运行结果

对应测试测试用例3的运行结果如图6-10所示。

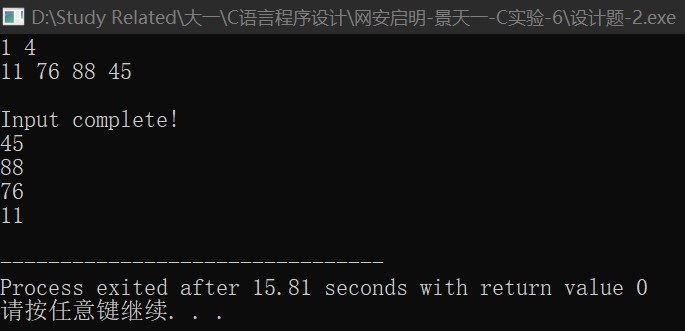


图6-10 程序设计题2的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本，且n行文本的存储无冗余，删除每一行中的前置空格（' '）和水平制表符（'\t'）。要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。

**解答：**

1. 程序设计题3的解题思路如图6-11所示：

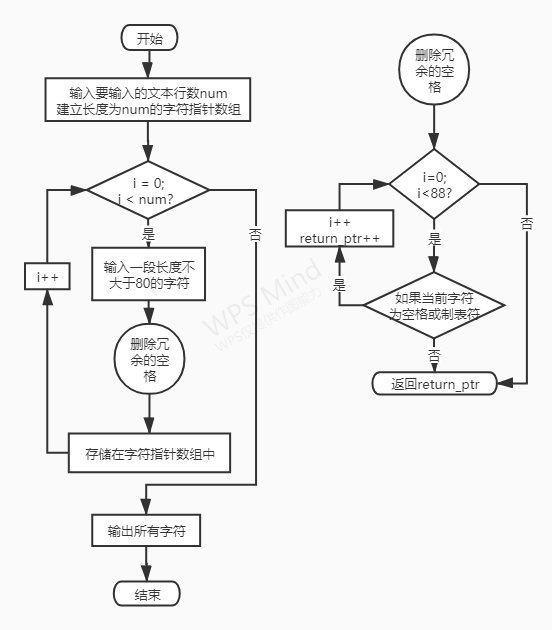


图6-11 程序设计题3的程序流程图

1. 源程序清单：
2. #include<stdio.h>
3. #include<malloc.h>
4. #include<string.h>
5. char\* delspc(char\*);
6. int main(void) {
7. int num=0;
8. scanf("%d", &num);
9. getchar();
10. char \*ch\_list[num], ch[80]={ch[79]='\0'}, \*ptr=ch;
11. for (int i=0; i<num; i++) {
12. gets(ch);
13. ptr = delspc(ptr);
14. ch\_list[i] = (char\*)malloc(strlen(ch)+1);
15. strcpy(ch\_list[i], ptr);
16. }
17. for (int i=0; i<num; i++) {
18. printf("%s\n", ch\_list[i]);
19. free(ch\_list[i]);
20. }
21. return 0;
22. }
23. char\* delspc(char \*ch) {
24. char \*return\_ptr=ch;
25. for (int i=0; i<80; i++) {
26. if (\*(ch+i)==' '||\*(ch+i)==' ') {
27. return\_ptr++;
28. } else {
29. break;
30. }
31. }
32. return return\_ptr;
33. }
34. 测试
35. 测试数据：

程序设计题3的测试数据如表6-3所示:

表6-3 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 用例 | 实际输入 | 预计输出 |
| 用例1 | 3(回车)  A a(回车)  Aaa aa(回车)  A a(回车) | A a  Aaa aa  A a |

1. 对应测试测试用例1的运行结果如图6-12所示。



图6-12 程序设计题3的测试用例一的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。

①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。

②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。

③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

例如，当输入13607122并回车，程序运行结果如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

**解答:**

1. 程序设计题4的解题思路如下所示:
2. 声明一个有8个元素的函数指针数组,依次初始化为8个任务函数;
3. 以字符串形式输入指令;
4. 声明一个函数指针数组,长度为输入的指令的长度;
5. 对输入的指令中的每个字符,依次执行1中声明的函数指针数组的元素(任务函数);
6. 结束;
7. 源程序清单:
8. #include<stdio.h>
9. #include<string.h>
10. void task0(void) {
11. printf("task0 is called!\n");
12. }
13. void task1(void) {
14. printf("task1 is called!\n");
15. }
16. void task2(void) {
17. printf("task2 is called!\n");
18. }
19. void task3(void) {
20. printf("task3 is called!\n");
21. }
22. void task4(void) {
23. printf("task4 is called!\n");
24. }
25. void task5(void) {
26. printf("task5 is called!\n");
27. }
28. void task6(void) {
29. printf("task6 is called!\n");
30. }
31. void task7(void) {
32. printf("task7 is called!\n");
33. }
34. void execute(int num, void (\*fun[])(void)) {
35. for (int i = 0; i < num; i++) {
36. fun[i]();
37. }
38. }
39. void scheduler(void) {
40. void (\*func\_list[8])(void)={
41. task0,task1,task2,task3,task4,task5,task6,task7
42. };
43. char names[9];
44. scanf("%s", &names);
45. int num = strlen(names);
46. void (\*function[num])(void);
47. for (int i = 0; i < num; i++) {
48. function[i] = func\_list[(int)names[i]-48];
49. }
50. execute(num, function);
51. }
52. int main(void) {
53. scheduler();
54. return 0;
55. }
56. 测试:
57. 测试数据:

程序设计题4的测试数据如表6-4所示:

表6-4 编程题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 实际输入 | 预计输出 |
| 用例1 | 13607122 | task1 is called!  task3 is called!  task6 is called!  task0 is called!  task7 is called!  task1 is called!  task2 is called!  task2 is called! |

1. 对应测试测试用例1的运行结果如图6-13所示。

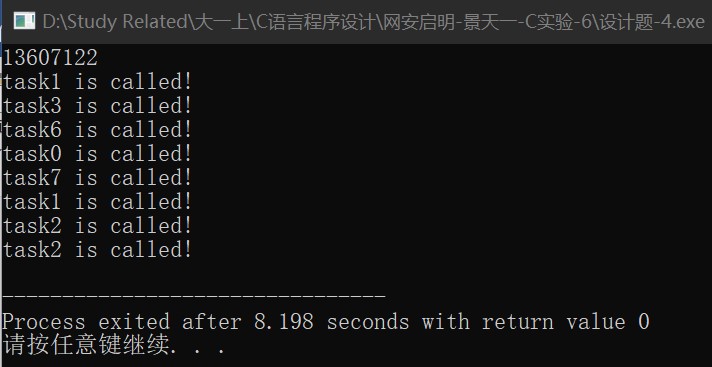


图6-13 程序设计题3的测试用例一的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**6.2.5 选做题**

1. 设有N位整数和M位小数（N=20，M=10）的数据a,b。编程计算a+b并输出结果。

如：12345678912345678912.1234567891 + 765432109876543210.0123456789

**解答:**

1. 解题思路:

