

**课 程 实 验 报 告**

**课程名称： C语言程序设计**

**报告日期： 20170607**

**计算机科学与技术学院**

**目 录**

**实验1 表达式和标准输入与输出实验 1**

**1.1 实验目的 1**

**1.2 实验内容 1**

**1.2.1 源程序改错 1**

**1.2.2 源程序修改替换 2**

**1.2.3 程序设计题 4**

**1.3 自设题 9**

**1.4 实验小结 10**

**实验2 流程控制实验 11**

**2.1 实验目的 11**

**2.2 实验内容 11**

**2.2.1 源程序改错题 11**

**2.2.2 源程序修改替换题 12**

**2.2.3 编程设计题 13**

**2.3 自设题 24**

**2.4 实验小结 26**

**实验3 函数与程序结构实验 27**

**3.1 实验目的 27**

**3.2 实验内容 27**

**3.2.1 源程序改错题 27**

**3.2.2 源程序修改替换题 28**

**3.2.3 跟踪调试题 29**

**3.2.4 编程设计题 30**

**3.2.5 选做s题 35**

**3.3 自设题 36**

**3.4 实验小结 37**

**实验4 编译预处理实验 38**

**4.1 实验目的 38**

**4.2 实验内容 38**

**4.2.1 源程序改错题 38**

**4.2.2 源程序修改替换题 39**

**4.2.3 跟踪调试题 40**

**4.2.4 编程设计题 43**

**4.3 自设题 47**

**4.4 实验小结 47**

**实验5 数组实验 49**

**5.1 实验目的 49**

**5.2 实验内容 49**

**5.2.1 源程序改错 49**

**5.2.2 源程序完善、修改、替换 50**

**5.2.3 跟踪调试源程序 52**

**5.2.4 编程设计题 53**

**5.2.5 选做题 60**

**5.3 自设题 64**

**5.4 实验小结 65**

**实验6 指针实验 66**

**6.1 实验目的 66**

**6.2 实验内容 66**

**6.2.1 源程序改错题 66**

**6.2.2 源程序完善、修改、替换题 67**

**6.2.3 跟踪调试题 70**

**6.2.4 编程设计题 71**

**6.2.5 选做题 82**

**6.3 自设题 84**

**6.4 实验小结 86**

**实验7 结构与联合实验 87**

**7.1 实验目的 87**

**7.2 实验内容 87**

**7.2.1 表达式求值的程序验证题 87**

**7.2.2 源程序修改替换题 89**

**7.2.3 编程设计题 91**

**7.2.4 选做题 99**

**7.3 自设题 110**

**7.4 实验小结 114**

**实验8 文件实验 115**

**8.1 实验目的 115**

**8.2 实验题目及要求 115**

**8.2.1 文件类型的程序验证题 115**

**8.2.2 源程序修改替换题 116**

**8.2.3 编程设计题 118**

**8.3 实验小结 120**

**参考文献 121**

# 实验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；

1 #include<stdio.h>

2 #define PI 3.14159;

3 intmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

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

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

13 /\* for task 2 \*/

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

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

16 s = PI \* r \* r;

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

18 /\* for task 3 \*/

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

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

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

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

23 return 0;

24 }

**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intmain(void)之间应有空格隔开，正确形式为：

int main(void)

3）第10行的格式化输入语句格式错误，正确形式为：

scanf("%d", &f );

4）第11行的右式使表达式进行了赋值转换，使c的形式转化为了int型，正确形式为：

c = 5.0/9.0\*(f-32)；

5）第15行的格式化输入语句中，转换说明%f与参数&r的类型不匹配，正确形式为：

scanf("%lf", &r);

6）第17行的格式化输出中，不需要用到取地址符号，并且s为double型变量，严格输出格式应为lf，正确形式为：

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

7)第20行的输入格式中，格式字符串“%x”需要类型“unsigned int ”的参数，但k,p均为短整型，所以正确形式为：

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

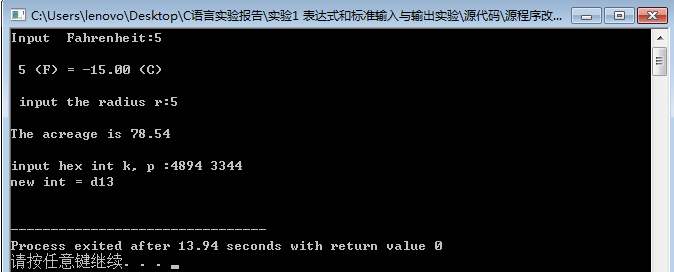
8）第21行变量newint未进行定义，且根据题意，(k&0xff00)的结果应该是进行右移，正确形式为：

