

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程2003班**

**学 号： U202010783**

**姓 名： 刘铭宸**

**指导教师： 唐赫**

**报告日期： 2020.12.26**

**软件工程**

**目 录**

[**1 表达式和标准输入与输出实验 2**](#_Toc91425364)

[1.1 实验目的 2](#_Toc91425365)

[1.2 实验内容 2](#_Toc91425366)

[1.3 实验小结 10](#_Toc91425367)

[**2 流程控制实验 11**](#_Toc91425368)

[2.1 实验目的 11](#_Toc91425369)

[2.2 实验内容 11](#_Toc91425370)

[2.3 实验小结 24](#_Toc91425371)

[**3 函数与程序结构试验 25**](#_Toc91425372)

[3.1 实验目的 25](#_Toc91425373)

[3.2 实验内容 25](#_Toc91425374)

[3.3 实验小结 42](#_Toc91425375)

[**4 编译预处理实验 43**](#_Toc91425376)

[4.1 实验目的 43](#_Toc91425377)

[4.2 实验内容 43](#_Toc91425378)

[4.3 实验小结 55](#_Toc91425379)

[**5 数组试验 57**](#_Toc91425380)

[5.1 实验目的 57](#_Toc91425381)

[5.2 实验内容 57](#_Toc91425382)

[5.3 实验小结 72](#_Toc91425383)

[**6 指针试验 73**](#_Toc91425384)

[6.1 实验目的 73](#_Toc91425385)

[6.2 实验内容 73](#_Toc91425386)

[6.3 实验小结 95](#_Toc91425387)

[**7 结构与联合试验 96**](#_Toc91425388)

[7.1 实验目的 96](#_Toc91425389)

[7.2 实验内容 96](#_Toc91425390)

[7.3 实验小结 118](#_Toc91425391)

[**8 文件操作实验 119**](#_Toc91425392)

[8.1 实验目的 119](#_Toc91425393)

[8.2 实验内容 119](#_Toc91425394)

[8.3 实验小结 127](#_Toc91425395)

[**参考文献 128**](#_Toc91425396)

# 1 表达式和标准输入与输出实验

## 1.1 实验目的

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

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

（3）掌握简单C程序（顺序结构程序）的编写方法。

## 1.2 实验内容

**1.2.1 源程序改错**

下面给出了一个简单C语言程序例程，用来完成以下工作：

（1）输入华氏温度f，将它转换成摄氏温度c后输出；

（2）输入圆的半径值ｒ，计算并输出圆的面积ｓ；

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

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

1 #include<stdio.h>

2 #define PI 3.14159;

3 voidmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

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

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

13 /\* for task 2 \*/

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

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

16 s = PI \* r \* r;

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

18 /\* for task 3 \*/

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

20 scanf("%x %x", &k, &p );

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

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

}

**解答：**

（1）错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.14159

2) 第3行的voidmain( void )void后面没有空格，正确形式为：

Void main( void )

3) 第6行的p和k不能定义为short类型，为保证右移补0，正确形式为：

unsigned short p, k;

4) 第10行的scanf函数f前应该有&，正确形式为：

scanf(" %d", &f);

5) 第11行的5/9默认为int的整除，结果为0错误，正确形式为：

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

6) 第15行的%f是浮点数类型，应是double类型，正确形式为：

scanf("%lf", &r);

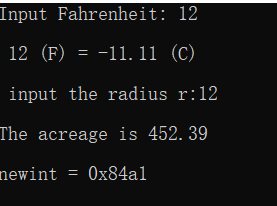
7) 第17行printf函数s前面不应该有&，正确形式为：

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

8) 第21行newint变量未被定义且不应该左移8位，正确形式为：

unsigned short newint = p & 0xff00 | k >> 8;

（2）错误修改后运行结果：



**1.2.2 源程序修改替换**

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

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

t=a ；a=b；b=t；

prinf(“\na=%d,b=%d”,a,b);

}

**解答：**

通过数值的运算可以实现两个数没有第三个变量的交换：先令a=a-b，求出a与b之间的差值存放在a中；再使b=a+b，求出原来的a存入b中；最后令a=b-a，从而借助原来的a和a，b间的差值求出b存入a中。替换后的程序如下所示：

#include<stdio.h>

void main( )

{

int a, b;

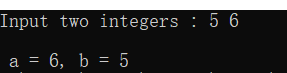
printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

a=a-b ；b=a+b；a=b-a；

prinf(“\na=%d,b=%d”,a,b);

}



**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程序结束。

**解答：**

1） 算法流程如图1.1所示。

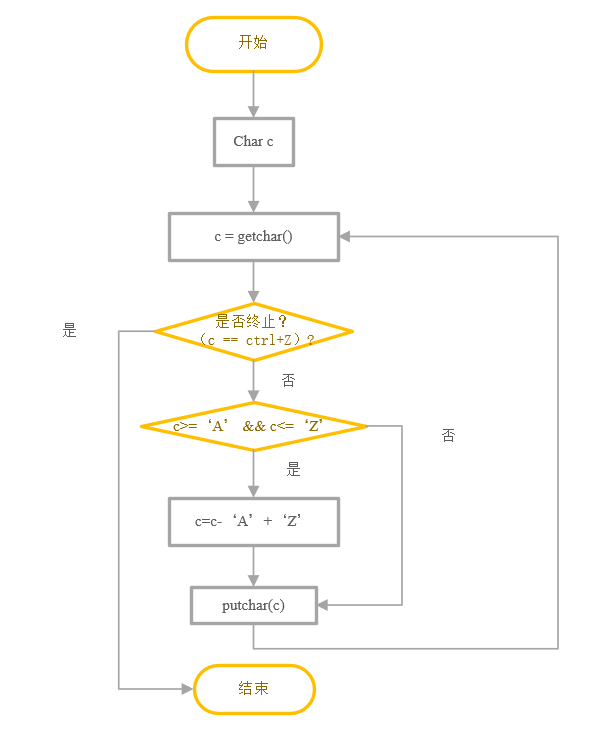


图1-1 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

int main(void)

{

char c;

while ((c = getchar())!= EOF)

{

if (c >= 'A' && c <= 'Z')c = c + 'a' - 'A';

putchar(c);

}

return 0;

}

3）测试

（a） 测试数据：

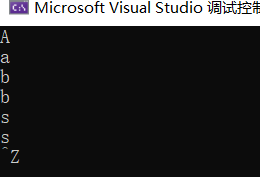
A

b

s

Ctrl+Z

（b） 对应测试数据的运行结果截图



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

**解答：**

1) 解题思路：

1.输入x，m，n，为了方便分析测试结果，x的输入采用16进制

2.如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，转2.1，否则转3.

2.1 首先x>>m，将要处理的n位移动到最右；

2.2 再将上一步的结果左移16-n位，即： (x>>m)<<(16-n)

2.3 用16进制输出结果并转4.

3. 显示输入错误信息；

4. 结束

2）程序清单

#include <stdio.h>

int main(void) {

unsigned short x, m, n;

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

if (m >= 0 && m <= 15 && n >= 1 && n <= 16 - m) printf("%hx", (x>>m)<<(16-n));

else printf("error");

return 0;

}

3）测试

（a） 测试数据：

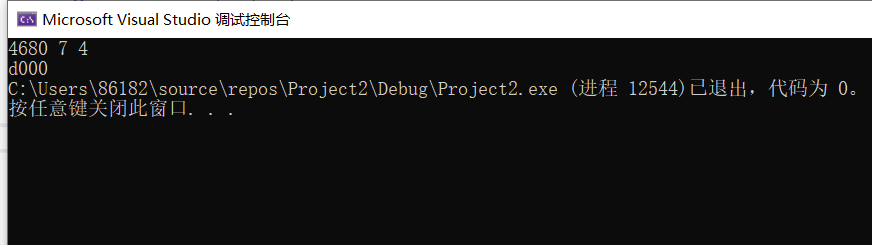
根据题目要求以及各种可能出现的错误情况构造数据如表1-1所示。

表1-1 编程题3的测试数据

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

（b） 对应测试测试用例1的运行结果如图1-2所示。

图1-2 编程题3的测试用例一的运行结果



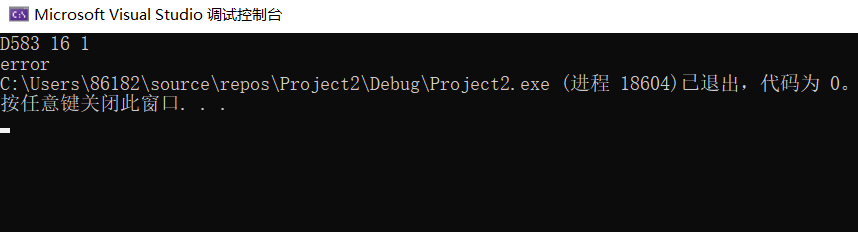
对应测试测试用例2的运行结果如图1-3所示。

图1-3 编程题3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1-4所示。

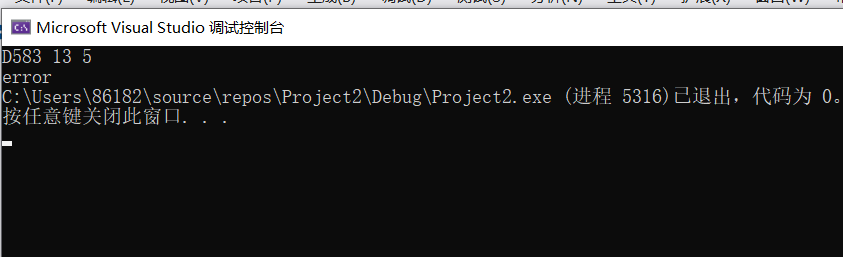


图1-4 编程题3的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（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：

1.输入x，如果输入Ctrl+Z，跳转至4结束。

2.定义i= 3;

3.如果i>0执行以下步骤

3.1 首先将要处理的8位移动到最后，x >> (8 \* i)

3.2 然后取出这8位并输出，printf("%d.", x >> (8 \* i) & 0xff);

3.3 i = i -1

4. 输出最后8位，printf("%d\n", x & 0xff);

5. 结束。

2）程序清单

#include <stdio.h>

int main(void) {

unsigned long x;

while (scanf("%lu", &x) != EOF) {

for (int i = 3; i > 0; i--) printf("%d.", x >> (8 \* i) & 0xff);

printf("%d\n", x & 0xff);

}

return 0;

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-2所示。

表1-2 编程题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| X |
| 用例1 | 3232235876 | 192.168.1.100 | 192.168.1.100 |
| 用例2 | 66322345376 | 113.30.173.160 | 113.30.173.160 |
| 用例3 | 3232235876  66322345376  187232376 | 192.168.1.100  113.30.173.160  11.40.240.120 | 192.168.1.100  113.30.173.160  11.40.240.120 |

（b） 对应测试测试用例1的运行结果如图1-5所示。

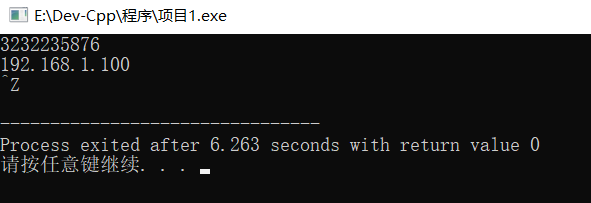


图1-5 编程题4的测试用例一的运行结果

对应测试测试用例2的运行结果如图1-6所示。

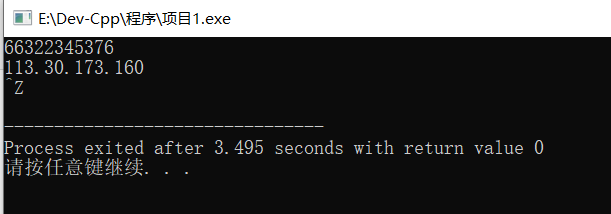


图1-6 编程题4的测试用例二的运行结果

对应测试测试用例3的运行结果如图1-7所示。

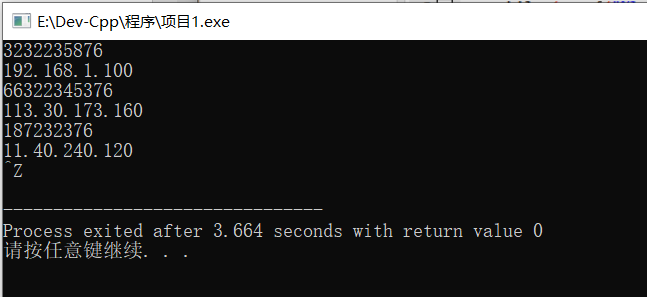


图1-7 编程题4的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 1.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验1源程序改错中，不知道如何再vs中进行单步调试，在同学和老师的指导与帮助下学会了单步调试的方法。

在编程题4中，发现用vs编译器需要多次输入Ctrl+Z才能退出程序，在百度后发现是编译器本身的设定不同，在使用了dev C++后成功解决了这个问题。

在画流程图的过程中，通过百度一些基本功能学习了Visio软件。

# 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 源程序改错**

**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\*/

#include <stdio.h>

int main( )

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

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

for(i=2,k=x>>1;i<=k;i++)

if (!x%i) {

flag = 1;

break;

}

if(flag=1) printf("%d是合数", x);

else printf("%d不是合数", x);

}

return 0;

}

**解答：**

（1）错误修改：

1) 第7行的while循环内flag应该每次置为0，正确形式为：

flag = 0;

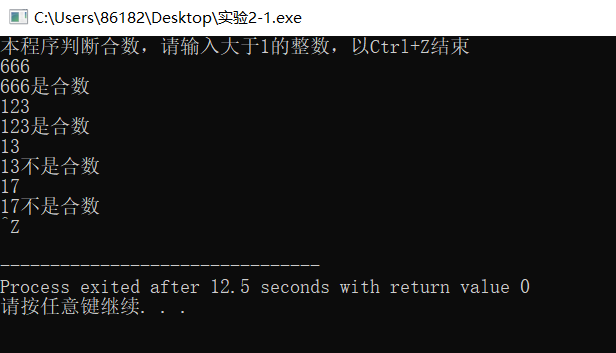
2) 第10行的!运算优先级高，产生错误运算结果，正确形式为：

if (!(x % i))

3) 第14行的判断语句应使用==，正确形式为：

if (flag == 1)

（2）错误修改后运行结果：



**2.2.2 源程序修改替换**

（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。

**解答：**

For循环中的break非结构语句破坏了单出口结构，将flag的判断放入for循环的条件控制语句，这样中途如果出现一个模数使其取模为0，即其为合数，便退出循环。

/\* 实验2-1改错题程序：合数判断器\*/

/\* 程序修改替换：改单出口 \*/

#include <stdio.h>

int main(void)

{

int i, x, k;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

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

int flag = 0;

for (i = 2, k = x >> 1; i <= k&&flag!=1; i++)

if (!(x % i)) flag=1;