选做题1的解题思路如下所示:

1. 输入需要相加的两个数字分别存储在字符串num\_1,num\_2中;
2. 声明整型变量add=0表示进位;
3. 从低到高遍历两个数字的每一位,取两数与进位add相加后除以十的余数记为该位的字符,若相加大于十,则记进位add为1;
4. 为输出的字符串的最后一位添加’\0’;
5. 源代码清单:
6. #include<stdio.h>
7. #define N 20
8. #define M 10
9. int main(void) {
10. char num\_1[M+N+2], num\_2[M+N+2], result[M+N+3];
11. printf("Input the first number: ");
12. gets(num\_1);
13. printf("Input the second number:");
14. gets(num\_2);
15. result[N+1] = '.';
16. int add = 0;
17. for (int i = M+N; i >= 0; i--) {
18. if (num\_1[i]!='.') {
19. int num\_one = num\_1[i] - '0', num\_two = num\_2[i] - '0';
20. result[i+1] = (num\_one + num\_two + add)%10 + '0';
21. if ((num\_one + num\_two+ add)>=10)
22. add = 1;
23. else
24. add = 0;
25. }
26. }
27. result[0] = add + '0';
28. if (result[0]=='0')
29. printf("%s", result+1);
30. else
31. printf("%s", result);
32. return 0;
33. }
34. 测试

选做题1的运行结果如图6-14所示:

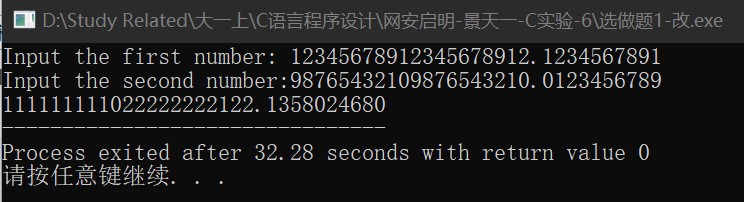


图6-14 选做题1的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 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strstr等函数名。

**解答:**

1. 解题思路:

题目中声明了一个有两个元素的函数指针数组,指向参数为两个const char型指针,返回值为char型指针的函数;

1. 源程序清单:
2. #include<stdio.h>
3. #include<string.h>
4. char \*func(const char \*ch\_1,const char \*ch\_2){
5. printf("%s\n%s\n", ch\_1, ch\_2);
6. char return\_p[30];
7. strcpy(return\_p, ch\_1);
8. return return\_p;
9. }
10. int main(void) {
11. char str1[20],str2[20]={"Hello World!"},str3[20]={" World!"};
12. char \*(\*p[2])(const char \*,const char \*)={
13. strstr,
14. func,
15. };
16. printf("%s", p[1](str2, str3));
17. return 0;
18. }
19. 测试：
20. 选做题2的运行结果如图6-15所示:

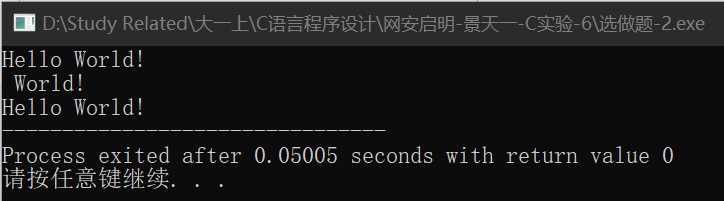


图6-15 选做题2的运行结果

1. 指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

**解答：**

1. 解题思路：

在main函数中打印输入的非文件名参数；

1. 源程序清单：
2. #include<stdio.h>
3. int main(int argc,char \*\*argv) {
4. for (int i=1; i<argc; i++) {
5. printf("%s\n",argv[i]);
6. }
7. return 0;
8. }
9. 测试：

向命令行输入参数的截图如图6-16所示：



图6-16 选做题3向命令行输入参数

选做题3的运行结果如图6-17所示：

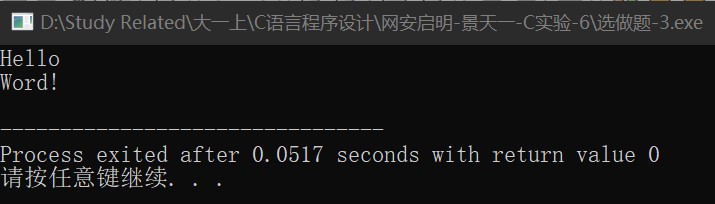


图6-17 选做题3的运行结果

## 6.3 实验小结

指针是C语言学习和使用的重难点。本次实验主要练习了指针的使用，包括指针变量的声明和引用，利用指针引用所指变量，作为函数参数和返回值的指针的使用，复杂类型指针的使用等。

指针常常被称为C语言的精华，这显示出了它的重要意义。借由指针，C语言可以对计算机内存进行直接操作，更加贴近底层，大大提高了执行效率。但同时，这也要求设计程序时格外重视对指针的使用，防止出现野指针，内存泄漏等情况。

# 7 结构与联合实验

## 7.1 实验目的

1. 通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。
2. 通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。
3. 了解字段结构和联合的用法。

## 7.2 实验内容

**7.2.1 表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,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 |  |  |

**解答：**

表达式求值的程序验证题的结果如表7-1所示：

表7-1 表达式求值的程序验证题的结果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | ‘B’ | ‘B’ |
| 3 | \*p++->t,\*p->t | ‘x’ | ‘x’ |
| 4 | \*(++p)->t | ‘x’ | ‘x’ |
| 5 | \*++p->t | ‘V’ | ‘V’ |
| 6 | ++\*p->t | ‘V’ | ‘V’ |

**7.2.2 源程序修改替换题**

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

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。

**解答：**

1. 存在的错误：在create\_list函数中传入结构指针无法改变其该指针作为头指针所指的位置；
2. 程序改错：
3. 第8行函数声明void create\_list(struct s\_list \*headp, int \*p);，改为使用指向结构体的指针的指针，正确形式为：

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

1. 第13行create\_list(head, s);，改为向create\_list函数传入head的地址，正确形式为：

create\_list(&head, s);

1. 第23行函数定义void create\_list(struct s\_list \*headp, int \*p)改为使用指向结构体指针的指针，正确形式为：

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

1. 第43行headp=loc\_head;，无法使headp指向新创建的链表的头指针，改为headp所指的结构指针指向loc\_head，正确形式为：

\*headp=loc\_head;