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

9）第22行在对newint进行输出时，短整形不应用%x，正确形式为：

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

（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>

int 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);

return 0;

}

**解答：**

方法一：根据对一个数字进行两次相同的按位加运算结果不变的规则，通过按位加位运算对数据进行处理，替换后的程序如下所示：

#include<stdio.h>

int main( )

{

int a, b;

printf(“Input two integers:”);

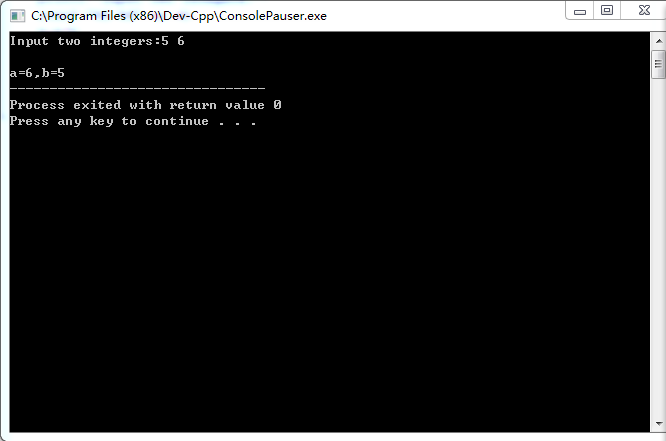
scanf(“%d %d”,&a,&b);

a=a^b; b=a^b; a=a^b;

printf(“\na=%d,b=%d”,a,b);

return 0;

}



方法二：把a,b看做坐标轴上的点，以距离进行运算，替换后的程序如下所示：

#include<stdio.h>

int main( )

{

int a, b;

printf("Input two integers:");

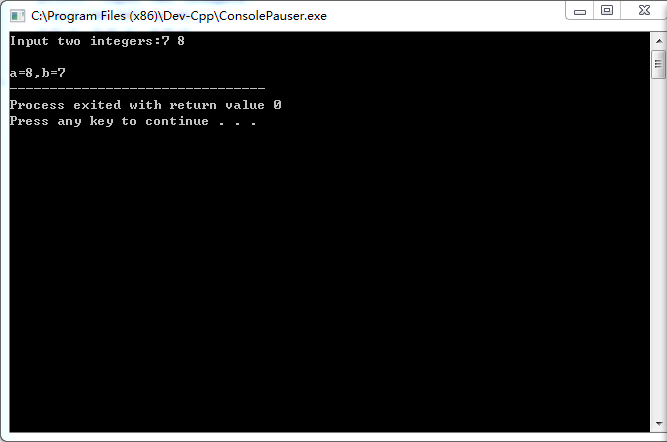
scanf("%d %d",&a,&b);

a=b-a; b=b-a; a=b+a;

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

return 0;

}



### 1.2.3 程序设计题

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

**解答：**

1）算法流程如图1.1所示。

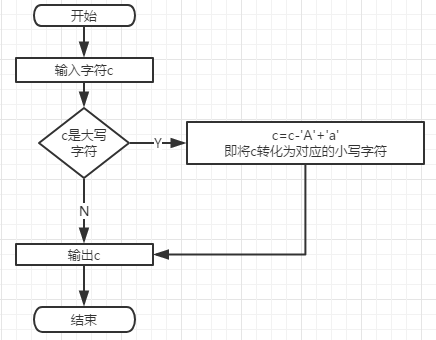


图1-1 编程设计题1的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

char c;

scanf("%c",&c);

printf("%c",(c>='A'&&c<='Z')?(c-'A'+'a'):c);

return 0;

}

3）测试

（a） 测试数据：

选取不同大小写的三组数据，如表1-1所示。

表1-1 编程设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| c |
| 用例1 | C | c | 如图1-2 |
| 用例2 | c | c | 如图1-3 |
| 用例3 | Z | z | 如图1-4 |