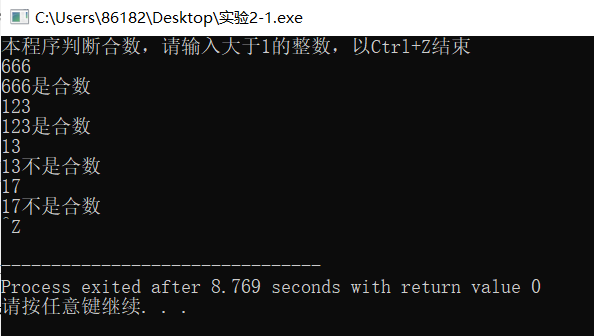
if (flag==1) printf("%d是合数\n", x);

else printf("%d不是合数\n", x);

}

return 0;

}



（2）修改实验2-1程序，将for循环改用do-while循环。

**解答：**

将for循环的判断语句放入while的判断语句，需注意do-while循环至少执行一次，所以对数字2需要特殊判断。

/\* 实验2-1改错题程序：合数判断器\*/

/\* 程序修改替换：改用dowhile循环 \*/

#include <stdio.h>

int main(void)

{

int i, x, k;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

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

int flag = 0;

k=x>>1;

i=2;

do{

if(!(x%i)){

flag=1;

break;

}

i++;

}while(i<=k);

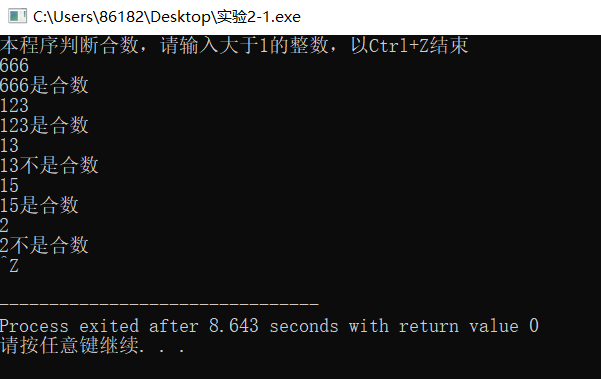
if (flag==1 && x>2) printf("%d是合数\n", x);

else printf("%d不是合数\n", x);

}

return 0;

}



（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：**

算法流程图如下：

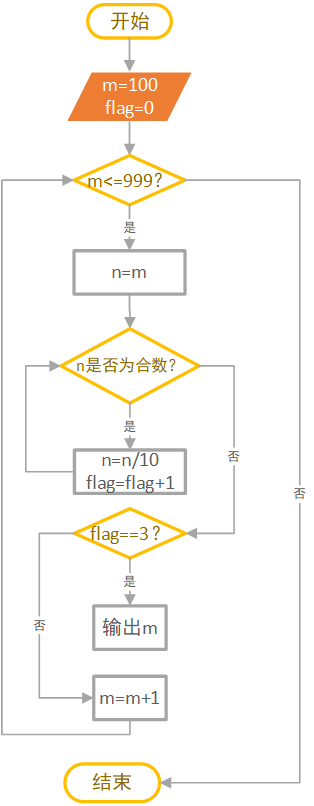


图2-1 源程序修改替换3的程序流程图

/\* 实验2-1改错题程序：合数判断器\*/

/\* 程序修改替换：纯粹合数求解器 \*/

#include <stdio.h>

int judge(int);

int main()

{

for (int m = 100; m < 1000; m++) {

int n = m, flag = 0;

while (judge(n)) {

n /= 10;

flag++;

}

if (flag == 3)printf("%d ", m);

}

return 0;

}

int judge(int x) {

int i, k, flag = 0;

for (i = 2, k = x >> 1; i <= k; i++)

if (!(x % i)) {

flag = 1;

break;

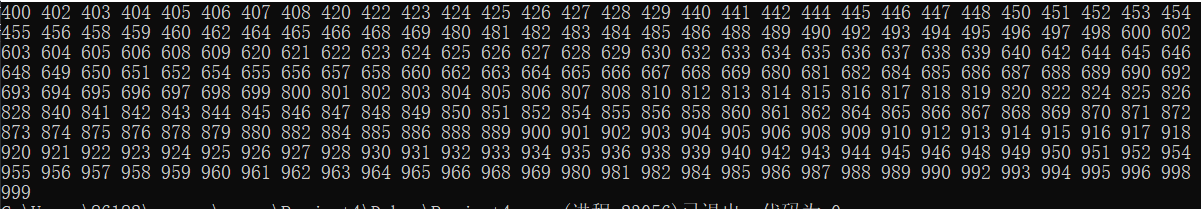
}

if (flag == 1) return 1;

else return 0;

}

程序输出展示如下：



**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；5000 ≤ x，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） 本题，可使用累加的计算方法从而避免手动计算大量的公式，简单而直接。

若使用if语句：我们便注意到：

x>=5000的部分(x-4999)应该征收25%的税金，为保证阶梯计税，x=4999

x>=4000的部分(x-3999)应该征收20%的税金，为保证阶梯计税，x=3999

x>=3000的部分(x-2999)应该征收15%的税金，为保证阶梯计税，x=2999

x>=2000的部分(x-1999)应该征收10%的税金，为保证阶梯计税，x=1999

x>=1000的部分(x-999)应该征收5%的税金，为保证阶梯计税，x=999

若使用switch语句：我们注意到case必须是整数值，则可利用int(x/1000)作为参量值，使用deflaut语句避免x>=6000，的情况，我们仍然要使用累加的方法，所以我们从大到小排列case，并不break，令其一路运算到底。

2）源程序清单

#include <stdio.h>

int main() {

int x = 0; double tax = 0;

printf("请输入工资金额：");

scanf\_s("%d", &x);

if (x >= 5000) {

tax += 0.25 \* (x - 4999);

x = 4999;

}

if (x >= 4000) {

tax += 0.2 \* (x - 3999);

x = 3999;

}

if (x >= 3000) {

tax += 0.15 \* (x - 2999);

x = 2999;

}

if (x >= 2000) {

tax += 0.1 \* (x - 1999);

x = 1999;

}

if (x >= 1000) {

tax += 0.05 \* (x - 999);

x = 999;

}

printf("应收取税金为 %lf元\n", tax);

return 0;

}

#include <stdio.h>

int main() {

int x = 0; double tax = 0;

printf("请输入工资金额：");

scanf\_s("%d", &x);

switch (x / 1000) {

default:

case 5:tax += 0.25 \* (x - 4999); x = 4999;

case 4:tax += 0.2 \* (x - 3999); x = 3999;

case 3:tax += 0.15 \* (x - 2999); x = 2999;

case 2:tax += 0.1 \* (x - 1999); x = 1999;

case 1:tax += 0.05 \* (x - 999); x = 999;

case 0: break;

}

printf("应收取税金为%lf元\n", tax);

return 0;

}}3）测试

（a） 测试数据：

500

1500

2500

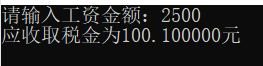
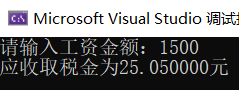
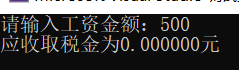
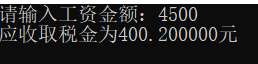
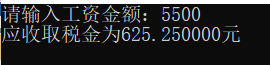
3500

4500

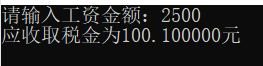
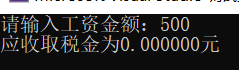
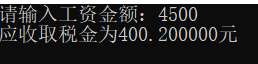
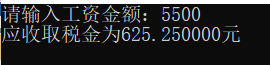
5500

（b） 对应测试数据的运行结果截图

/\*if \*/

/\* switch \*/

（2）输入一段以!结尾的短文(最多5行,每行不超过50个字符)，要求将它复制到输出，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)解题思路：

1.定义一个字符数组str用以存储读入的字符，数字i作为短文长度。

2.如果str[i-1]!=’!’，读入该字符，否则跳转3

3. 判断是否存在每行一个以上的空格，若不存在，显示输出；若存在，输出一个空格；

4. 结束

2）程序清单

#include <stdio.h>

int main(void)

{

char str[1000] = { 0 };

int i = 0, j = 0;

while (str[i - 1] != '!') {

str[i] = getchar();

i++;

}

for (j; j < i; j++) {

while (str[j] == ' ' && str[j + 1] == ' ')j++;

printf("%c", str[j]);

}

return 0;

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-1所示。

表2-1 编程题2的测试数据（以’#’表示空格符号）

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| 输入文本 |
| 用例1 | 我#爱###你  就##像#老鼠####爱  ####大米  ! | 我#爱#你  就#像#老鼠#爱  #大米  ! | 我#爱#你  就#像#老鼠#爱  #大米  ! |
| 用例2 | I#like#apples! | I#like#apples! | I#like#apples! |
| 用例3 | 生活#就像####海洋，  ####Life#is#like#a######sea. ###! | 生活#就像#海洋，  #Life#is#like#a#sea. #! | 生活#就像#海洋，  #Life#is#like#a#sea. #! |

（b） 对应测试测试用例1的运行结果如图2-2所示。

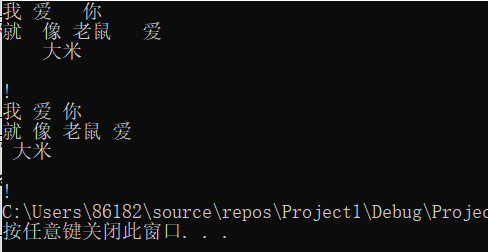


图2-2 编程题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图2-3所示。



图2-3 编程题3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2-4所示。

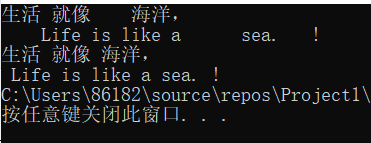


图2-4 编程题3的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（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

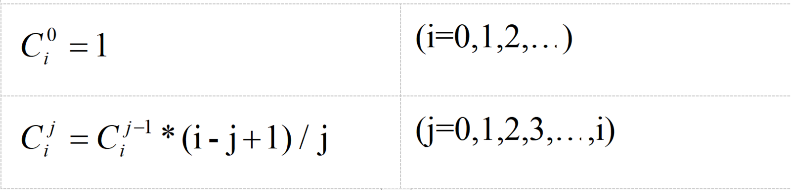
1 6 15 20 15 6 1

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C表示，C的计算如下：



根据以上公式，采用顺推法编程，输入最后一行的编号N（0<=N<=6），要求输出金字塔效果的杨辉三角形。

特别要注意空格的数目，每个数字占位4格，例如，1位数后补3个空格。第N行行首是N个空格（每向上一行，行首空格数对应增加）。每行末尾是换行符。

**解答：**

1) 解题思路：

1.输入N.

2.利用公式将要输出的数字存放在二维数组array中.

3.定义m表示当前行，在每行行首输出3N-2m个空格.

4.判断当前输出的数字是几位数

4.1若是一位数，则在输出该数字后输出三个空格；

4.2若是两位数，则在输出该数字后输出两个空格；

4.3若是三位数，则在输出该数字后输出一个空格；

5.结束

2）程序清单

#include <stdio.h>

int main(void) {

int N = 0;

scanf("%d", &N);

int array[100][100] = { 0 };

for (int i = 0; i < N + 1; i++) {

array[i][0] = 1;

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

array[i][j] = array[i][j - 1] \* (i - j + 1) / (double)j;

}

}

for (int m = 0; m < N + 1; m++) {

for (int k = 0; k < 3 \* N - 2 \* m; k++)printf(" ");

for (int n = 0; n < m + 1; n++) {

printf("%d", array[m][n]);

if (array[m][n] >= 10 && array[m][n] < 100)printf(" ");

else if (array[m][n] >= 100)printf(" ");

else printf(" ");

}

printf("\n");

}

return 0;

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-2所示。

表2-2 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 输 入 | 理 论 结 果 | 运 行 结 果 |
| N |
| 例1 | 0 | 1 | 1 |
| 例2 | 3 | 1  1 1  1 2 1  1 3 3 1 | 1  1 1  1 2 1  1 3 3 1 |
| 例3 | 6 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1  1 6 15 20 15 6 1 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1  1 6 15 20 15 6 1 |

（b） 对应测试测试用例1的运行结果如图2-5所示。



图2-5 编程题3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。

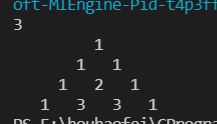


图2-6 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图2-7所示。

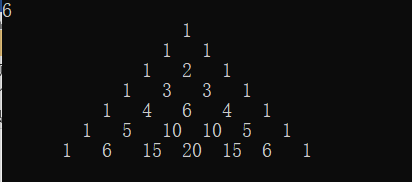


图2-7 编程题3的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（4） 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。要求这些数字从小到大排列，每个数字单独占一行。

**解答：**

1. 解题思路：算法流程可用算法流程图表示

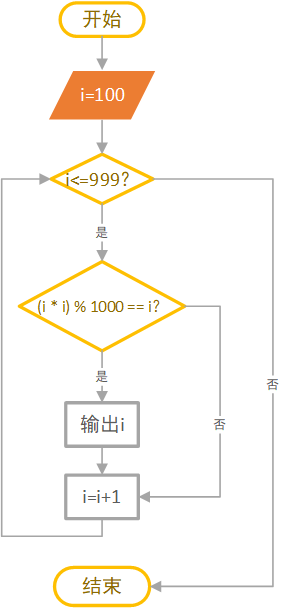


图2-8 编程题4的程序流程图

2）程序清单

#include <stdio.h>

int main(void) {

for (int i = 100; i < 1000; i++)

if ((i \* i) % 1000 == i)printf("%d\n", i);

return 0;

}

3）测试

（a） 测试数据：

由数学推导已知，这类数的个数是有限的，可演算得仅2个三位数满足条件，即376和625。

（b） 对应测试的运行结果如图2-8所示。

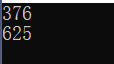


图2-9 编程题4的测试的运行结果

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

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验1源程序改错中，采用逐步调试的方法成功找出了语法上的错误，最终解决了所有改错。

在修改题1中不知道如何修改实验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。和同学交流过后，明白了单出口的含义，For循环中的break非结构语句破坏了单出口结构，将flag的判断放入for循环的条件控制语句，这样中途如果出现一个模数使其取模为0，即其为合数，便退出循环。既没有破坏单出口的性质，又保证了运算的效率。

在编程题3中，学会了仔细阅读题面然后逐渐分析得出空格的个数解决了问题，明白了学会审题的重要意义。

# 3 函数与程序结构试验

## 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 源程序改错**

**1．程序改错**

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

1 /\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

2 #include <stdio.h>

3 int main(void)

4 {

5 int k;

6 for(k=1;k<=20;k++)

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

8 return 0;

9 }

10long sum\_fac(int n)

11{

12 long s=0;

13 int i,fac;

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

15 fac\*=i;

16 s+=fac;

17}

**解答：**

（1）错误修改：

1) 第2，3行间应声明函数sum\_fac，正确形式为：

long long sum\_fac(int );