1. 调试与修改后的源代码清单：
2. #include "stdio.h"
3. #include "stdlib.h"
4. struct s\_list
5. {
6. int data; /\* 数据域 \*/
7. struct s\_list \*next; /\* 指针域 \*/
8. };
9. void create\_list(struct s\_list \*\*headp, int \*p); // 改为指向结构指针的指针
10. int main(void)
11. {
12. struct s\_list \*head = NULL, \*p;
13. int s[] = {1, 2, 3, 4, 5, 6, 7, 8, 0}; /\* 0为结束标记 \*/
14. create\_list(&head, s); /\* 创建新链表 \*/ // 输入head的地址
15. p = head; /\*遍历指针p指向链头 \*/
16. while (p)
17. {
18. printf("%d\t", p->data); /\* 输出数据域的值 \*/
19. p = p->next; /\*遍历指针p指向下一结点 \*/
20. }
21. printf("\n");
22. return 0;
23. }
24. void create\_list(struct s\_list \*\*headp, int \*p) // 改为指向结构指针的指针
25. {
26. struct s\_list \*loc\_head = NULL, \*tail;
27. if (p[0] == 0) /\* 相当于\*p==0 \*/
28. ;
29. else
30. {
31. /\* loc\_head指向动态分配的第一个结点 \*/
32. loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list));
33. loc\_head->data = \*p++; /\* 对数据域赋值 \*/
34. tail = loc\_head; /\* tail指向第一个结点 \*/
35. while (\*p)
36. {
37. /\* tail所指结点的指针域指向动态创建的结点 \*/
38. tail->next = (struct s\_list \*)malloc(sizeof(struct s\_list));
39. tail = tail->next; /\* tail指向新创建的结点 \*/
40. tail->data = \*p++; /\* 向新创建的结点的数据域赋值 \*/
41. }
42. tail->next = NULL; /\* 对指针域赋NULL值 \*/
43. }
44. \*headp = loc\_head; /\* 使头指针headp指向新创建的链表 \*/ // 使headp所指的结构指针指向loc\_head
45. }
46. 修改替换题优化和替换后运行结果如图7-1所示。

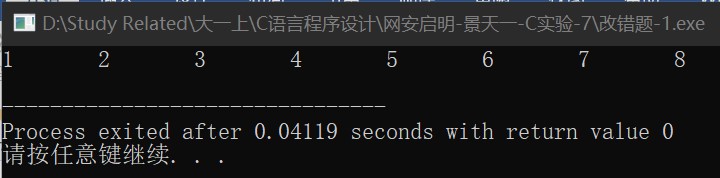


图7-1 源程序修改替换题优化替换后运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 修改替换create\_list函数后的源代码清单：
2. #include "stdio.h"
3. #include "stdlib.h"
4. struct s\_list
5. {
6. int data; /\* 数据域 \*/
7. struct s\_list \*next; /\* 指针域 \*/
8. };
9. void create\_list(struct s\_list \*\*headp, int \*p); // 改为指向结构指针的指针
10. int main(void)
11. {
12. struct s\_list \*head = NULL, \*p;
13. int s[] = {1, 2, 3, 4, 5, 6, 7, 8, 0}; /\* 0为结束标记 \*/
14. create\_list(&head, s); /\* 创建新链表 \*/ // 输入head的地址
15. p = head; /\*遍历指针p指向链头 \*/
16. while (p)
17. {
18. printf("%d\t", p->data); /\* 输出数据域的值 \*/
19. p = p->next; /\*遍历指针p指向下一结点 \*/
20. }
21. printf("\n");
22. return 0;
23. }
24. void create\_list(struct s\_list \*\*headp, int \*p) // 改为指向结构指针的指针
25. {
26. struct s\_list \*tail = NULL, \*loc\_head;
27. if (p[0] == 0)
28. ;
29. else
30. {
31. tail = (struct s\_list \*)malloc(sizeof(struct s\_list));
32. tail->data = \*p++;
33. tail->next = NULL;
34. while (\*p)
35. {
36. loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list));
37. loc\_head->data = \*p++;
38. loc\_head->next = tail;
39. tail = loc\_head;
40. }
41. }
42. \*headp = loc\_head;
43. }
44. 修改create\_list函数后的运行结果如图7-2所示：

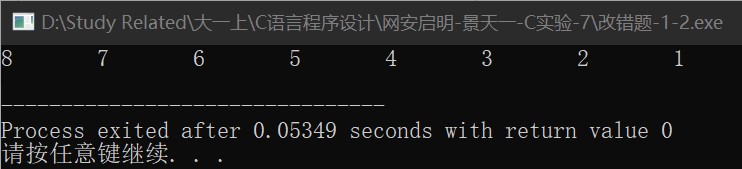


图7-2 create\_list函数修改后后运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**7.2.3 程序设计**

1. 设计一个字段结构struct bits，它将一个8位无符号字节从最低位向最高位声明为8个字段，各字段依次为bit0, bit1, …, bit7，且bit0的优先级最高。同时设计8个函数，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( int n)

{

printf(“the function %d is called!\n”,n);

}

**解答：**

1. 程序设计题1的设计思路如下所示：
2. 声明一个函数指针数组p\_fun，分别指向f0到f7 八个函数；
3. 声明一个长度为一个字节的字段类型bit，声明一个bit类型变量b；
4. 输入指令，存储在字符串order中；
5. 依次将order中的数据存储在b的每一位中；
6. 遍历b的每一位，若第i位不为0，则执行p\_fun[i](i)；
7. 源程序清单：
8. #include <stdio.h>
9. struct bits {
10. int bit0 : 1;
11. int bit1 : 1;
12. int bit2 : 1;
13. int bit3 : 1;
14. int bit4 : 1;
15. int bit5 : 1;
16. int bit6 : 1;
17. int bit7 : 1;
18. } b;
19. void f0(int n) {
20. printf("The function %d is called!\n", n);
21. };
22. void f1(int n) {
23. printf("The function %d is called!\n", n);
24. };
25. void f2(int n) {
26. printf("The function %d is called!\n", n);
27. };
28. void f3(int n) {
29. printf("The function %d is called!\n", n);
30. };
31. void f4(int n) {
32. printf("The function %d is called!\n", n);
33. };
34. void f5(int n) {
35. printf("The function %d is called!\n", n);
36. };
37. void f6(int n) {
38. printf("The function %d is called!\n", n);
39. };
40. void f7(int n) {
41. printf("The function %d is called!\n", n);
42. };
43. int main(void) {
44. void (\*p\_fun[8])(int) = {
45. f0, f1, f2, f3, f4, f5 ,f6, f7,
46. };
47. char order[9];
48. printf("Input your order:");
49. scanf("%s", order);
50. b.bit0 = order[0]-'0';
51. b.bit1 = order[1]-'0';
52. b.bit2 = order[2]-'0';
53. b.bit3 = order[3]-'0';
54. b.bit4 = order[4]-'0';
55. b.bit5 = order[5]-'0';
56. b.bit6 = order[6]-'0';
57. b.bit7 = order[7]-'0';
58. if (b.bit0) p\_fun[0](0);
59. if (b.bit1) p\_fun[1](1);
60. if (b.bit2) p\_fun[2](2);
61. if (b.bit3) p\_fun[3](3);
62. if (b.bit4) p\_fun[4](4);
63. if (b.bit5) p\_fun[5](5);
64. if (b.bit6) p\_fun[6](6);
65. if (b.bit7) p\_fun[7](7);
66. return 0;
67. }
68. 测试：
69. 测试数据：