（b）对应测试测试用例1的运行结果如图1-2所示。

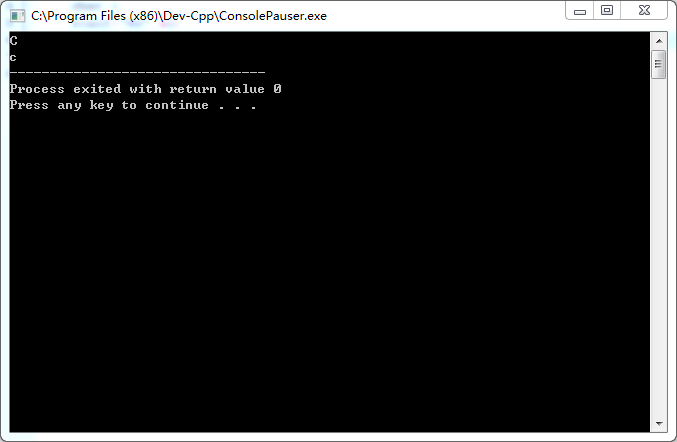


图1-2 编程设计题1的测试用例1的运行结果

对应测试测试用例2的运行结果如图1-3所示：

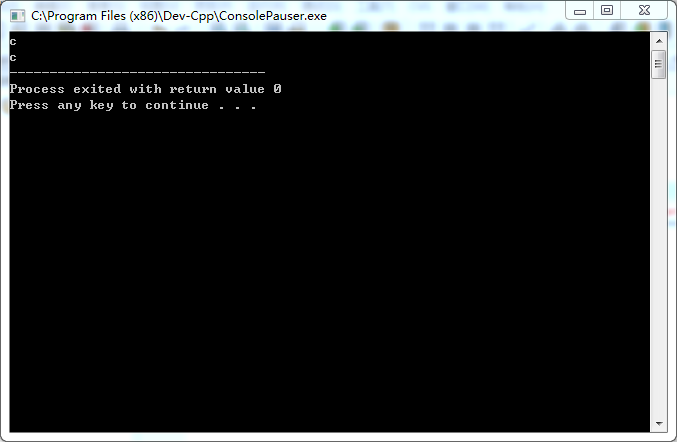


图1-3 编程设计题1的测试用例2的运行结果

对应测试测试用例3的运行结果如图1-4所示：

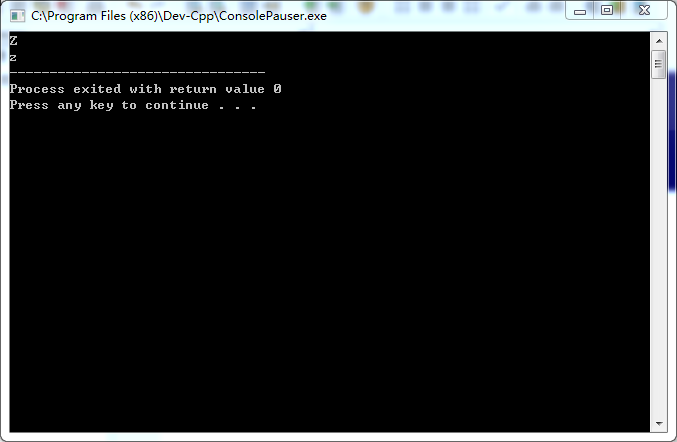


图1-4 编程设计题1的测试用例3的运行结果

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。

**（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）算法流程如图1.5所示。

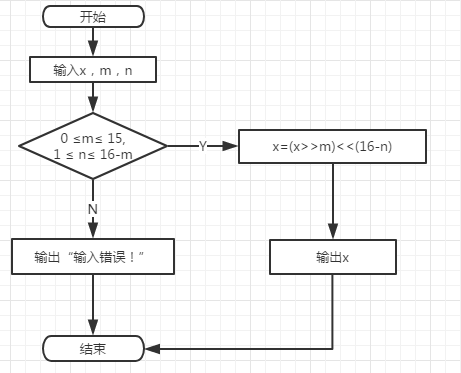


图1-5 编程设计题2的程序流程图

2）程序清单

#include<stdio.h>

int main()

{

unsigned short x,m,n;

printf("输入x、m（0~15）和n（1~16-m）：\n ");

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

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

{

x=(x>>m)<<(16-n);//先右移m位，在左移16-n位

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

}

else

printf("输入错误！\n");

return 0;