2) 第7行的sum\_fac返回值类型为long long，正确形式为：

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

3) 第10，12，13行，由于计算的是阶乘之和，应使用long long作为函数返回值，正确形式为：

long long sum\_fac(int n)

long long s = 0;

long long i, fac = 1;

4) 第16行为计算阶乘之和而非单独一个阶乘，应进入循环中，正确形式为：

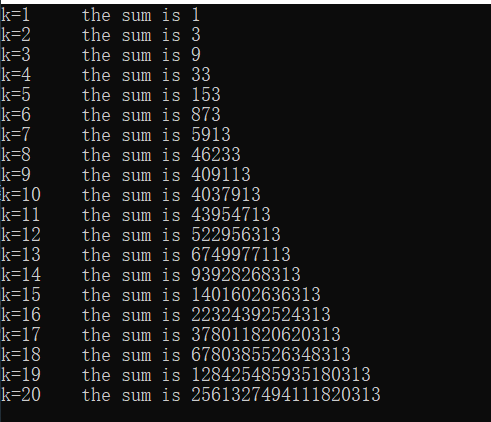
for(i=1;i<=n;i++) {

fac\*=i;

s+=fac; //挪进来

}

（2）错误修改后运行结果：



**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：**

由方程知，可写出递归方程sum\_fac(n) = sum\_fac(n-1) + n!。

而递归的边界条件为当n=0时，sum\_fac(n) = 0。

使用C语言完成递归的程序。

/\*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\*/

#include <stdio.h>

long long sum\_fac(int);

int main(void)

{

int k;

for (k = 1; k <= 20; k++)

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

return 0;

}

long long sum\_fac(int n)

{

long long s = 0;

int i;

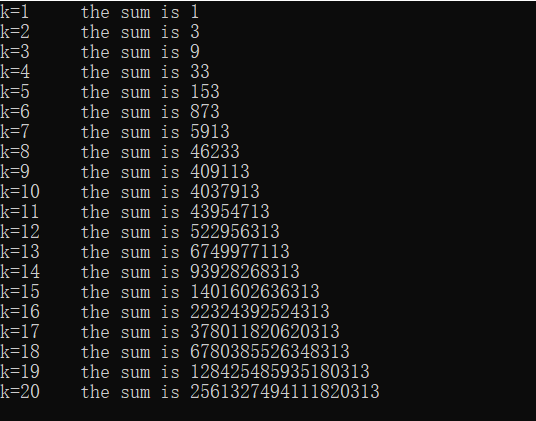
long long fac=1;

for (i = 1; i <= n; i++) fac \*= i;

if (n == 1)return 1;

return fac+sum\_fac(n-1);

}



（2）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\*/

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n)

{

int i;

double z=1.0;

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

{

z=z+mulx(x,i)/fac(i);

}

return z;

}

double mulx(double x,int n)

{

int i;

double z=1.0;

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

{

z=z\*x;

}

return z;

}

long fac(int n)

{

int i;

long h=1;

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

{

h=h\*i;

}

return h;

}

int main()

{

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf:",sum(x,n));

return 0;

}

**解答：**

我们观察到计算x的n次方和n!时与前项有着明显的递推关系，若已知前一个值便可计算下一个值，即x的n+1次方，(n+1)!可被轻松计算。便在理论上达到了最小的计算量，为在函数中保存上一次的计算值，应将mulx中的z变量改为static型，fac中的h变量也改为static类型，改变计算方式即可。

/\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

#include<stdio.h>

double mulx(double x, int n);

long fac(int n);

double sum(double x, int n)

{

int i;

double z = 1.0;

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

z = z + mulx(x, i) / fac(i);

return z;

}

double mulx(double x, int n)

{

static double z = 1.0;

z = z \* x;

return z;

}

long fac(int n)

{

static long h = 1;

h = h \*n;

return h;

}

int main()

{

double x;

int n;

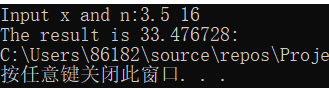
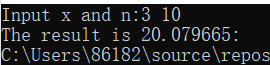
printf("Input x and n:");

scanf\_s("%lf%d", &x, &n);

printf("The result is %lf:", sum(x, n));

return 0;

}





**3.2.3 程序跟踪调试**

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：

**/\*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);

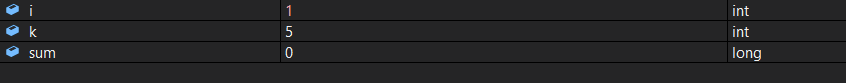
}

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

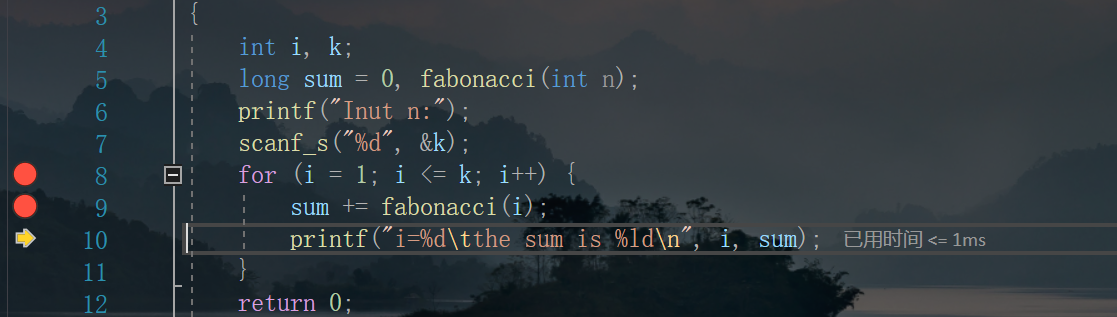


如图，刚执行完“scanf("%d",&k);”语句时，sum、k的值分别为0，5。

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？

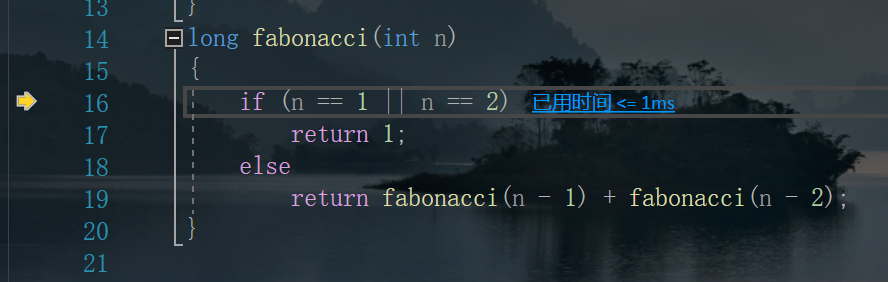


如图，在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值i，k，sum的值分别为1，5，0。



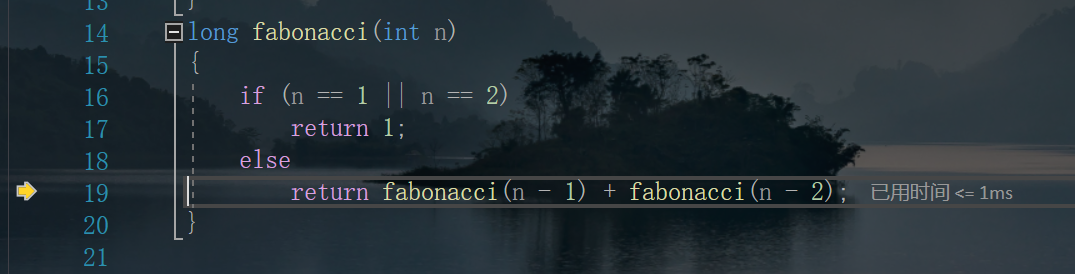
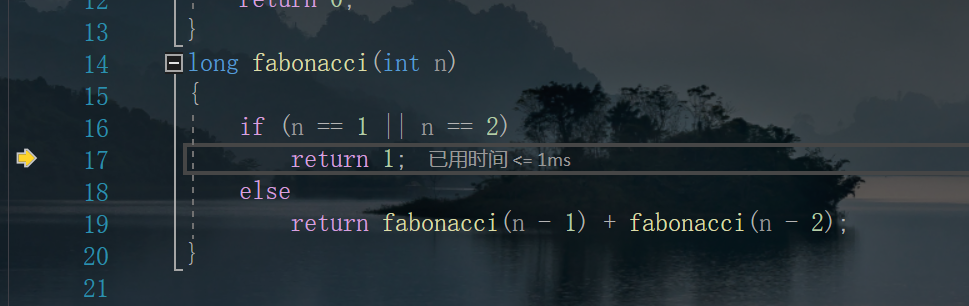
返回后光标停留在如图第10行的输出语句。

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？



如图，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fabonacci函数实际执行的第一行，即第16行的if(n==1 || n==2)语句。

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？



当函数接收到n时，若n为1或2，达到递归终止条件直接返回答案1。否则，将递归调用fabonacci(n-1)和fabonacci(n-2)。当两个计算完成时，继续完成函数，返回答案值。结束递归时，然后直接从第一个return语句返回到了计算fabonacci(n-1)和fabonacci(n-2)之和并返回的第19行return fabonacci(n-1)+fabonacci(n-2);。

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？

K、sum为main函数的局部变量，在另一个函数中并未声明。故在另一个函数体内不可见。

**3.2.4 程序设计**

（1） 编程验证歌德巴赫猜想：任何一个大于等于4的偶数都是两个素数之和。要求设计一个函数，接受形参n，以“n=n1+n2”的形式输出结果，若有多种分解情况，取n1最小的一个输出。

例如：n=6，输出“6=3+3”并换行。

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，直至输入Ctrl+Z程序结束。

解答：

1）通过分析本题的算法流程，可建立如下算法流程图。

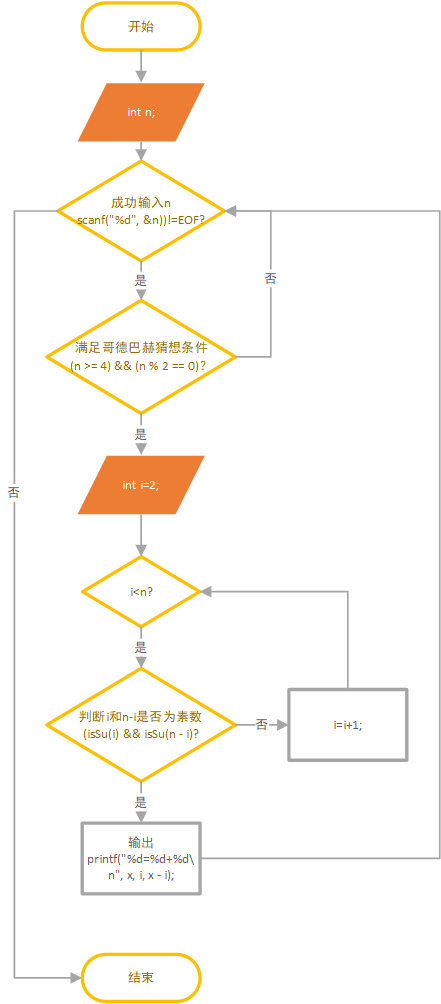


图3-1 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

int isSu(int);

void geDe(int);

int main(void) {

int n = 4;

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

if ((n >= 4) && (n % 2 == 0)) {

geDe(n);

}

}

}

int isSu(int n) {

for (int k = 2; k < n; k++)

if (n % k == 0)return 0;

return 1;

}

void geDe(int x) {

for (int i = 2; i < x; i++) {

if (isSu(i) && isSu(x - i)) {

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

}

}

}

3）测试

（a） 测试数据：

200

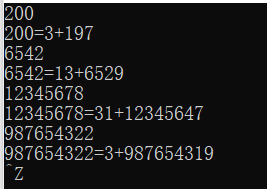
6542

12345678

987654322

Ctrl + Z

（b） 对应测试数据的运行结果截图



（2）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）之和恰好等于它本身。例如6=1+2+3，28=1+2+4+7+14等。

编程寻找10000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，返回1，否则返回0。在main函数中调用该函数求10000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”。程序输出中，每个完全数单独占一行。**解答：**

1. 解题思路：

通过分析本题的算法流程，可对需要设计的函数建立如下算法流程图。

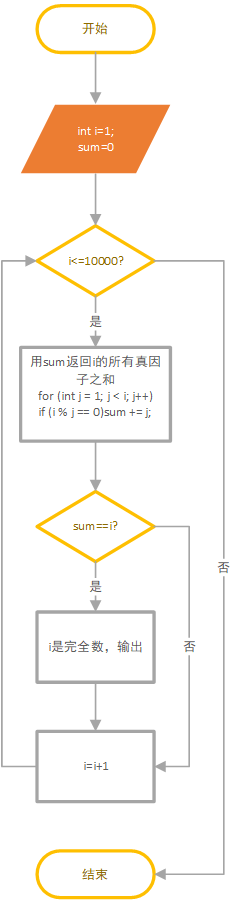


图3-2 编程题2的程序流程图

2）程序清单

#include <stdio.h>

int perfect(int);

int main(void) {

for (int i = 1; i <= 10000; i++) {

if (perfect(i)) {

int str[10000], j = 0;

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

if (i % k == 0) {

str[j] = k;

j++;

}

}

printf("%d=", i);

for (int m = 0; m < j-1; m++)printf("%d+", str[m]);

printf("%d\n", str[j - 1]);

}

}

return 0;

}

int perfect(int x) {

int sum=0;

for (int i = 1; i < x; i++)

if (x % i == 0)sum += i;

if (sum == x)return 1;

else return 0;

}

3）测试

（a） 测试数据：

本测试无输入，有固定答案。

可知1-10000内有4个完全数。

6=1+2+3

28=1+2+4+7+14

496=1+2+4+8+16+31+62+124+248

8128=1+2+4+8+16+32+64+127+254+508+1016+2032+4064

（b） 对应测试数据的运行结果截图

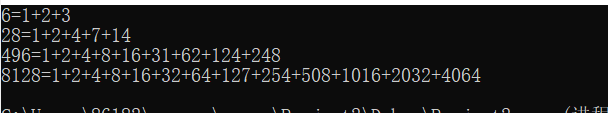


图3-3 编程题2的测试运行结果

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） 自幂数是指一个n位数，它每位数字的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，返回1；否则，返回0。

要求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所有k位的自幂数，输出相应信息并换行，例如“3位的水仙花数共有4个153,370,371,407\n”。当k=0时程序结束执行。