程序设计题1的测试数据如表7-2所示：

表7-2 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入 | 预计输出 |
| 用例1 | 10010111 | The function 0 is called!  The function 3 is called!  The function 5 is called!  The function 6 is called!  The function 7 is called! |
| 用例2 | 00111001 | The function 2 is called!  The function 3 is called!  The function 4 is called!  The function 7 is called! |

1. 对应测试数据的运行结果截图：

测试用例一对应的运行结果如图7-3所示：

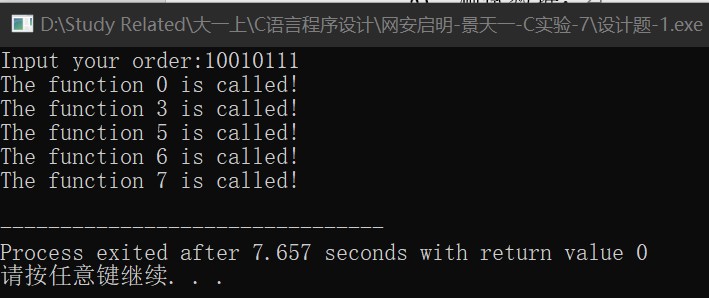


图7-3 程序设计题1的测试用例一的运行结果

测试用例二对应的运行结果如图7-4所示：

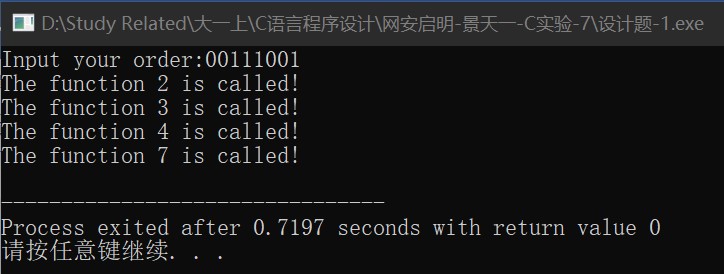


图7-4 程序设计题1的测试用例二的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

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

**解答：**

1. 解题思路：

程序设计题2的解题思路如图7-5所示：

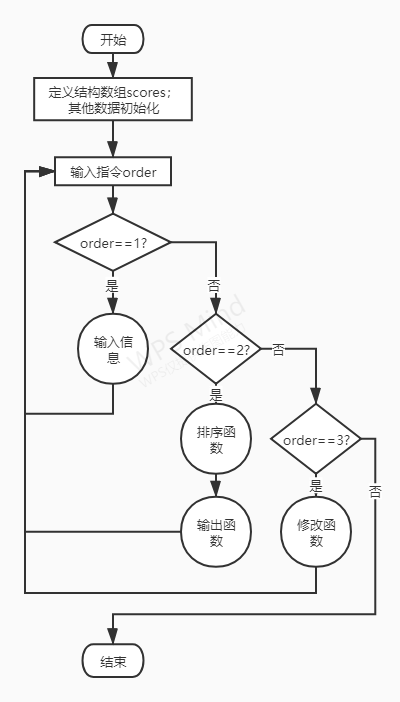


图7-5 程序设计题2的程序流程图

1. 源程序清单：
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #define SWAP(a,b,c) \
6. temp->c = a->c; \
7. a->c = b->c; \
8. b->c = temp->c;
9. // 外部变量声明
10. struct scores
11. {
12. struct scores \*prior;
13. char std\_num[20];
14. char name[10];
15. int math;
16. int english;
17. int physics;
18. int cpl;
19. float average;
20. struct scores \*next;
21. };
22. // 函数原型声明
23. int main(void); // 主函数
24. void input(struct scores \*\*); // 输入函数
25. void output(struct scores \*); // 输出函数
26. void change\_info(struct scores \*); // 改变内容
27. struct scores \*search(struct scores \*, char \*); // 查找学生
28. // 函数定义
29. int main(void)
30. { // 主函数：目录和指令输入
31. int order;
32. int ifadd;
33. struct scores \*head = NULL;
34. struct scores \*\*tail = &head;
35. do
36. {
37. system("cls");
38. printf("\t==学生成绩信息管理系统==\n");
39. printf("\t0.输入 0 退出应用\n");
40. printf("\t1.输入 1 录入新同学成绩\n");
41. printf("\t5.输入 2 输出学生成绩和平均分\n");
42. printf("\t4.输入 3 修改指定学生的某项信息\n");
43. printf("\tUser:Admin />");
44. scanf("%d", &order);
45. ifadd = 0;
46. switch (order)
47. {
48. case 1:
49. input(tail);
50. ifadd = 1;
51. break;
52. case 2:
53. output(head);
54. break;
55. case 3:
56. change\_info(head);
57. break;
58. case 0:
59. return 0;
60. default:
61. continue;
62. }
63. if (head == NULL)
64. head = \*tail;
65. if (ifadd)
66. tail = &(\*tail)->next;
67. } while (true);
68. while ((head->next) != NULL)
69. { // 清理内存
70. head = head->next;
71. free(head->prior);
72. }
73. return 0;
74. }
75. // 输入函数：输入新学生的信息
76. void input(struct scores \*\*head)
77. {
78. struct scores \*p = (struct scores \*)malloc(sizeof(struct scores));
79. printf("\n输入学号：");
80. scanf("%s", p->std\_num);
81. printf("\n输入学生姓名：");
82. scanf("%s", p->name);
83. printf("\n数学成绩：");
84. scanf("%d", &p->math);
85. printf("\n英语成绩：");
86. scanf("%d", &p->english);
87. printf("\n物理成绩：");
88. scanf("%d", &p->physics);
89. printf("\nC语言程序设计成绩：");
90. scanf("%d", &p->cpl);
91. int score\_full = p->english + p->math + p->physics + p->cpl;
92. p->average = score\_full / 4.0;
93. p->prior = \*head;
94. p->next = NULL;
95. \*head = p;
96. }
97. // 修改函数：修改指定学生的指定信息
98. void change\_info(struct scores \*head)
99. {
100. char target[10];
101. int order;
102. int data;
103. printf("请输入需要修改的学生名：");
104. scanf("%s", target);
105. struct scores \*temp = search(head, target);
106. if (temp)
107. {
108. printf("\t请输入需要修改的信息项：\n");
109. printf("\t输入 1 修改数学成绩\n");
110. printf("\t输入 2 修改物理成绩\n");
111. printf("\t输入 3 修改英语成绩\n");
112. printf("\t输入 4 修改C语言程序设计成绩\n\n");
113. printf(">");
114. scanf("%d", &order);
115. switch (order)
116. {
117. case 1:
118. printf("数学成绩:\n");
119. scanf("%d", &data);
120. temp->math = data;
121. break;
122. case 2:
123. printf("物理成绩:\n");
124. scanf("%d", &data);
125. temp->physics = data;
126. break;
127. case 3:
128. printf("英语成绩:\n");
129. scanf("%d", &data);
130. temp->english = data;
131. break;
132. case 4:
133. printf("C 语言程序设计成绩:\n\n");
134. scanf("%d", &data);
135. temp->cpl = data;
136. break;
137. }
138. int score\_full = head->english + head->math + head->physics + head->cpl;
139. temp->average = score\_full / 4.0;
140. }
141. else
142. printf("Student not found\n");
143. system("pause");
144. }
145. // 搜索函数：搜索是否输入了指定学生的信息
146. struct scores \*search(struct scores \*head, char \*target)
147. {
148. while (head && strcmp(head->name, target))
149. head = head->next;
150. return head;
151. }
152. // 输出函数：输出平均分 总分 单科成绩
153. void output(struct scores \*head)
154. {
155. int score\_full = 0;
156. while (head)
157. {
158. score\_full = head->english + head->math + head->physics + head->cpl;
159. printf("学号: %s\n", head->std\_num);
160. printf("姓名: %s\n", head->name);
161. printf("数学: %d\n", head->math);
162. printf("英语: %d\n", head->english);
163. printf("物理: %d\n", head->physics);
164. printf("C语言程序设计: %d\n", head->cpl);
165. printf("总分: %d\n", score\_full);
166. printf("%s的平均分为%.2f\n\n", head->name, head->average);
167. head = head->next;
168. }
169. system("pause");
170. }
171. 测试
172. 测试数据：