}

3）测试

（a） 测试数据：

选择符合m、n取值范围的两组数据以及一组不符合取值范围的数据进行程序运行。如表1-2所示。

表1-2 编程题2的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理论结果 | 运行结果 |
| x | m | n |
| 用例1 | 12（0000 0000 0000 1100） | 0 | 4 | 49152  （即1100 0000 0000 0000） | 如图1-6 |
| 用例2 | 6（0000 0000 0000 0110） | 16 | 1 | 输入错误！（m值超范围） | 如图1-7 |
| 用例3 | 6（0000 0000 0000 0110） | 13 | 5 | 输入错误！（n值超范围） | 如图1-8 |
| 用例4 | 6（0000 0000 0000 0110） | 0 | 5 | 12288  （即0011 0000 0000 0000） | 如图1-9 |

（b） 对应测试测试用例1的运行结果如图1-6所示。

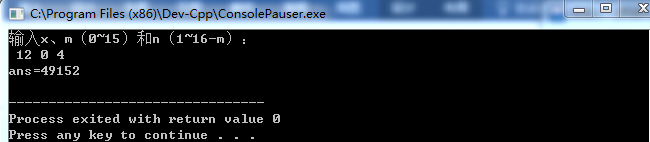


图1-6 编程设计题2的测试用例1的运行结果

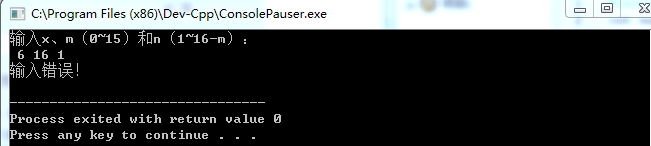
对应测试测试用例2的运行结果如图1-7所示。

图1-7 编程设计题2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1-8所示。

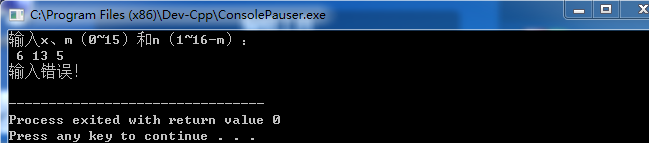


图1-8 编程设计题2的测试用例3的运行结果

对应测试测试用例3的运行结果如图1-9所示。

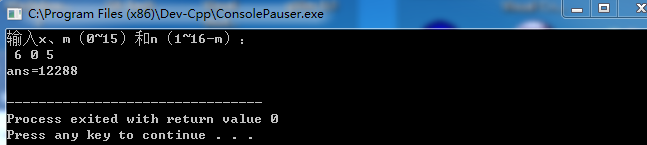


图1-9 编程设计题2的测试用例4的运行结果

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。

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

**解答：**

1）算法流程如图1.10所示。

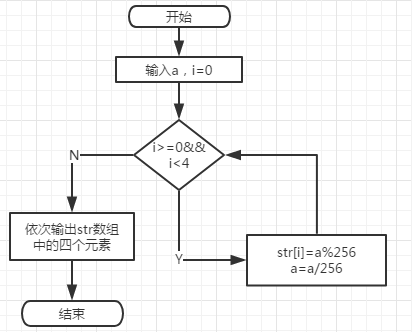


图1-10 编程设计题3的程序流程图

2）程序清单

#include<stdio.h>

int main()

{

unsigned long a;

scanf("%ul",&a);

int str[4];

for(int i=0;i<4;i++) //判断i是否符合条件

{

str[i]=a%256;

a=a/256;

}

printf("%d.%d.%d.%d",str[0],str[1],str[2],str[3]);

}

3）测试

1. 测试数据：：

选取不同的三组数据，如表1-3所示。

表1-3 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 |
| a |
| 用例1 | 45678（0000 0000 0000 0000 1011 0010 0110 1110） | 110.178.0.0 | 如图1-11 |
| 用例2 | 8564538（0000 0000 1000 0010 1010 1111 0011 1010） | 58.175.130.0 | 如图1-12 |
| 用例3 | 4863545（0000 0000 0100 1010 0011 0110 0011 1001） | 57.54.74.0 | 如图1-13 |