**解答：**

1. 解题思路：

实现的函数判断i是否为n位的自幂数的算法流程如下：

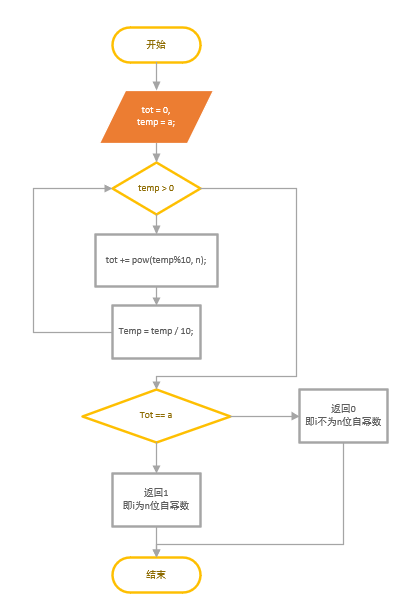


图3-4 编程题3的部分函数程序流程图

2）程序清单

#include <stdio.h>

#include <math.h>

int isZi(int);

int main(void) {

int k=1;

char a1[100][100] = { "水仙花数","四叶玫瑰数","五角星数","六合数","北斗星数","八仙数" };

while (k) {

scanf("%d", &k);

if (k) {

int j = 0, str[100];

for (int i = pow(10, (double)k - 1); i < pow(10, (double)k); i++) {

if (isZi(i)) {

str[j] = i;

j++;

}

}

printf("%d位的", k);

printf(a1[k - 3]);

printf("共有%d个", j);

for (int m = 0; m < j - 1; m++)printf("%d,", str[m]);

printf("%d\n", str[j - 1]);

}

}

return 0;

}

int isZi(int n) {

int count = 0, sum = 0,m=n,l=n;

while (m) {

m /= 10;

count++;

}

while (l) {

sum += pow(l % 10,count);

l/= 10;

}

if (sum == n)return 1;

else return 0;

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-2所示。

表3-1 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 输 入 | 理 论 结 果 | 运 行 结 果 |
| N |
| 例1 | 3  4  5  0 | 3位的水仙花数共有4个153,370,371,407  4位的四叶玫瑰数共有3个1634,8208,9474  5位的五角星数共有3个54748,92727,93084 | 3位的水仙花数共有4个153,370,371,407  4位的四叶玫瑰数共有3个1634,8208,9474  5位的五角星数共有3个54748,92727,93084 |
| 例2 | 6  7  0 | 6位的六合数共有1个548834  7位的北斗星数共有4个1741725,4210818,9800817,9926315 | 6位的六合数共有1个548834  7位的北斗星数共有4个1741725,4210818,9800817,9926315 |
| 例3 | 5  8  4  7  5  0 | 5位的五角星数共有3个54748,92727,93084  8位的八仙数共有3个24678050,24678051,88593477  4位的四叶玫瑰数共有3个1634,8208,9474  7位的北斗星数共有4个1741725,4210818,9800817,9926315  5位的五角星数共有3个54748,92727,93084 | 5位的五角星数共有3个54748,92727,93084  8位的八仙数共有3个24678050,24678051,88593477  4位的四叶玫瑰数共有3个1634,8208,9474  7位的北斗星数共有4个1741725,4210818,9800817,9926315  5位的五角星数共有3个54748,92727,93084 |

（b） 对应测试测试用例1的运行结果如图3-5所示。

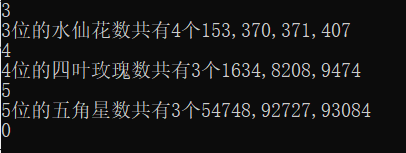


图3-5 编程题3的测试用例一的运行结果

对应测试测试用例2的运行结果如图3-6所示。

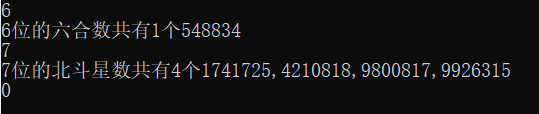


图3-6 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图3-7所示。

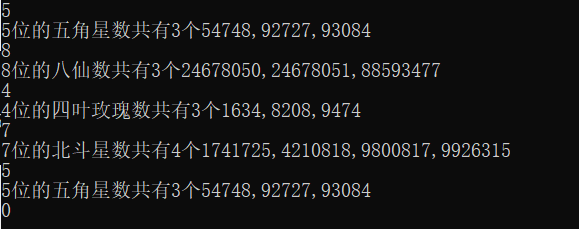


图3-7 编程题3的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 3.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验1改错中，我知道了要根据具体问题来选取具体的数据类型。在修改题1中初步了解了递归的实现和意义，使用递归完成了阶乘之和的实现，并知道了具有递推特征的数学量计算复杂度可简化至O(1)，并且运用了static修饰符实现了之前计算量的存储；在逐步调试和观察调试代码时，更加深入的理解了递归的实现方式，和运算方式。

在编程题中，我也学会了合理设计函数，用以简化程序的编写。

# 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 源程序改错**

**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）错误修改：

1) 第2，3行在宏定义时应加上括号，正确形式为：

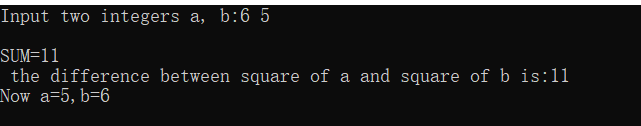
#define SUM (a+b)

#define DIF (a-b)

2) 第4行交换两数的程序有误，正确形式为：

#define SWAP(a,b) a=b-a,b=b-a,a=a+b

（2）错误修改后运行结果：



**4.2.2 源程序修改替换**

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

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；

/\*实验4-2程序修改替换题程序\*/

1#include<stdio.h>

2int 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}

14int 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}

23float sum(float x, float y)

24{

25 return x+y;

26}

**解答：**

（1）错误修改：

1) max和sum在main函数之前应有声明，正确形式为：

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

float sum(float x, float y);

2)max函数存在逻辑错误，正确形式为：

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

{

int m = z;

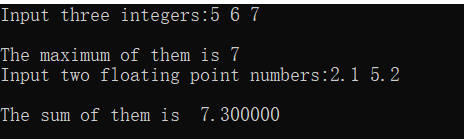
if (x > y && x > z)m = x;

else if (y > x && y > z)m = y;

return m;

}

1. 错误修改后运行结果：



（2）用带参数的宏替换函数max，来实现求最大数的功能。

**解答：**

max函数的功能是返回三个数中的最大值，那么我们可以用三目运算符直接将所传入的三个数中的最大值作为max的值。

#include<stdio.h>

#define max(a,b,c) ((a)>(b)?(a>c?a:c):(b>c?b:c))

float sum(float x, float y);

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

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

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

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

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

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

return 0;

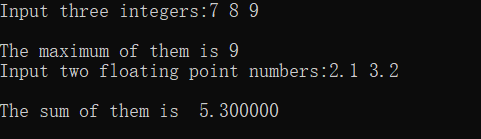
}

float sum(float x, float y)

{

return x + y;

}



**4.2.3 程序跟踪调试**

下面程序利用R计算圆的面积s，以及面积s的整数部分。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

#define R

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

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

s\_integer=integer\_fraction(s);

assert((s-s\_integer)<0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

（1）修改程序，使程序编译通过且能运行；

**解答：**

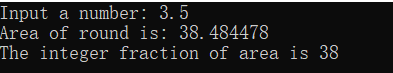
(1)程序没有引入头文件，应在开头添加：

#include <stdio.h>

#include <assert.h>

(2) integer\_fraction函数没有声明，应在main函数之前添加：

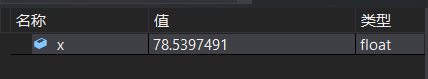
int integer\_fraction(float x);



（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？

**解答：**

(1)输入5，进入函数integerl\_fraction时，x的值如下：



(2) 在返回main时, i的值如下：



（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。

**解答：**

要求对面积s的值四舍五入，即需要修改integer\_fraction函数，判断s的小数部分是否大于等于0.5，大于0.5就取s的整数部分加一。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

#define R

#include <stdio.h>

#include <assert.h>

int integer\_fraction(float x);

int main(void)

{

float r, s;

int s\_integer = 0;

printf("Input a number: ");

scanf("%f", &r);

#ifdef R

s = 3.14159 \* r \* r;

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

s\_integer = integer\_fraction(s);

assert((s - s\_integer) < 0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

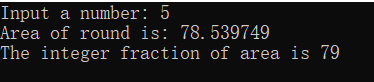
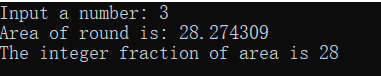
{

int i=x;

if (x - i >= 0.5)i++;

return i;

}



**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）通过分析本题的算法流程，可建立如下算法流程图。

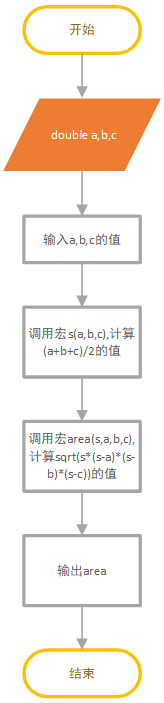


图4-1 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

#include <math.h>

#define s(a,b,c) (a+b+c)/2

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

int main(void) {

double a, b, c;

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

printf("area=%lf", area(s(a, b, c), a, b, c));

return 0;

}

3）测试

（a） 测试数据：

5 6 7

（b） 对应测试数据的运行结果截图



（2）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：

通过分析本题的算法流程，可对需要设计的函数建立如下算法流程图。

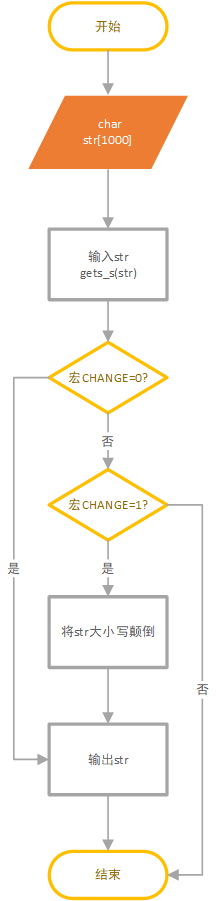


图4-2 编程题2的程序流程图

2）程序清单

#include <stdio.h>

#define CHANGE 1

int main(void) {

char str[1000];

gets\_s(str);

#if CHANGE == 0

printf(str);

#elif CHANGE == 1

for (int i = 0; str[i]; i++) {

if (str[i] >= 'a' && str[i] <= 'z') str[i] += ('A' - 'a');

else if (str[i] >= 'A' && str[i] <= 'Z') str[i] += ('a' - 'A');

}

printf(str);

#endif

return 0;

}

3）测试

（a） 测试数据：

1) CHANGE=0 输入AbCd Ef G

2) CHANGE=1 输入AbCd Ef G

（b） 对应测试数据的运行结果截图





图4-3 编程题2的测试运行结果

（3）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。

/\*源文件file1.c的内容\*/

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

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

func1();

return 0;

}

/\*源文件file2.c的内容为：\*/

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

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

}

**解答：**

1. 解题思路：

因为在file2中调用了在file1中定义的x,y,ch变量，在file1中调用了file2中定义的函数func1，所以要在头文件file.h中用extern关键字声明变量x.y.ch和函数func1，以保证在file1和file2中可以对其定义并调用。

2）程序清单

/\*头文件file.h的内容为：\*/

#include <stdio.h>

extern int x, y;

extern char ch;

void func1(void);

/\*源文件file1.c的内容\*/

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

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

func1();

return 0;

}

/\*源文件file2.c的内容为：\*/

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

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

}

3）测试

（a） 测试数据：

a

2

（b） 对应测试测试用例1的运行结果如图4-4所示。

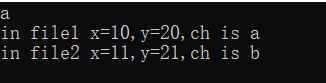


图4-4 编程题3的测试用例一的运行结果

对应测试测试用例2的运行结果如图4-5所示。

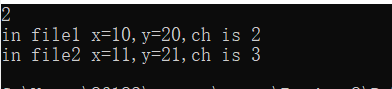


图4-5编程题3的测试用例二的运行结果

## 4.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验1改错中，我知道了宏定义的方式和特性，对宏的使用方法有了进一步的了解；在源程序替换修改中，我通过用宏代替函数的方式体会到了宏运算的简便性，也让我掌握了如何不用条件判断求出三个数的最大值的方法；在编程题中，我也学会了合理设计函数，用以简化程序的编写；了解了头文件和源文件之间的关系以及extern关键字的使用。

# 5 数组试验

## 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**

**1．程序改错**

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

ProgramingLnguage

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。

/\*实验5-1程序改错与跟踪调试题程序\*/

#include<stdio.h>

void strcate(char [],char []);

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[]="Programming";

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

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

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

return 0;

}

void strcate(char t[],char s[])

{

int i = 0, j = 0;

while(t[i++]) ;

while((t[i++] = s[j++] )!= '\0');

}

void strdelc(char s[], char c)

{

int j,k;

for(j=k=0; s[j] != '\0'; j++)

if(s[j] != c) s[k++] = s[j];

}

**解答：**

（1）错误修改：

1) a、b数组定义过小，会导致数组越界，正确形式为：

char a[1000] = "Language", b[1000] = "Programming";

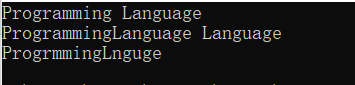
2) 第15行的while (t[i++]);会导致i多加了一遍，正确形式为：

while (t[++i]);

3)strdelc函数中s数组缺少结束符\0，应在最后加上：

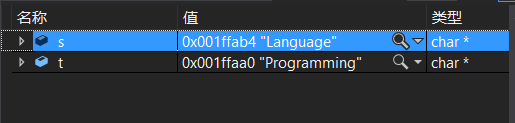
s[k] = '\0';

（2）错误修改后运行结果：

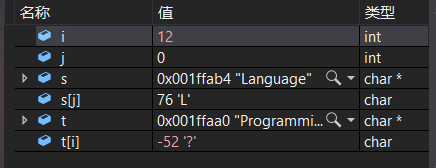


**2．程序跟踪调试**

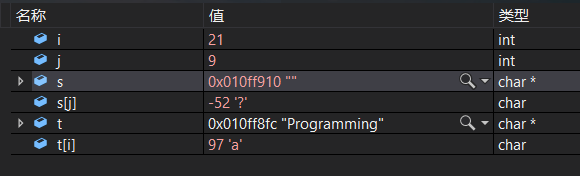
(1) 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。



如图，进入strcate时，数组s中的内容为Language，数组t中的内容为Programming，结果正确。

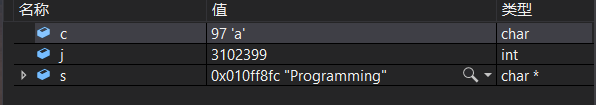


如图，当单步执行光条刚落在第二个while语句所在行时，i的值为12，t[i]为’?’，结果错误，i的值应为11，t[i]应为’\0’。

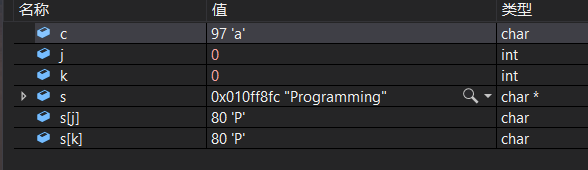
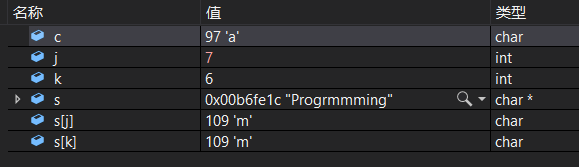