设计题-2的测试数据如表7-3所示：

表7-3 编程题2的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 第一次指令 | 第二次指令 | 第三次指令 | 第四次指令 | 第五次指令 |
| 用例1 | 1  01 aaa 89 89 90 90  02 sss 78 98 90 80  03 ddd 86 96 85 97 | 2 | 3  sss  1  100 | 1 | 0 |
| 预计输出 | 无 | 输出三名学生的各科成绩，总成绩和平均成绩 | 无 | 输出三名学生的各科成绩，总成绩和平均成绩；学生sss的数学成绩改为100 | 退出 |

1. 对应测试测试用例1的运行结果如图7-6到图7-12所示所示。

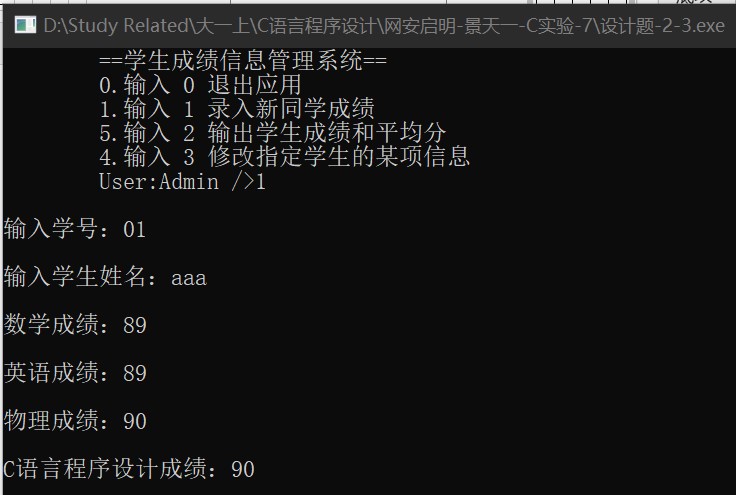


图7-6 程序设计题2的测试用例一的运行结果

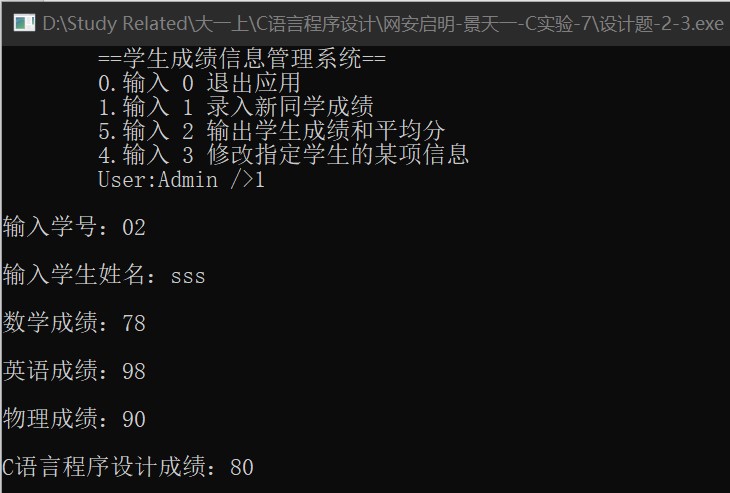


图7-7 程序设计题2的测试用例一的运行结果

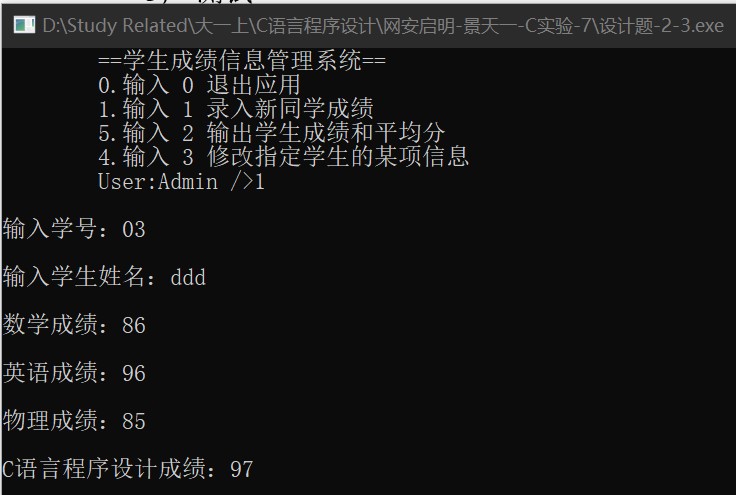


图7-8 程序设计题2的测试用例一的运行结果

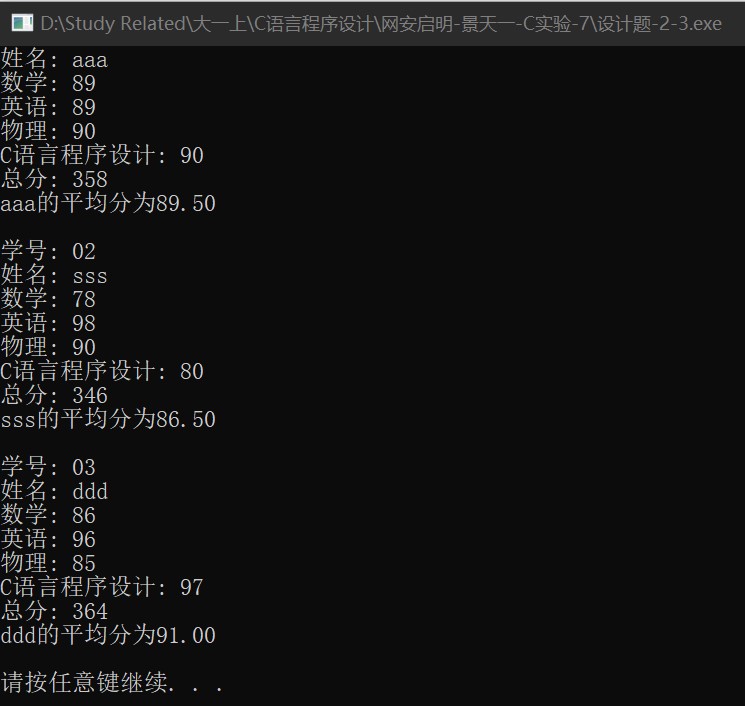


图7-9 程序设计题2的测试用例一的运行结果

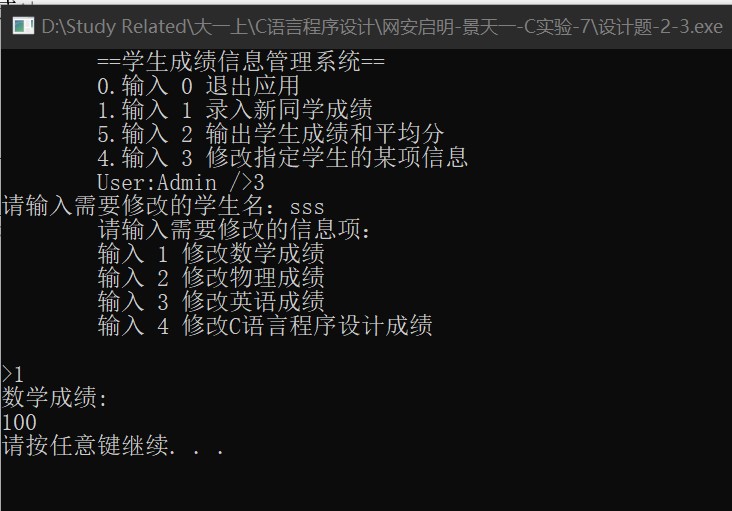


图7-10 程序设计题2的测试用例一的运行结果

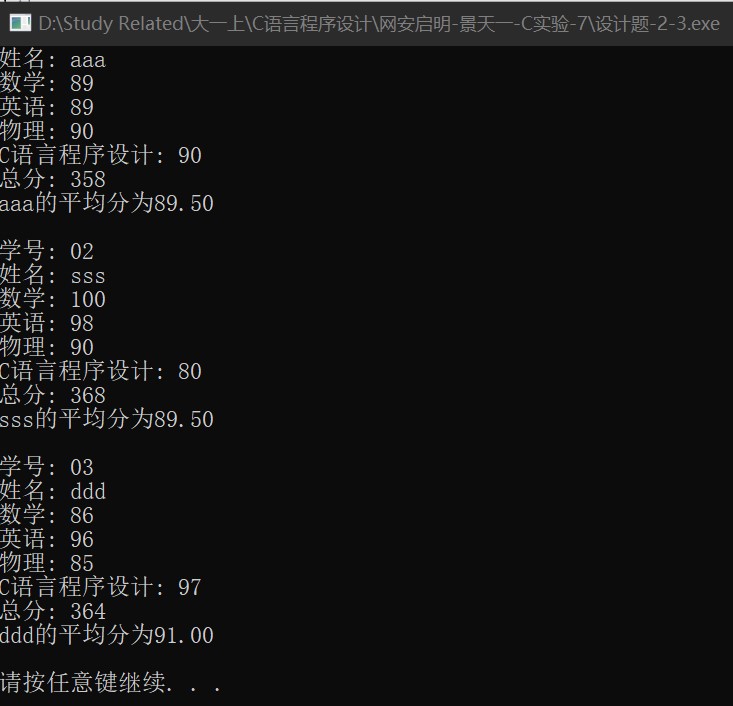


图7-11 程序设计题2的测试用例一的运行结果

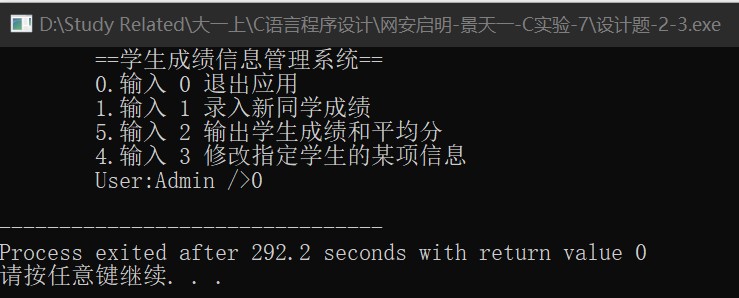


图7-12 程序设计题2的测试用例一的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 1.3 实验小结

本次实验主要学习了结构体、结构体指针的使用，字段结构的声明和使用，联合的声明和使用，链表的声明和使用。结构体是C语言中一种重要的构造类型，借助结构体，可以实现链表等多种复杂的数据类型，突破数组仅能存储相同类型数据的限制。结构体和结构指针的应用，也可以达到接近面对对象语言中“对象”的效果。