1. 对应测试测试用例1的运行结果如图1-11所示。

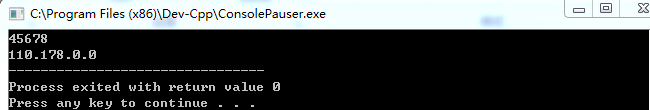


图1-11 编程设计题3的测试用例1的运行结果

对应测试测试用例2的运行结果如图1-12所示。

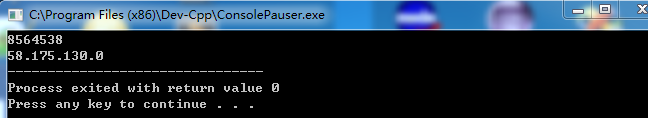


图1-12 编程设计题3的测试用例2的运行结果

对应测试测试用例3的运行结果如图1-13所示。

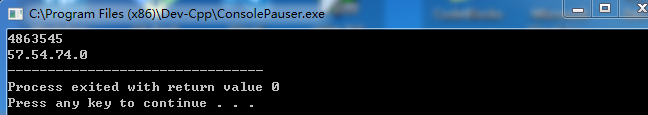


图1-13 编程设计题3的测试用例3的运行结果

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。

## 1.3 自设题

**（1）**自设实验题目： 观察各种输出格式下的输出结果。

**（2）**实验目的：通过设计实验程序，理解并熟悉掌握各种输出格式。

**（3）**实验程序：

#include<stdio.h>

int main()

{

int k=1234;

double f=123.456;

printf("%d,%6d,%2d,%06d,%.6d\n",k,k,k,k,k);

printf("%f,%12f,%12.6f\n",f,f,f);

printf("%2.6f,%.0f,%12.2f\n",f,f,f);

printf("%g,%10g\n",f,f);

printf("%e,%3.8e,%13.2e,%.0e\n",f,f,f,f);

}

**（4）**实验用例：int型k=1234和double型f=123.456;

对应测试试验用例的运行结果如图1-14所示：

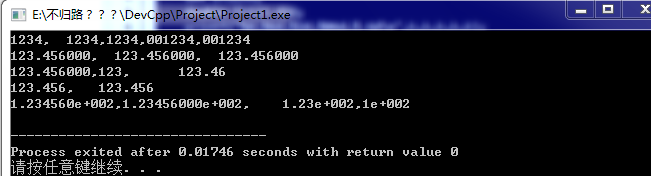


图1-14 自设题的测试运行结果

**（5）**实验结论：在书写输出格式时，应注意输出类型以及是否指定宽度，各种的情况下的输出结果可能会有差异。

## 1.4 实验小结

**1.4.1** 本次试验中主要在第一道改错题中遇到了一些细节问题，有些细节虽然编译器没有提示有error但是却给出了warning，最后是根据警warning的提示对程序进行了修改,这提醒了我在编写程序是应注意细节问题，每次都要达到0 error,0 warning的结果。

**1.4.2** 在自设题中列举了几种输出宽度和不指定宽度情况下的输出结果，以此进行观察，初步了解了几种格式化输出的书写方式，以后在程序编写过程中可根据需求进行不同的输出。

# 实验2 流程控制实验

## 2.1 实验目的

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

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用Turbo C 2.0集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 2.2 实验内容

### 2.2.1 源程序改错题

下面是计算s=n!的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。例如，8！=40320。

1 #include <stdio.h>

2 void main(void)

3 {

4 int i,n,s=1;

5 printf("Please enter n:");

6 scanf("%d",n);

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

8 s=s\*i;

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

10 }

**解答：**

（1）错误修改：

1) 第2行的程序返回值类型有错，应该为‘int’型，正确形式为：

int main(void)

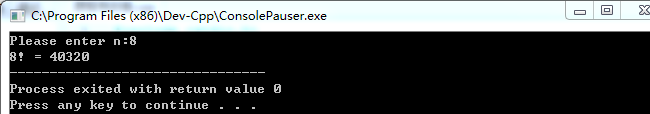
2) 第6行的标准输入格式有误，应该添加取地址符，正确形式为：

scanf("%d",&n);

3) 第7行的for循环语句的条件书写有误，应为分号，正确形式为：

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

（2）错误修改后运行结果：

****

### 2.2.2 源程序修改替换题

（1）修改第1题，分别用while和do-while语句替换for语句。

**解答：**，

1. 用while语句替换for语句，替换后的程序如下所示,运行结果如图2-1所示：

#include <stdio.h>

int main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d",&n);

i=1;

while(i<=n)

{

s=s\*i;

i++;