如图，当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t为Programming，s为空，没有实现字符串拼接。

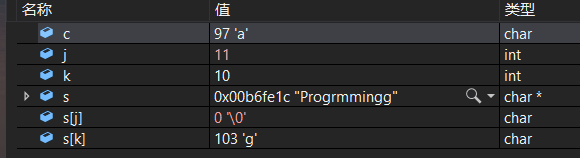
（2）跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。



如图，进入函数strdelc时，字符数组s的内容为Programming，字符c的值为a，结果正确。

如图，在单步执行for语句过程中，当s[j]==c时，k不变，j加一，s中删除了字符c；否则k和j都加一，s不变。结果没有问题。



如图，当单步执行光条落在strdelc函数块结束标记“}”所在行时，s的内容为Progrmmingg，实现了删除字符的功能，但由于没有添加结束符，导致s中最后一个字符重复两次，结果错误。

**5.2.2 源程序完善和修改替换**

(1) 下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。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。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

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

int i, j, k;

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

a[i] = i + 1;

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

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

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

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

b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/

if(j)

for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

printf(“%6d”, b[i]);

printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

**解答：**

当j不为0时，表明编号为a[j-1]的人报数为N，所以将其存入数组b；当j为0时，表明最后一个人刚好报到N，所以此时将最后一个人的编号a[i-1]存入数组b；一轮报数结束后，需要压缩数组a，将报数为N的人的编号从a中删除。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

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

int i, j, k;

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

a[i] = i + 1;

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

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

for (k = 1; k <= N; k++) /\* 1至N报数 \*/

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

//printf("%d", j);

b[M - i] = j ? a[j-1] :a[i-1] ; /\* 将报数为N的人的编号存入数组b \*/

if (j)

for (k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

a[k] = a[k + 1];

}

for (i = 0; i < M - 1; i++) /\* 按次序输出出圈人的编号 \*/

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

printf(" % 6d\n", a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

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

**解答：**

定义数组a存放圈中人的编号，当有人报数为N时，将其编号置为0，在输出时增加编号不为零的判断条件即可。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M]; /\* 数组a存放圈中人的编号 \*/

int i,x=M,k=0;

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

a[i] = i+1;

while(x>1){

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

if (a[i]) k++;

if (k == N) {

printf("%6d", a[i]); /\* 按次序输出出圈人的编号 \*/

a[i] = 0;

k = 0;

x--;

}

}

}

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

if(a[i])printf(" %6d\n", a[i]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}



**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）通过分析本题的算法流程，可建立如下算法流程图。

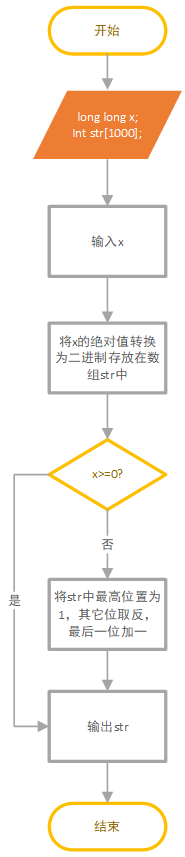


图5-1 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

#include <math.h>

int main(void) {

long long x = 0, k = 0;

int str[1000] = { 0 };

scanf("%lld", &x);

int m = abs(x);

while (m) {

int b = m / 2;

str[k] = m - b \* 2;

k++;

m /= 2;

}

if (x >= 0) {

for (int i = 31; i >= 0; i--)printf("%d", str[i]);

}

else {

int n = 0;

str[31] = 1;

printf("%d", str[31]);

for (int j = 30; j >= 0; j--) {

if (str[j] == 1)str[j] = 0;

else str[j] = 1;

}

if (str[0] == 0)str[0] = 1;

else {

while (str[n]) {

if (str[n] == 1)str[n] = 0;

else str[n] = 1;

n++;

}

str[n] = 1;

}

for(int q=30;q>=0;q--)

printf("%d", str[q]);

}

printf("\n");

return 0;

}

3）测试

（a） 测试数据：

7

-7

0

200

-200

（b） 对应测试数据的运行结果截图











（2）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“成绩输入”，输入n个学生的姓名和C语言课程的成绩。

②“成绩排序”，将成绩按从高到低的次序排序，姓名同时进行相应调整。成绩相同的，按照输入先后次序排列。

③“成绩输出”，输出排序后所有学生的姓名和C语言课程的成绩。

④“成绩查找”，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：

通过分析本题的算法流程，可对需要设计的函数建立如下算法流程图。

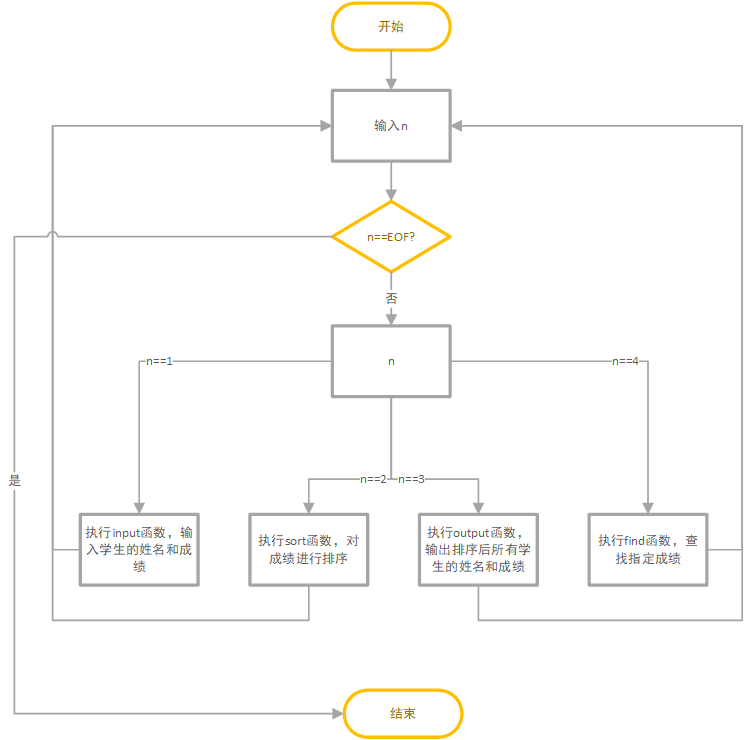


图5-2 编程题2的程序流程图

2）程序清单

#include <stdio.h>

#include <string.h>

void input(char name[][100], int score[100], int\*);

void sort(char name[][100], int score[100], int\*);

void output(char name[][100], int score[100], int\*);

void find(char name[][100], int score[100], int\*);

int main(void) {

int n = 1, score[100] = { 0 }, k = 0;

char name[100][100];

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

switch (n)

{

case 0:return 0;

case 1:input(name, score, &k); break;

case 2:sort(name, score, &k); break;

case 3:output(name, score, &k); break;

case 4:find(name, score, &k); break;

}

}

//printf("（程序退出）\n");

return 0;

}

void input(char name[][100], int score[100], int\* k) {

scanf("%d", k);

for (int i = 0; i < \*k; i++) {

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

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

}

printf("%d records were input!\n", \*k);

}

void sort(char name[][100], int score[100], int\* k) {

for (int i = 0; i < \*k - 1; i++) {

for (int j = 0; j < \*k - i - 1; j++) {

if (score[j] < score[j + 1]) {

int temp = score[j];

score[j] = score[j + 1];

score[j + 1] = temp;

char temp1[100] = { 0 };

strcpy(temp1, name[j]);

strcpy(name[j], name[j + 1]);

strcpy(name[j + 1], temp1);

}

}

}

printf("Reorder finished!\n");

}

void output(char name[][100], int score[100], int\* k) {

for (int i = 0; i < \*k; i++)

printf("%s %d\n", name[i], score[i]);

}

void find(char name[][100], int score[100], int\* k) {

//sort(name, score, k);

int low = 0, high = \*k - 1, mid=0,s;

scanf("%d", &s);

while (low <= high) {

mid = (low + high) / 2;

if (score[mid] > s)low = mid + 1;

else if (score[mid] == s) { printf("%s %d\n", name[mid], score[mid]); break; }

else high = mid - 1;

}

if(score[mid]!=s) printf("not found!\n");

}

3）测试

（a） 测试数据：

1

6

Jack 95

Mike 90

Joe 75

Andy 95

Rose 89

Sophia 77

2

3

4

89

0

（b） 对应测试数据的运行结果截图

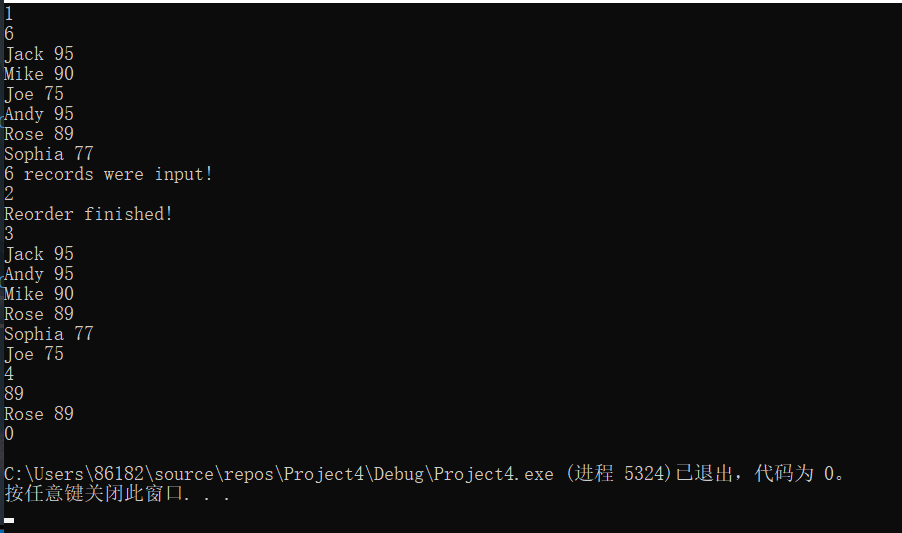


图5-3 编程题2的测试运行结果

（3） 求解N皇后问题，即在N×N的棋盘上摆放N个皇后，要求任意两个皇后不能在同一行、同一列、同一对角线上。输入棋盘的大小N（N取值1-10），如果能满足摆放要求，则输出所有可能的摆放法的数量，否则输出“无解。”

**解答：**

1. 解题思路：

用回溯法在递归时判断是否满足条件，如果不满足，则不用递归下去，返回上一层进行处理。递归时判断当前的皇后是否和前面的皇后在一条对角线上，如果在一条直线上，就不需要递归下去了，返回上一层；如果不在，就继续递归，下一个继续进行判断，直到满足条件为止。

2）程序清单

#include <stdio.h>

int N, tot, row[100], lex[100], rex[100];

int maplex[100][100], maprex[100][100];

void pre () {

int count = 0;

for(int i = 1; i <= N; i++) {

count ++;

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

maplex[j][N-i+j] = count;

}

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

count ++;

for(int j = 1; j <= N - i + 1; j++)

maplex[i+j-1][j] = count;

}

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

for(int j = 1; j <= N; j++)

maprex[i][j] = maplex[i][N-j+1];

}

void dfs (int n) {

if(n == N + 1) {

tot ++;

return ;

}

for(int j = 1; j <= N; j ++) {

if((!row[j]) && (!lex[maplex[n][j]]) && (!rex[maprex[n][j]])) {

row[j] = lex[maplex[n][j]] = rex[maprex[n][j]] = 1;

dfs(n+1);

row[j] = lex[maplex[n][j]] = rex[maprex[n][j]] = 0;

}

}

}

int main () {

scanf("%d", &N);

pre();

dfs(1);

if(tot == 0)

printf("无解");

else

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

return 0;

}

3）测试

（a） 测试数据：

1

2

6

10

（b） 对应测试测试用例1的运行结果如图5-4所示。



图5-4 编程题3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5-5所示。



图5-5 编程题3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5-6所示。



图5-6 编程题3的测试用例三的运行结果

对应测试测试用例4的运行结果如图5-7所示。



图5-7 编程题3的测试用例四的运行结果

## 5.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在源程序完善和修改替换中，我知道了要如何解决约瑟夫问题，并通过改进算法减小了算法的复杂度。在编程题1中我加深巩固了正数和负数的二进制存储方法以及二进制的运算；在编程题3中了解了如何求解N皇后问题，并对回溯算法有了一定的理解，增强了我解决问题的能力，也学会了合理设计函数，用以简化程序的编写。

# 6 指针试验

**6.1实验目的**

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

## 6.2 实验内容

**6.2.1 源程序改错**

**1．程序改错**

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

#include<stdio.h>

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

int main(void)

{

char \*s1, \*s2, \*s3;

printf("Input a string:\n", s2);

scanf("%s", s2);

strcopy(s1, s2);

printf("%s\n", s1);

printf("Input a string again:\n", s2);

scanf("%s", s2);

s3 = strcopy(s1, s2);

printf("%s\n", s3);

return 0;

}

/\*将字符串s复制给字符串t，并且返回串t的首地址\*/

char \* strcopy(char \*t, const char \*s)

{

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

return (t);

}

**解答：**

（1）错误修改：

1)第5行应给s1,s2,s3分配内存，同时应引入头文件，正确形式为：

#include<stdlib.h>

char\*s1=(char\*)malloc(sizeof(char)\*100),\*s2=(char\*)malloc(sizeof(char)\*100), \*s3=(char\*)malloc(sizeof(char) \* 100);

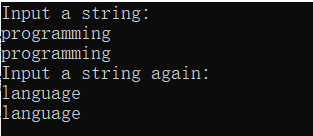
2)在strcopy函数中t指针发生了变化，未能正确返回串t的首地址，应另设一指针指向t的首地址，正确形式为：

char\* p = t;

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

return (t);

（2）错误修改后运行结果：



**6.2.2 源程序完善和修改替换**

（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。

#include<stdio.h>

#include<\_\_\_\_\_\_\_\_\_>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

\_\_\_\_\_\_\_temp;

int i, j;

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

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

if (\_\_\_\_\_\_\_\_\_\_\_)

{

temp = s[j];

\_\_\_\_\_\_\_\_\_\_\_;

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

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

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(\_\_\_\_\_\_\_);

}

strsort(\_\_\_\_\_\_\_\_);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

**解答：**

程序中使用了malloc函数，故应引入头文件<stdlib.h>；temp作为交换s[j]和s[j+1]的中间变量，数据类型应与s[j]相同；第11行的判断条件应是s[j]的字典序比s[j+1]高，故使用库函数strcmp。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char\* s[], int size)

{

char\* temp;

int i, j;

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

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

if (strcmp(s[j],s[j+1])>0)

{

temp = s[j];

s[j]=s[j+1];

s[j + 1] = temp;

}

}

int main()

{

int i;

char\* s[N], t[50];

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

{

gets\_s(t);

s[i] = (char\*)malloc(strlen(t) + 1);

strcpy(s[i],t);

}

strsort(s,N);

for (i = 0; i < N; i++) { puts(s[i]); free(s[i]); }

return 0;

}

**测试：**

（a） 测试数据：

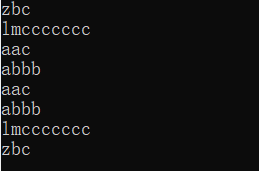
zbc

lmccccccc

aac

abbb

（b） 对应测试数据的运行结果截图



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：**

定义两个二级指针p和q，使p首先指向s中第一个字符串的首地址，q指向p后一位，在每次循环中使p和q后移一位，每次让\*p和\*q交换，即可达到不使用下标引用完成函数功能的效果。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*\*s, int size)

{

char\* temp;

char\*\* p=s, \*\* q;

int i, j;

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

for (j = 0,p=s,q=s+1; j < size - i - 1; j++,p++,q++)

if (strcmp(\*p,\*q) > 0)

{

temp = \*p;

\*p = \*q;

\*q = temp;

}

}

int main()

{

int i;

char\* s[N], t[50];

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

{

gets\_s(t);

s[i] = (char\*)malloc(strlen(t) + 1);

strcpy(s[i], t);

}

strsort(s, N);

for (i = 0; i < N; i++) { puts(s[i]); free(s[i]); }

return 0;

}

**测试：**

（a） 测试数据：

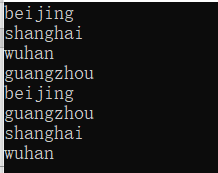
beijing

shanghai

wuhan

guangzhou

（b） 对应测试数据的运行结果截图



（2）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.