在实际运用中，我发现字段结构类型变量的使用具有一定的局限性，如：当比特数小于一字节时，无法通过键访，指针等方式进行引用，进而进行遍历操作。另外，链表在进行遍历和删除操作时，虽然效率比压缩数组高很多，但是将对记数造成很大影响，（例如排序时）必须对引入的计数变量进行更复杂的操作。

# 8 文件操作实验

## 8.1 实验目的

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

## 8.2 实验内容

**8.2.1 文件类型的程序验证题**

设有程序：

1. #include <stdio.h>
2. int main(void)
3. {
4. short a=0x253f,b=0x7b7d;
5. char ch;
6. FILE \*fp1,\*fp2;
7. fp1=fopen("d:\\abc1.bin","wb+");
8. fp2=fopen("d:\\abc2.txt","w+");
9. fwrite(&a,sizeof(short),1,fp1);
10. fwrite(&b,sizeof(short),1,fp1);
11. fprintf(fp2,"%hx %hx",a,b);
12. rewind(fp1); rewind(fp2);
13. while((ch = fgetc(fp1)) != EOF)
14. putchar(ch);
15. putchar('\n');
17. while((ch = fgetc(fp2)) != EOF)
18. putchar(ch);
19. putchar('\n');
20. fclose(fp1);
21. fclose(fp2);
22. return 0;
23. }
24. 请思考程序的输出结果，然后通过上机运行来加以验证。
25. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？
26. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