}

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

}

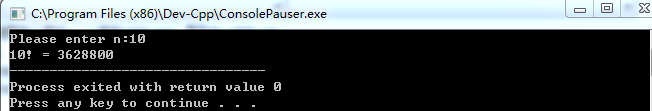


图2-1 用while语句替换for语句后的运行结果

2.用do-while语句替换for语句，替换后的程序如下所示,运行结果如图2-2所示：

#include <stdio.h>

int main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d",&n);

i=1;

do{

s=s\*i;

i++;

}while(i<=n);

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

}

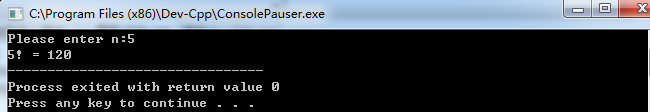


图2-2 用do-while语句替换for语句后的运行结果

1. 修改第1题，输入改为“整数S”，输出改为“满足n！≥S的最小整数n”。例如输入整数40310，输出结果为n=8.

**解答：**

根据题意修改判断语句，替换后的程序如下所示，运行结果如图2-3所示：

#include <stdio.h>

int main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d",&n);

i=0;

while(s<=n)

{

i++;

s=s\*i;

}

printf("%d",i);

}

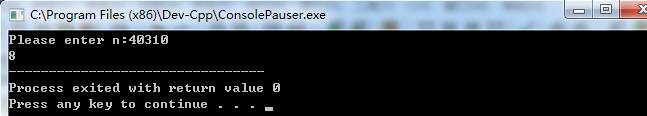


图2-3按题意修改后程序的运行结果

### 2.2.3 编程设计题

**（1）**打印如下杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 、 84 126 126 84 36 9 1

每个数据值可以由组合计算（表示第i行第j列位置的值），而的计算如下：

 (i=0,1,2,…)

 (j=0,1,2,3,…,i)

本程序中为了打印出金字塔效果，要注意空格的数目。一位数之间是3个空格，两位数之间有2个空格，3位数之间只有一个空格，程序编制过程中要注意区分。

**解答：**

1. 算法流程如图2.4所示。

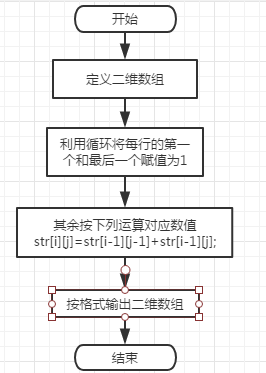


图2-4 编程设计题1的程序流程图。

2）程序清单

#include<stdio.h>

int main()

{

int str[100][100];

int i,j;

int n;

scanf("%d",&n);

for(i=0,j=0;i<n;i++)str[i][j]=1;

for(i=0;i<n;i++) {for(j=0;j<=i;j++) if(i==j) str[i][j]=1;}

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

{

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

str[i][j]=str[i-1][j-1]+str[i-1][j];

}

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

{

for(int k=0;k<2\*(n-i-1);k++)printf(" ");

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

{

if(j==0)printf("%d",str[i][j]);

else printf("%4d",str[i][j]);

}

}

}

3）测试

（a） 测试数据：

随机选取一个整数，如表2-1所示。

表2-1 编程设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| N |
| 用例1 | 4 | 4行杨辉三角 | 如图2-5 |
| 用例2 | 12 | 12行杨辉三角 | 如图2-6 |