# include<stdio.h>

# include<string.h>

int main (void)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

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

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

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

result = \_\_\_\_\_\_\_\_\_\_\_(a, b);

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

}

down:

return 0;

}

**解答：**

p应为一函数指针，根据用户输入的choice的值分别指向strcpy、strcat、strtok函数，在使用时格式为p(a,b)。

# include<stdio.h>

# include<string.h>

int main (void)

{

char\* (\*p) (char \*a,char \*b);//

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

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

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

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}

while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

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

gets(a);//

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

gets(b);//

result = p(a, b);//

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

}

down:

return 0;

}

**测试：**

（a） 测试数据：

1

hello

world

2

world

helloworld

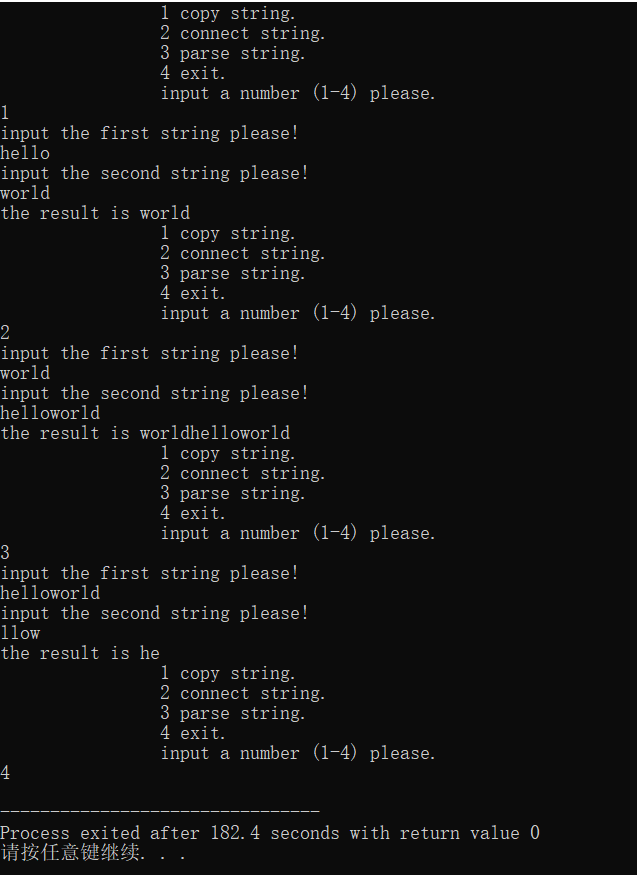
3

helloworld

llo

4

（b） 对应测试数据的运行结果截图



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：**

将p定义为一个函数指针数组，将strcpy、strcat、strtok函数作为元素放入p中，将用户输入的choice值作为数组下标分别调用三个函数，即可避免用switch语句对p进行不同情况的赋值。

# include<stdio.h>

# include<string.h>

int main (void)

{

char\* (\*p[]) (char\* , char\*) = {0, strcpy, strcat, strtok};

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

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

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

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}

while(choice<1 || choice>4);

if(choice == 4)

break;

getchar();

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

gets(a);//

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

gets(b);//

result = p[choice](a, b);//

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

}

down:

return 0;

}

**测试：**

（a） 测试数据：

1

huazhong

zhong hua

2

zhonghua

hua zhong

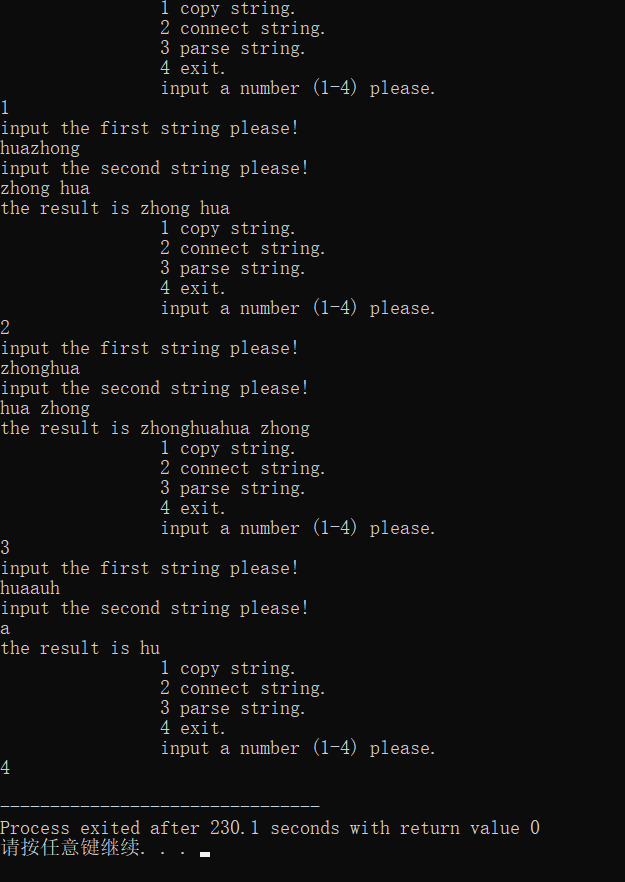
3

huaauh

a

4

（b） 对应测试数据的运行结果截图



**6.2.3 程序跟踪调试**

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。

#include "stdio.h"

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

void main(void)

{

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

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

}

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

{

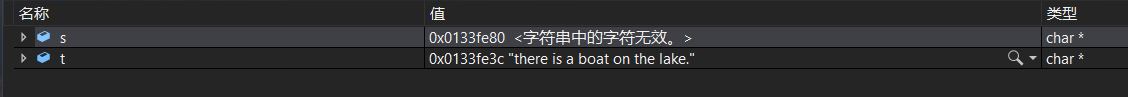
while(\*s++=\*t++)

;

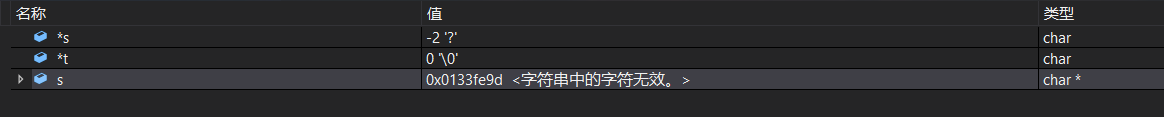
return (s);

}

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



如图，进入strcpy时s未被初始化，字符串中的字符无效。



如图，返回main时，s中的字符无效

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

**解答：**

数组a容量过小，“there is a boat on the lake”已经超过了20个字符；strcpy函数中s指针发生了变化，return(s)不能返回首地址，应另设一指针指向s首地址。



**6.2.4 程序设计**

（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）在main函数中设置两个形参：int类型的argc和二维数组args。argc表示

传入参数的个数，argc用来存储传入的参数。

2）源程序清单

#include <stdio.h>

int main(int argc,char\*args[]){

for(int i=1;i<=argc;i++){

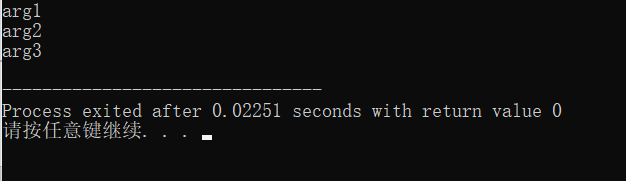
printf("%s\n",args[i]);

}

}

3）测试

对应测试数据的运行结果截图



（2）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。输入一个长整型变量，要求从高字节开始，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，通过指针取出每字节。

样例输入：15

样例输出：0000000F

**解答：**

1. 解题思路：

设置一指针变量，首先指向输入的长整型变量的第一个字节，将每个字节的高4位和低4位取出并以十六进制形式输出。

2）程序清单

#include <stdio.h>

int main(void) {

long long x;

while ((scanf("%lld", &x)) != EOF)

printf("%08X\n", x);

return 0;

}

3）测试

（a） 测试数据：

2147483647

15

4294967295

（b） 对应测试数据的运行结果截图

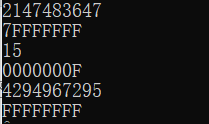


图6-1 编程题2的测试运行结果

（3） 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。

**解答：**

1. 解题思路：

将一个矩阵逆时针旋转90度，就是要把原矩阵中的最后一列作为第一行，直到把第一列作为最后一行。所以应该从最后一列开始输出，每输完一列输出一个换行符。

2）程序清单

#include <stdio.h>

int array[2000][2000];

void out(int, int, int a[][2000]);

int main(void) {

int n, m;

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

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

for (int j = 0; j < m; j++)

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

out(n, m, array);

return 0;

}

void out(int n, int m, int a[][2000]) {

for (int i = m - 1; i >= 0; i--)

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

printf("%d%s", a[j][i],(j==n-1) ? "\n" : " ");

}

3）测试

（a） 测试数据一：

1 1

5

测试数据二：

2 3

1 5 3

3 2 4

测试数据三：

2 3

4 7 8

5 3 2

（b） 对应测试测试用例1的运行结果如图6-2所示。



图6-2 编程题3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6-3所示。

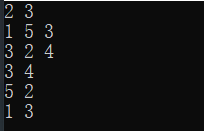


图6-3 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图6-4所示。

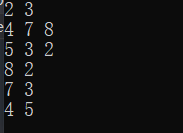


图6-4 编程题3的测试用例三的运行结果

（4）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（n不作为输入，可理解为循环输入多行，以CTRL+Z结束），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。（并在最后输出换行符。）

**解答：**

1. 解题思路：

设置一个标志数组，将其初值都设为0，当标志数组序号对应的字符是每一行中的前置空格或者水平制表符时，将其标志位设为1。输出时判断每个字符所对应的标志元素是否为0，是则输出，否则不输出。

2）程序清单

#include <stdio.h>

void Delete(char str[][100], int flag[][100], int n);

int main(void) {

char str[100][100];

int flag[100][100] = { 0 };

int i = 0, j, n = 0;

while ((gets(str[i])) != NULL) {

n++;

i++;

}

Delete(str, flag, n);

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

for (j = 0; str[i][j]; j++) {

if (!flag[i][j])printf("%c", str[i][j]);

}

printf("\n");

}

return 0;

}

void Delete(char str[][100],int flag[][100], int n) {

int i = 0, j = 0, k = 0;

while (k < n) {

j = 0;

while (str[k][j] == ' ') {

flag[k][j] = 1;

j++;

}

k++;

}

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

for (j = 0; str[i][j]; j++)

if (str[i][j] == '\t')flag[i][j] = 1;

}

3）测试

（a） 测试数据一：

123/t45

67/t89

测试数据二：

aab c

bcd/te

测试数据三：

a c f/tl

abcd

（b） 对应测试测试用例1的运行结果如图6-5所示。

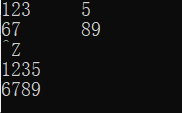


图6-5 编程题4的测试用例一的运行结果

对应测试测试用例2的运行结果如图6-6所示。

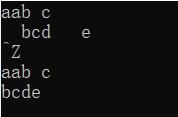


图6-6 编程题4的测试用例二的运行结果

对应测试测试用例3的运行结果如图6-7所示。

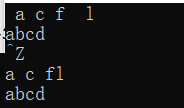


图6-7 编程题4的测试用例三的运行结果

（5） 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。

**解答：**

1. 解题思路：

设置一个函数指针数组，每当用户输入数字时，将此数字代表的函数地址作为元素存放在数组中，并将此数组作为参数传进execute函数中，依次按用户输入的顺序，即数组中元素的顺序执行函数。

2）程序清单

#include <stdio.h>

void task0() { printf("task0 is called!"); }

void task1() { printf("task1 is called!"); }

void task2() { printf("task2 is called!"); }

void task3() { printf("task3 is called!"); }

void task4() { printf("task4 is called!"); }

void task5() { printf("task5 is called!"); }

void task6() { printf("task6 is called!"); }

void task7() { printf("task7 is called!"); }

void scheduler();

void execute(void (\*fun[100])(), int n);

int main(void) {

scheduler();

return 0;

}

void scheduler() {

char str[1][100];

int count = 0,count1=0;

void (\*fun[100])();

scanf("%s", str[0]);

do

{

switch (str[0][count])

{

case '0':fun[count1] = task0; break;

case '1':fun[count1] = task1; break;

case '2':fun[count1] = task2; break;

case '3':fun[count1] = task3; break;

case '4':fun[count1] = task4; break;

case '5':fun[count1] = task5; break;

case '6':fun[count1] = task6; break;

case '7':fun[count1] = task7; break;

default:count1--;

}

count++;

count1++;

} while (str[0][count]);

execute(fun, count1);

}

void execute(void (\*fun[100])(), int n) {

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

fun[i]();

if(i!=n-1)printf("\n");