**解答：**

1. 猜测结果为：

?%}{

253f 7b7d

上机运行得到的结果如图8-1所示：

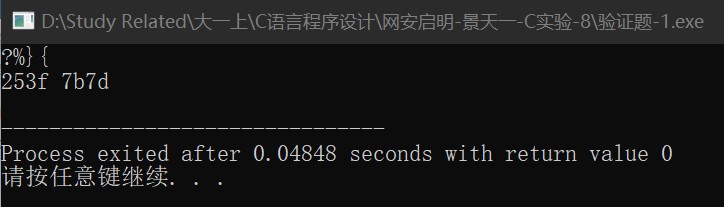


图8-1 验证题1运行结果截图

与猜测的结果一致；

1. sizeof(char)的结果是1，sizeof(short)的结果是2，因此，fwrite函数从文件中每次读取的字节数不同；sizeof(char)运行的结果如图8-2所示：

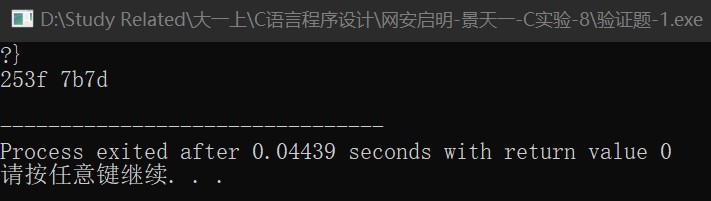


图8-2 验证题1修改后运行结果截图

1. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)后，运行结果如图8-3所示：

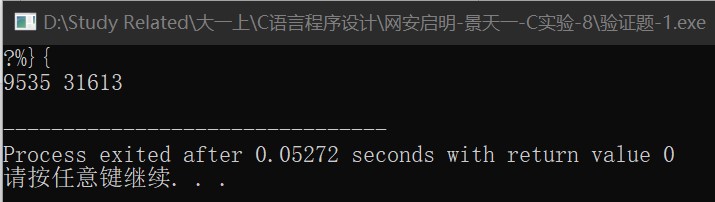


图8-3 验证题1修改后运行结果截图

可以观察到，更改后以十进制整数进行了输出。

**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。
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[])
5. {
6. char ch;
7. FILE \*fp;
8. if(argc!=2){
9. printf("Arguments error!\n");
10. exit(-1);
11. }
12. if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/
13. printf("Can't open %s file!\n",argv[1]);
14. exit(-1);
15. }
16. while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/
17. putchar(ch); /\* 向显示器中写字符 \*/
18. fclose(fp); /\* 关闭filename \*/
19. return 0;
20. }
21. 用输入输出重定向freopen改写main函数。

**解答：**

1. 程序改错：

第15行while(ch=fgetc(fp)!=EOF)中，！=的优先级高于=，因此需要在靠前的表达式外补充括号，正确行使如下：

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

1. 源代码清单：
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[]) {
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. freopen(argv[1], "r", fp);
16. while((ch=fgetc(fp))!=EOF) /\* 从filename中读字符 \*/ // 优先级错误
17. putchar(ch); /\* 向显示器中写字符 \*/
18. fclose(fp); /\* 关闭filename \*/
19. return 0;
20. }
21. 测试：

改错题1的运行结果如图8-4所示：

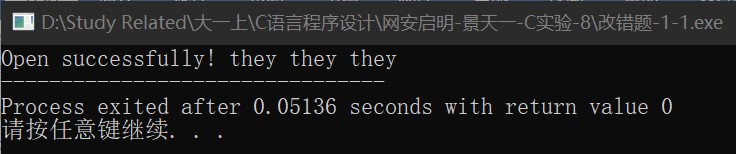


图8-4 改错题1修改后运行结果截图

1. 用freopen函数重写main函数：freopen函数的原型是：

FILE \*freopen(const char \* restrict filename, const char \* restrict mode, FILE \* restrict stream);

其作用是，使文件指针stream以mode重定向至打开的文件filename处；

源程序清单如下：

1. #include<stdio.h>
2. #include<stdlib.h>
3. int main(int argc, char\* argv[]) {
4. char ch;
5. FILE \*fp;
6. if(argc!=2) {
7. printf("Arguments error!\n");
8. exit(-1);
9. }
10. if((fp=fopen(argv[1],"r"))==NULL) { /\* fp 指向 filename \*/
11. printf("Can't open %s file!\n",argv[1]);
12. exit(-1);
13. }
14. freopen(argv[1], "r", fp);
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. }
20. 测试：

用freopen函数改写main函数后运行结果如图8-5所示：

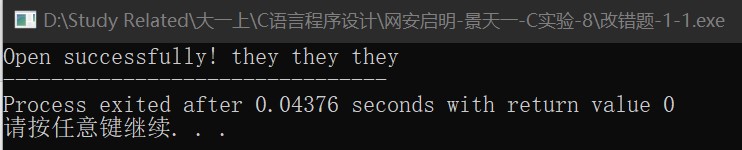


图8-5 改错题1修改后运行结果截图

**8.2.3 程序设计**

1. 编写一个程序replace，采用命令行方式，用给定的字符串替换指定文件中的目标字符串，并显示输出替换的个数。例如，命令行：

replace filename.txt you they

**解答：**

1） 算法流程如图8-6所示。

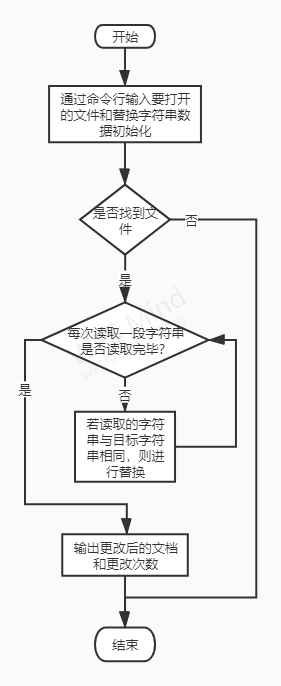


图8-6 程序设计题1的程序流程图

1. 源程序清单：
2. #include <stdio.h>
3. #include <string.h>
4. #include <stdlib.h>
5. int main(int argc, char \*argv[]) {
6. if (argc != 4) {
7. printf("Undefined input");
8. exit(0);
9. }
10. FILE \*open\_file, \*temp = fopen("temp.txt", "w+");
11. int count = 0;
12. char temp\_char[50];
13. char sep\_char;
14. if ((open\_file = fopen(argv[1], "r+")) == NULL) {
15. printf("Can't open file:%s !\n", argv[1]);
16. exit(0);
17. }
18. while (fscanf(open\_file, "%s", temp\_char) != EOF) {
19. sep\_char = fgetc(open\_file);
20. if (!strcmp(temp\_char, argv[2])) {
21. fprintf(temp, "%s%c", argv[3], sep\_char);
22. count++;
23. } else {
24. fprintf(temp, "%s%c", temp\_char, sep\_char);
25. }
26. }
27. rewind(open\_file);
28. rewind(temp);
29. while((sep\_char = fgetc(temp)) != EOF) {
30. fputc(sep\_char, open\_file);
31. }
32. printf("Replaced %d words in total", count);
33. remove("temp.txt");
34. return 0;
35. 测试