}

}

3）测试

（a） 测试数据一：

13607122

测试数据二：

6735643

测试数据三：

012345678

（b） 对应测试测试用例1的运行结果如图6-8所示。

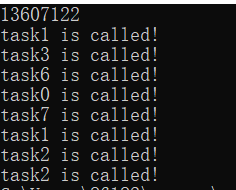


图6-8 编程题5的测试用例一的运行结果

对应测试测试用例2的运行结果如图6-9所示。

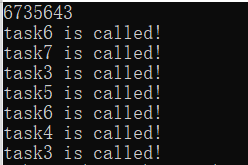


图6-9 编程题5的测试用例二的运行结果

对应测试测试用例3的运行结果如图6-10所示。

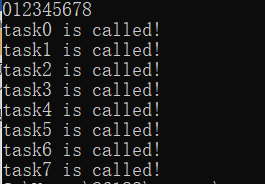


图6-10 编程题5的测试用例三的运行结果

## 6.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在源程序完善和修改替换中，我知道了strcpy、strcmp等库函数的用法和作用，掌握了不用数组下标而使用二级指针表示数组和数组元素的方法，了解了函数指针的作用与用法，体会到了用函数指针数组表示多个函数的方便性；在编程题一中，我了解了向main函数传递参数的方法；在编程题5中，我进一步巩固了函数指针和函数指针数组方面的知识。

# 7 结构与联合试验

**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：**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

(1)进行++p运算后，p指向{100,’B’,v}，此时p->x为100。

(2)首先进行p++运算，p指向{100,’B’,v}，此时p->c为B。

(3)首先进行\*p++->t后，p指向{100,’B’,v}，p->t指向字符数组v的首地址，此时\*p->t为v的首元素x。

(4)首先进行++p运算，p指向{100,’B’,v}，此时\*p->t为x。

(5)首先进行p->t，p->t指向u数组的首地址，++p->t后指向u中第二个元素的地址，故\*++p->t为V。

(6)p->t指向u数组的首地址，故\*p->t为U，进行++运算后结果为字符U的ASCII码加一，得到字符V。

**7.2.2 源程序修改替换**

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

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

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

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

void main(void)

{

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

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

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

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

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

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

{

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

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

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

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

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

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

tail=tail->next; /\* tail指向新创建的结点 \*/

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

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

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

}

**解答：**

在函数creat\_list中，由于传入的参数是在main函数中创建的头指针，所以仅仅对形参进行了创建链表的操作，并不能对实参进行改变，所以应该传入头指针的地址，使用指针传参。

#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 \*/

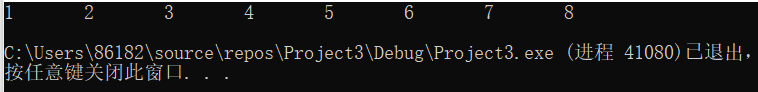
}

**测试：**

（a） 测试数据：

1,2,3,4,5,6,7,8,0

（b） 对应测试数据的运行结果截图



（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：**

首先建立尾结点tail，将第一个元素作为尾结点，并让头指针指向tail，之后的元素进行表头插入。每收到一个非0元素，建立新的结点，使其next域指向头指针，之后再更新头指针，使其指向该结点。

#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 {

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

tail->data = \*p++;

\*headp = tail;

tail->next = NULL;

while (\*p) {

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

loc\_head->next = \*headp;

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

\*headp = loc\_head;

}

}

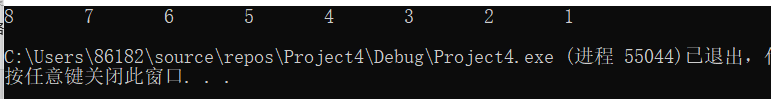
}

**测试：**

（a） 测试数据：

1,2,3,4,5,6,7,8,0

（b） 对应测试数据的运行结果截图



**7.2.3 程序设计**

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

**解答：**

1）解题思路

将每个学生定义成一个结点，num、name、score数组分别存放学生的学号、姓名和四门课程的成绩，avescore存放学生的平均成绩。按照题目要求，分别设计了CreatNode函数，根据输入的学生信息建立单链表并计算出平均成绩赋值给avescore；PrintNode函数，用来输出每个学生的各项信息；ChangeNode函数，根据输入的学号查找该学生对应的结点，对指定数据项进行修改；AveNode函数，输出每个学生的平均成绩；SumNode函数，计算每个学生的总成绩。

2）源程序清单

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node {

    char num[100], name[100];

    int score[4];

    double avescore;

    struct Node\* next;

}Node, \* CNode;

void CreatNode(CNode \*head) {

    int qua, i, j;

    scanf("%d", &qua);

    CNode p , tail;

    p = (CNode)malloc(sizeof(Node));

    \*head = p;

    for (i = 0; i < qua; i++) {

        tail = (CNode)malloc(sizeof(Node));

        scanf("%s", &tail->num);

        scanf("%s", &tail->name);

        for (j = 0; j < 4; j++)

            scanf("%d", &tail->score[j]);

        tail->avescore = (tail->score[0] + tail->score[1] + tail->score[2] + tail->score[3]) / 4.0;

        p->next = tail;

        p = tail;

    }

    tail->next = NULL;

    printf("完成了%d位同学的成绩输入。\n",qua);

}

void PrintNode(CNode\* head) {

    CNode p = \*head;

    while (p->next) {

        printf("%s %s ", p->next->num, p->next->name);

        for (int i = 0; i < 4; i++)

            printf("%d ", p->next->score[i]);

        p = p->next;

        printf("\n");

    }

}

void ChangeNode(CNode\* head) {

    CNode p = \*head;

    char num[100];

    int k = -1;

    scanf("%s", &num);

    while (p->next) {

        if (!strcmp(p->next->num, num)) {

            scanf("%d", &k);

            break;

        }

        p = p->next;

    }

    if (k == 0)

        scanf("%s", &p->next->name);

    if (k > 0) {

        scanf("%d", &p->next->score[k - 1]);

        p->next->avescore = (p->next->score[0] + p->next->score[1] + p->next->score[2] + p->next->score[3]) / 4.0;

    }

    printf("%s %s ", p->next->num, p->next->name);

    for (int i = 0; i < 4; i++)

        printf("%d ", p->next->score[i]);

    printf("\n");

}

void AveNode(CNode\* head) {

    CNode p = \*head;

    while(p->next) {

        printf("%s %s %.2lf\n", p->next->num, p->next->name, p->next->avescore);

        p = p->next;

    }

}

void SumNode(CNode\* head) {

    CNode p = \*head;

    while (p->next) {

        printf("%s %s ", p->next->num, p->next->name);

        double sum = 0;

        for (int i = 0; i < 4; i++)

            sum += p->next->score[i];

        printf("%.0lf %.2lf\n", sum, p->next->avescore);

        p = p->next;

    }

}

int main(void) {

    int n;

    CNode head = NULL;

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

        if (n == 0)break;

        switch (n) {

        case 1:CreatNode(&head); break;

        case 2:PrintNode(&head); break;

        case 3:ChangeNode(&head); break;

        case 4:AveNode(&head); break;

        case 5:SumNode(&head); break;

        }

    }

}

3）测试

(a) 测试数据：

**测试输入1：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

**预期输出1：**

完成了5位同学的成绩输入。

**测试输入2：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

3

2021003 0 Joan

**预期输出2：**

完成了5位同学的成绩输入。

2021003 Joan 77 86 90 75

**测试输入3：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

3

2021003 2 96

**预期输出3：**

完成了5位同学的成绩输入。

2021003 Joan 77 96 90 75

**测试输入4：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

4

**预期输出4：**

完成了5位同学的成绩输入。

2021001 Jack 91.00

2021002 Mike 80.00

2021003 Joan 84.50

2021004 Andy 94.75

2021005 Rose 88.50

**测试输入5：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

5

**预期输出5：**

2021001 Jack 364 91.00

2021002 Mike 320 80.00

2021003 Joan 338 84.50

2021004 Andy 379 94.75

2021005 Rose 354 88.50

完成了5位同学的成绩输入。

**测试输入6：**

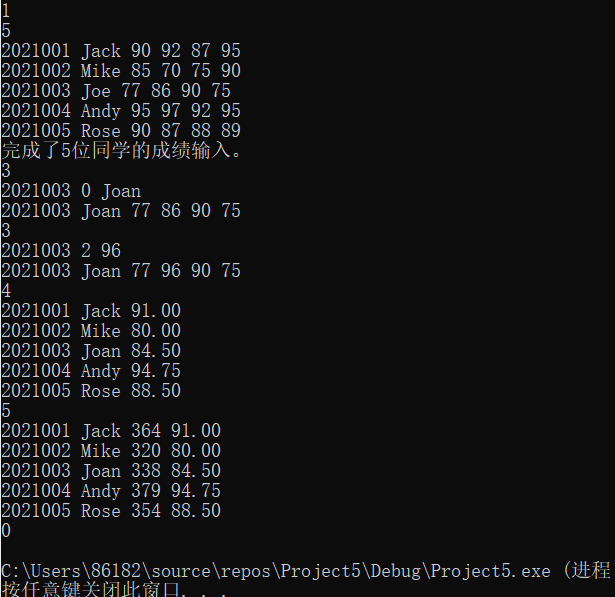
0

**预期输出6：**

（退出）

注意：以上用例需完整的进行，整体前后次序不得颠倒。

(b) 对应测试数据的运行结果截图



（2）对程序设计题第（1）题的程序，⑥增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。

**解答：**

1. 解题思路：

增加一个SortNode函数，使用冒泡排序，定义一指针p指向头结点，如果p->next->avescore大于p->next->next->avescore，则对这两个结点的所有数据域进行交换。

2）程序清单

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node {

    char num[100], name[100];

    int score[4];

    double avescore;

    struct Node\* next;

}Node, \* CNode;

void CreatNode(CNode \*head) {

    int qua, i, j;

    scanf("%d", &qua);

    CNode p , tail;

    p = (CNode)malloc(sizeof(Node));

    \*head = p;

    for (i = 0; i < qua; i++) {

        tail = (CNode)malloc(sizeof(Node));

        scanf("%s", &tail->num);

        scanf("%s", &tail->name);

        for (j = 0; j < 4; j++)

            scanf("%d", &tail->score[j]);

        tail->avescore = (tail->score[0] + tail->score[1] + tail->score[2] + tail->score[3]) / 4.0;

        p->next = tail;

        p = tail;

    }

    tail->next = NULL;

    printf("完成了%d位同学的成绩输入。\n",qua);

}

void PrintNode(CNode\* head) {

    CNode p = \*head;

    while (p->next) {

        printf("%s %s ", p->next->num, p->next->name);

        for (int i = 0; i < 4; i++)

            printf("%d ", p->next->score[i]);

        p = p->next;

        printf("\n");

    }

}

void ChangeNode(CNode\* head) {

    CNode p = \*head;

    char num[100];

    int k = -1;

    scanf("%s", &num);

    while (p->next) {

        if (!strcmp(p->next->num, num)) {

            scanf("%d", &k);

            break;

        }

        p = p->next;

    }

    if (k == 0)

        scanf("%s", &p->next->name);

    if (k > 0) {

        scanf("%d", &p->next->score[k - 1]);

        p->next->avescore = (p->next->score[0] + p->next->score[1] + p->next->score[2] + p->next->score[3]) / 4.0;

    }

    printf("%s %s ", p->next->num, p->next->name);

    for (int i = 0; i < 4; i++)

        printf("%d ", p->next->score[i]);

    printf("\n");

}

void AveNode(CNode\* head) {

    CNode p = \*head;

    while(p->next) {

        printf("%s %s %.2lf\n", p->next->num, p->next->name, p->next->avescore);

        p = p->next;

    }

}

void SumNode(CNode\* head) {

    CNode p = \*head;

    while (p->next) {

        printf("%s %s ", p->next->num, p->next->name);

        double sum = 0;

        for (int i = 0; i < 4; i++)

            sum += p->next->score[i];

        printf("%.0lf %.2lf\n", sum, p->next->avescore);

        p = p->next;

    }

}

void SortNode(CNode\* head) {

    CNode p = \*head, q = \*head;

    int n = 0;

    char numb[100], nam[100];

    while (p->next) {

        n++;

        p = p->next;

    }

    for (int i = 0; i < n - 1; i++,q=\*head) {

        for (int j = 0; j < n - 1 - i; j++,q=q->next) {

            if (q->next->avescore > q->next->next->avescore) {

                strcpy(numb, q->next->num);

                strcpy(q->next->num, q->next->next->num);

                strcpy(q->next->next->num, numb);

                strcpy(nam, q->next->name);

                strcpy(q->next->name, q->next->next->name);

                strcpy(q->next->next->name, nam);

                for (int k = 0; k < 4; k++) {

                    int tmp = q->next->score[k];

                    q->next->score[k] = q->next->next->score[k];

                    q->next->next->score[k] = tmp;

                }

                double tmp2 = q->next->avescore;

                q->next->avescore = q->next->next->avescore;

                q->next->next->avescore = tmp2;

            }

        }

    }

    p = \*head;

    while (p->next) {

        printf("%s %s %.2lf\n", p->next->num, p->next->name,p->next->avescore);

        p = p->next;

    }

}

int main(void) {

    int n;

    CNode head = NULL;

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

        if (n == 0)break;

        switch (n) {

        case 1:CreatNode(&head); break;

        case 2:PrintNode(&head); break;

        case 3:ChangeNode(&head); break;

        case 4:AveNode(&head); break;

        case 5:SumNode(&head); break;

        case 6:SortNode(&head); break;

        }

    }

}

3）测试

（a） 测试数据：

**测试输入1：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

**预期输出1：**

完成了5位同学的成绩输入。

**测试输入2：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

6

**预期输出2：**

完成了5位同学的成绩输入。

2021002 Mike 80.00

2021003 Joe 82.00

2021005 Rose 88.50

2021001 Jack 91.00

2021004 Andy 94.75

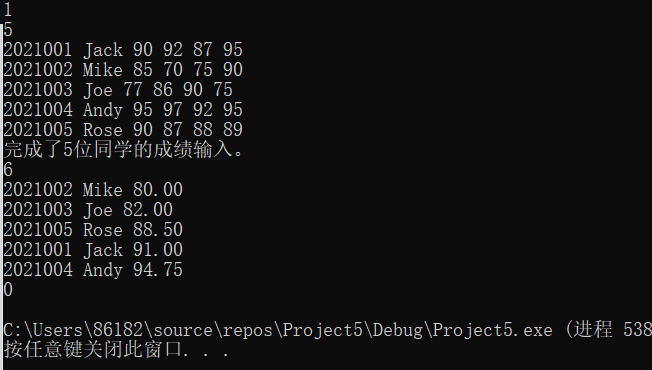
**测试输入3：**

0

**预期输出3：**

（退出）

（b） 对应测试数据的运行结果截图



（3）对程序设计题第（2）题，进一步写出用交换结点指针域的方法升序排序的函数。

**解答：**

1. 解题思路：

更改上一题中的SortNode函数，使用冒泡排序，定义一指针p指向头结点，如果p->next->avescore大于p->next->next->avescore，将这两个结点进行交换：找到第一个结点的前驱，使其指向该结点的后继；使该结点的后继指向第二个结点的后继；最后让第二个结点的后继指向第一个节点。

2）程序清单

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node {

char num[100], name[100];

int score[4];

double avescore;

struct Node\* next;