向命令行输入的参数如图8-7所示：

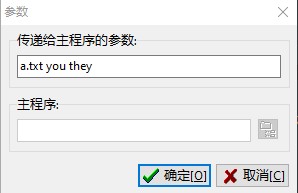


图8-6 程序设计题1向命令行传递的参数

a.txt文件中的内容为：Open successfully! they you you

程序设计题运行的结果如图8-8所示：

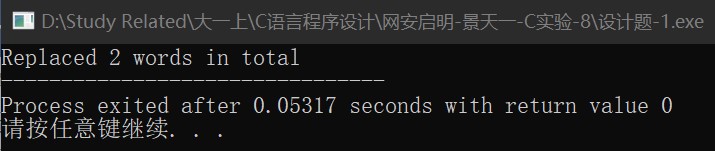


图8-8 程序设计题1的运行结果

a.txt文件中的内容为：Open successfully! they they they，与预期结果相符，证明了程序的正确性。

1. 从键盘输入10个单精度浮点数，以二进制形式存入文件float.dat中。再从文件中读出这10个单精度浮点数显示在屏幕上。之后要求将float.dat中的单精度浮点数按字节读出来，观察写入文件的浮点数字节数据是不是和计算机内存中表示的浮点数字节数据一致。

**解答：**

1. 解题思路如下所示：
2. 以wb+格式打开float.dat文件；
3. 输入10个float型数据，以二进制形式存入文件float.dat中
4. 利用fread函数将这10个数据打印出来；
5. 利用fread函数按sizeof(char)（一个字节）为单位读取float.dat中的数据；
6. 关闭文件，结束；
7. 源程序清单：
8. #include <stdio.h>
9. int main(void)
10. {
11. FILE \*open\_file = fopen("float.dat", "wb+");
12. float temp = 0;
13. for (int i = 0; i < 10; i++)
14. { printf("Input: Num%d:", i + 1);
15. scanf("%1f",&temp);
16. fwrite(&temp, sizeof(float), 1, open\_file);
17. }
18. rewind(open\_file);
19. char ch;
20. freopen("float.dat", "r", open\_file);
21. while(!feof(open\_file))
22. {
23. fread(&ch, sizeof(char), 1, open\_file);
24. printf("%d\n", ch);
25. }
26. fclose(open\_file);
27. return 0;
28. }
29. 测试：

程序设计题-2的运行结果如图8-9所示：

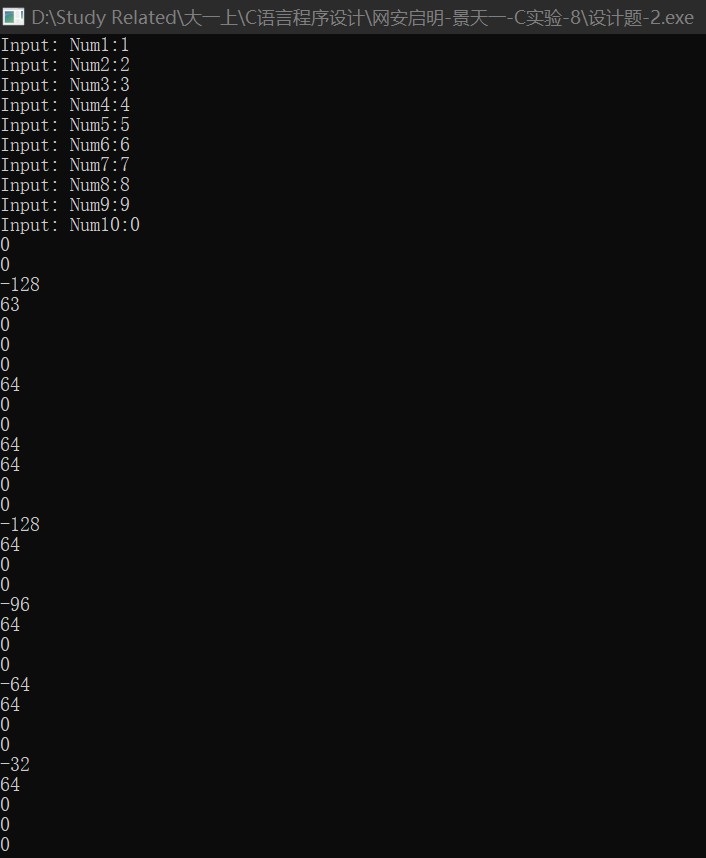


图8-9 程序设计题2的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 8.3 实验小结

本次实验主要学习了C语言中流式文件的使用和文件的存储方式。文件在磁盘中以二进制或字符形式存储，在程序中可以通过函数打开或读写，以特定的格式。格式决定了程序能对文件进行的操作。

尽管现阶段我学习的主要是对基本输入输出进行操作，但文件系统事实上至关重要。借由文件，程序可以进行更广泛的数据读取，不再局限于键盘的输入。流式文件的使用尽管复杂，但更贴近底层，固定的打开格式也在一定程度上保护率文件的读写安全。

# 9 实验总结

## 9.1 遇到的问题与收获

### 9.1.1 C语言学习中遇到的问题

初次接触C语言，我对C语言贴近底层的使用方式，相对繁琐的语法知识感到困惑。很多的函数和功能或是没有原生支持，需要自己动手编写，或是功能太弱，相比面对对象的程序设计语言要复杂得多。

学习标准输入输出时，我因繁杂的格式二晕头转向，翻看了很多次课本和资料才勉强吃透，直到实验后期多次使用和练习后才有了稍稍深入一点的理解。

在学习数组时，我对多维数组的声明和作为函数参数传入函数时的要求感到困惑，后来阅读了关于函数引用时对形参的内存分配后明白了原理。客观的说，我对指针的学习相对顺畅，但在引用方面依然产生了很大的问题，搞混了双重指针和指针在作为函数参数传入时的不同，使得当时的几次实验收到了较大的阻碍。

在编写涉及界面和实际应用的题目中，我尽可能追求界面的美观简洁，但现在还是未能在界面的简洁性和代码的简洁性之间达到良好的平衡。

### 9.1.2 C语言学习中的收获

在经过了一学期的C语言学习之后，很难说我对C语言有深入的了解，只能算最基础的入门。但是我需要承认，现在C语言是我使用最广泛的语言，以前用python完成的脚本都被C语言编写的所替代。

优雅的C语言代码应该简洁而高效，配合注释应当能够让所有阅读者理解，良好的注释是成功的一半。

工具的使用至关重要，各种IDE都具有强大的代码补全，代码风格整理和调试功能，合理的利用这些功能对编写良好的代码，提高编码效率至关重要。

算法是程序的核心组成部分，高效的算法能极大地提高程序地效率；而数据结构则为实现更多功能提供了基础，只有二者完美结合，才能编写出真正的“好程序”

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] 卢萍,李开,王多强等. C语言程序设计典型题解与实验指导, 北京：清华大学出版社, 2019

[3] Steve Summit. 你必须知道的495个C语言问题，人民邮电出版社，2017