}Node, \* CNode;

void CreatNode(CNode \*head) {

int qua, i, j;

scanf("%d", &qua);

CNode p , tail;

p = (CNode)malloc(sizeof(Node));

\*head = p;

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

tail = (CNode)malloc(sizeof(Node));

scanf("%s", &tail->num);

scanf("%s", &tail->name);

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

scanf("%d", &tail->score[j]);

tail->avescore = (tail->score[0] + tail->score[1] + tail->score[2] + tail->score[3]) / 4.0;

p->next = tail;

p = tail;

}

tail->next = NULL;

printf("完成了%d位同学的成绩输入。\n",qua);

}

void PrintNode(CNode\* head) {

CNode p = \*head;

while (p->next) {

printf("%s %s ", p->next->num, p->next->name);

for (int i = 0; i < 4; i++)

printf("%d ", p->next->score[i]);

p = p->next;

printf("\n");

}

}

void ChangeNode(CNode\* head) {

CNode p = \*head;

char num[100];

int k = -1;

scanf("%s", &num);

while (p->next) {

if (!strcmp(p->next->num, num)) {

scanf("%d", &k);

break;

}

p = p->next;

}

if (k == 0)

scanf("%s", &p->next->name);

if (k > 0) {

scanf("%d", &p->next->score[k - 1]);

p->next->avescore = (p->next->score[0] + p->next->score[1] + p->next->score[2] + p->next->score[3]) / 4.0;

}

printf("%s %s ", p->next->num, p->next->name);

for (int i = 0; i < 4; i++)

printf("%d ", p->next->score[i]);

printf("\n");

}

void AveNode(CNode\* head) {

CNode p = \*head;

while(p->next) {

printf("%s %s %.2lf\n", p->next->num, p->next->name, p->next->avescore);

p = p->next;

}

}

void SumNode(CNode\* head) {

CNode p = \*head;

while (p->next) {

printf("%s %s ", p->next->num, p->next->name);

double sum = 0;

for (int i = 0; i < 4; i++)

sum += p->next->score[i];

printf("%.0lf %.2lf\n", sum, p->next->avescore);

p = p->next;

}

}

void SortNode(CNode\* head) {

CNode p = \*head, q = \*head;

int n = 0;

while (p->next) {

n++;

p = p->next;

}

for (int i = 0; i < n - 1; i++, q = \*head) {

for (int j = 0; j < n - 1 - i; j++, q = q->next) {

if (q->next->avescore > q->next->next->avescore) {

CNode k = q->next, m = k->next;

q->next = m;

k->next = m->next;

m->next = k;

}

}

}

p = \*head;

while (p->next) {

printf("%s %s %.2lf\n", p->next->num, p->next->name, p->next->avescore);

p = p->next;

}

}

int main(void) {

int n;

CNode head = NULL;

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

if (n == 0)break;

switch (n) {

case 1:CreatNode(&head); break;

case 2:PrintNode(&head); break;

case 3:ChangeNode(&head); break;

case 4:AveNode(&head); break;

case 5:SumNode(&head); break;

case 6:SortNode(&head); break;

}

}

}

3）测试

（a）测试数据：

**测试输入1：**

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

**预期输出1：**

完成了5位同学的成绩输入。

**测试输入2：**

6

**预期输出2：**

2021002 Mike 80.00

2021003 Joe 82.00

2021005 Rose 88.50

2021001 Jack 91.00

2021004 Andy 94.75

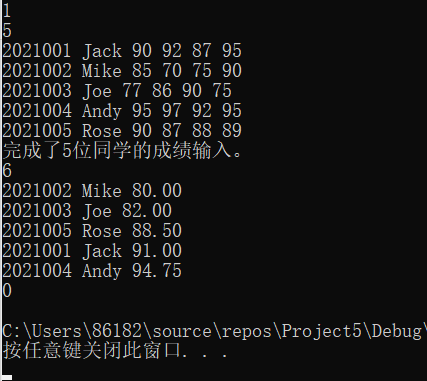
**测试输入3：**

0

**预期输出3：**

（退出）

（b） 对应测试数据的运行结果截图



## 7.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在表达式求值的程序验证题中，我熟悉和掌握了结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。在源程序修改替换题中，我掌握了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。在程序设计题中，我掌握了单链表的各种操作，熟悉了按数据域对结点排序的两种方法。

# 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：

#include <stdio.h>

int main(void)

{

short a=0x253f,b=0x7b7d;

char ch;

FILE \*fp1,\*fp2;

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

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

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

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

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

rewind(fp1); rewind(fp2);

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

putchar(ch);

putchar('\n');

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

putchar(ch);

putchar('\n');

fclose(fp1);

fclose(fp2);

return 0;

}

（1）请思考程序的输出结果，然后通过上机运行来加以验证。

**解答：**

fp1=fopen("d:\\abc1.bin","wb+")表示以二进制形式打开文件，fwrite(&a,sizeof(short),1,fp1)和fwrite(&b,sizeof(short),1,fp1)表示以字节形式向abc1.bin中写入a和b，又因为short类型占两个字节，高8位存放在高字节中，低8位存放在低字节中，所以abc1.bin中的内容应分别是ASCII码为3f、25、7d、7b的字符，即?%}{ 而fprintf(fp2,"%hx %hx",a,b)是将a和b以十六进制short型写入abc2.txt，故其内容应为253f 7b7d。

程序运行结果如下图：



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

**解答：**

abc1.bin中的内容变为?}，因为char类型占1个字节，fwrite在写入时只会保留a和b的低8位，即3f和7d，对应字符为?和}。

程序运行结果如下图：



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

**解答：**

fprintf(fp2,"%d %d",a,b)会将a和b转为十进制整型后再写入。

程序运行结果如下图：



**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。

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/

printf("Can't open %s file!\n",argv[1]);

exit(-1);

}

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

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

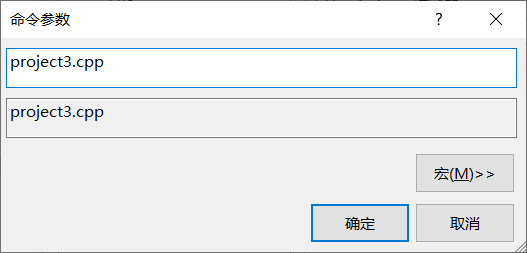
**解答：**

第16行while(ch=fgetc(fp)!=EOF)中会先计算fgetc(fp)!=EOF的值，最后再将该值赋给ch，所以在putchar(ch)时会输出ASCII码为1的字符。正确形式为：while((ch=fgetc(fp))!=EOF)。

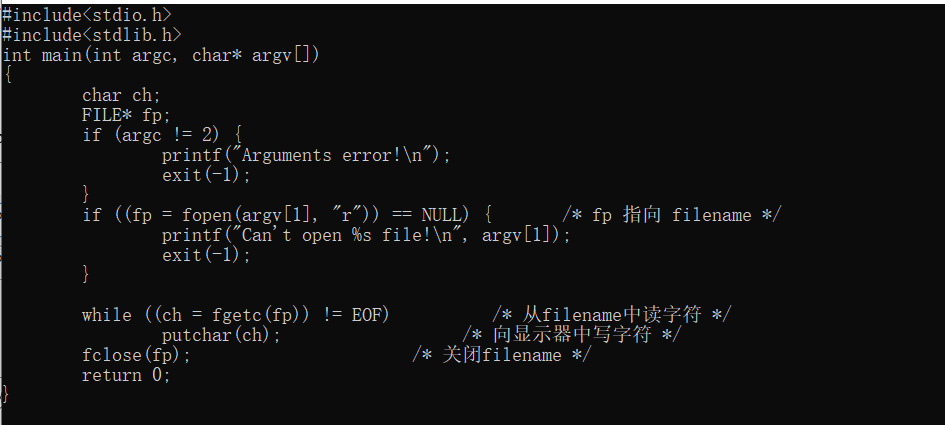
**测试：**

（a） 测试数据：

project3.cpp为该程序所在文件



（b） 对应测试数据的运行结果截图



（2）用输入输出重定向freopen改写main函数。

**解答：**

用freopen函数将标准输入流stdin重定向为参数argv[1]并将其赋值给文件指针fp，这样每次使用scanf函数的时候便会从argv[1]代表的文件中读入数据。

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

FILE\* fp;

if (argc != 2) {

printf("Arguments error!\n");

exit(-1);

}

if ((fp = freopen(argv[1], "r",stdin)) == NULL) { /\* fp 指向 filename \*/

printf("Can't open %s file!\n", argv[1]);

exit(-1);

}

while ((scanf("%c",&ch)) != EOF) /\* 从filename中读字符 \*/

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

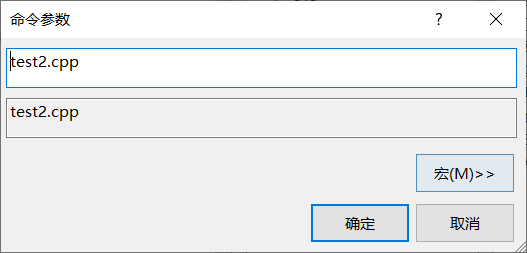
return 0;

}

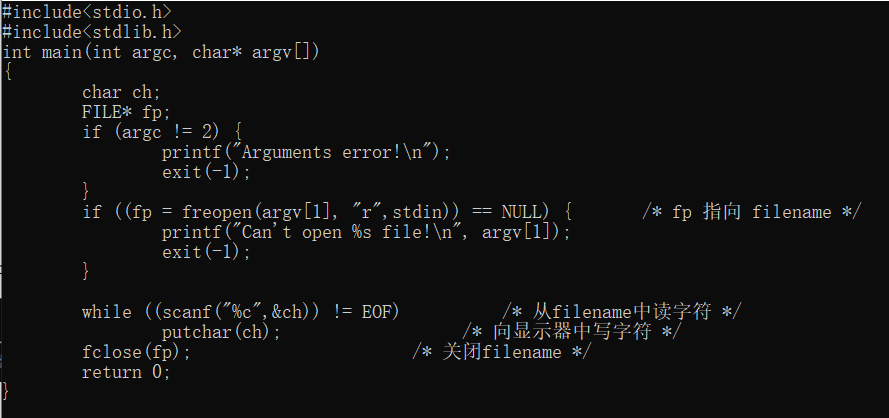
**测试：**

（a） 测试数据：

test2.cpp为该程序所在文件



（b） 对应测试数据的运行结果截图



**8.2.3 程序设计**

（1）本关任务：编写一个程序，用给定的字符串替换文件中的目标字符串，并显示输出替换的个数。

注意：读取的文件路径请使用experiment/src/step8/source.txt

**解答：**

1）解题思路

用两个字符数组str和str1分别存放需要替换的目标字符串和给定的字符串。首先将文件中的内容读取出来存放在text数组中，用strstr函数定位到text中第一次出现str的位置并赋值给字符指针p，调用str\_replace函数进行替换，替换后将将p指向text数组中从p指向的元素开始到最后中第一次出现str的位置再进行替换，直到p为NULL即text中不存在str为止。

2）源程序清单

#define MAX 10000

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void str\_replace(char\* str\_src, int n, char\* str\_copy)

{

int lenofstr;

char\* tmp;

lenofstr = strlen(str\_copy);

if (lenofstr < n)

{

tmp = str\_src + n;

while (\*tmp)

{

\*(tmp - (n - lenofstr)) = \*tmp;

tmp++;

}

\*(tmp - (n - lenofstr)) = \*tmp;

}

else if (lenofstr > n)

{

tmp = str\_src;

while (\*tmp) tmp++;

while (tmp >= (str\_src + n))

{

\*(tmp + (lenofstr - n)) = \*tmp;

tmp--;

}

}

strncpy(str\_src, str\_copy, lenofstr);

}

int main(void) {

FILE\* fp = fopen("experiment/src/step8/source.txt", "r+");

if (fp == NULL) { printf("fuck"); exit(0); }

char str[100], str1[100], text[MAX];

scanf("%s%s", str, str1);

int n = strlen(str);

int count = 0;

if (fgets(text, MAX, fp)) {

char\* p = strstr(text, str);

while (p) {

str\_replace(p, n, str1);

count++;

p += strlen(str1);

p = strstr(p, str);

}

printf("%d\n%s", count, text);

}

return 0;

}

3）测试

(a) 测试数据：

**文件内容1：**

There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!

**测试输入1：**

you they

**预期输出1：**

3

There are moments in life when they miss someone so much that they just want to pick them from theyr dreams and hug them for real!

**文件内容2：**

There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!

**测试输入2：**

express y

**预期输出2：**

0

There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!

**文件内容3：**

May you have enough happiness to make you sweet, enough trials to make you strong, enough sorrow to keep you human, enough hope to make you happy? Always put yourself in others' shoes. If you feel that it hurts you, it probably hurts the other person, too.

**测试输入3：**

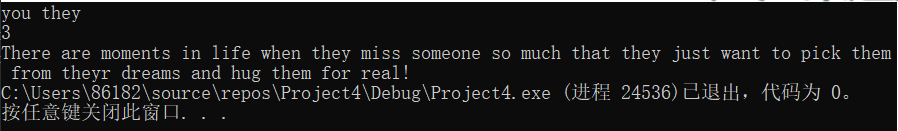
it they

**预期输出3：**

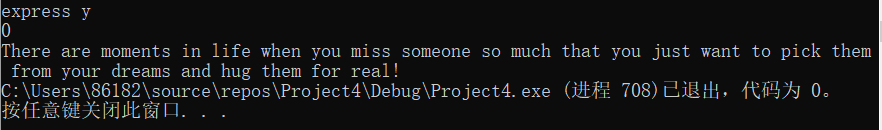
2

May you have enough happiness to make you sweet, enough trials to make you strong, enough sorrow to keep you human, enough hope to make you happy? Always put yourself in others' shoes. If you feel that they hurts you, they probably hurts the other person, too.

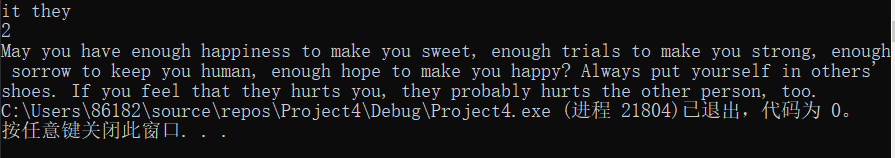
(b) 对应测试数据的运行结果截图



**测试数据一的运行结果截图**



**测试数据二的运行结果截图**



**测试数据三的运行结果截图**

## 8.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在文件类型的程序验证题和源程序修改替换题中，我熟悉了文本文件和二进制文件在磁盘中的存储方式，熟练掌握了流式文件的读写方法，熟悉了输入输出重定向函数freopen的用法。在程序设计题中，我掌握了如何方便快速地对文件中的文本进行查找替换的操作。

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计,北京： 科学出版社,2013

[2] 李开,卢萍,曹计昌. C语言实验与课程设计, 北京：科学出版社,2011