（b） 对应测试测试用例1的运行结果如图2-5所示。

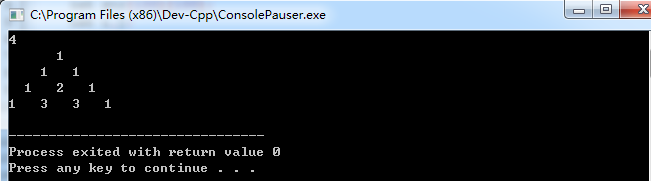


图2-5 编程设计题1的测试用例一的运行结果

对应测试测试用例2的运行结果如图2-6所示。

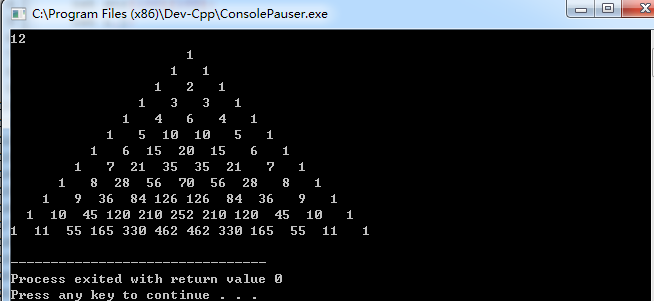


图2-6 编程设计题1的测试用例二的运行结果

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

**解答：**

1. 算法流程如图2.7所示。

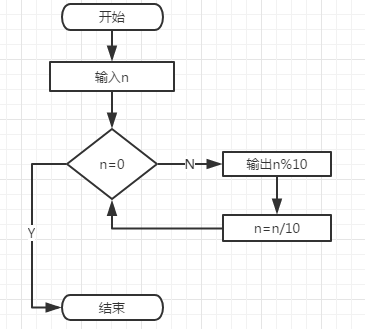


图2-7 编程设计题2的程序流程图

2）程序清单

#include<stdio.h>

int main()

{

int n;

scanf("%d",&n);

while(n!=0)

{

printf("%d",n%10);

n=n/10;

}

}

3）测试数据

（a） 随机输入数据，如表2-2所示。

表2-2 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| n |
| 用例1 | 1234 | 4321 | 如图2-8 |
| 用例2 | 145 | 541 | 如图2-9 |
| 用例3 | 1 | 1 | 如图2-10 |

（b） 对应测试测试用例1的运行结果如图2-8所示。

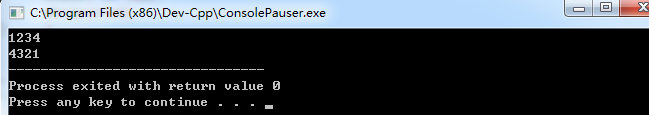


图2-8 编程设计题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图2-2所示。

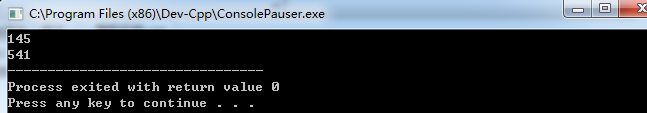


图2-9 编程设计题2的测试用例二的运行结果

对应测试测试用例3的运行结果如图2-3所示。

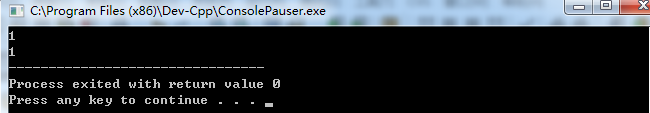


图2-10 编程设计题2的测试用例三的运行结果

（3）求方程的近似根。（作业提交系统归档编号：20170330091057 题号1）

**解答：**

1） 算法流程如图2.11所示。

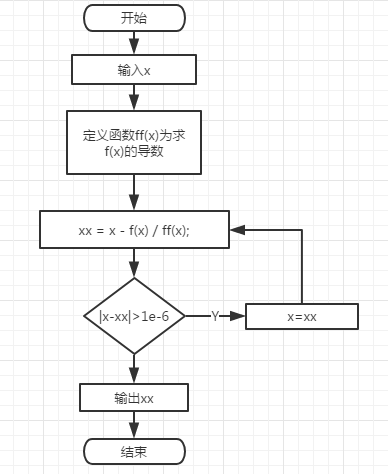


图2-11 编程设计题3的程序流程图

2）程序清单

#include<stdio.h>

#include<math.h>

/\* 函数原型：计算f(x) \*/

double f(double);

/\* 函数原型：计算f(x)的导数 \*/

double ff(double);

int main()

{

double x = 3;

double xx = x - f(x) / ff(x);

while (fabs(xx - x)>1e-6)

{

x = xx;

xx = x - f(x) / ff(x);

}

printf("%lf\n", xx);

}

/\*函数介绍：计算f(x)

输入参数：double x

返 回 值：f(x)\*/

double f(double x)

{

return (3 \* pow(x, 3) - 4 \* pow(x, 2) - 5 \* x + 13);

}

/\*

函数介绍：计算f(x)

输入参数：double x

返 回 值：f(x)的导数

\*/

double ff(double x)

{

return (9 \* pow(x, 2) - 8 \* x - 5);

}

3）测试

（a）

表2-3 编程设计题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试 | 理 论 结 果 | 运 行 结 果 |
|
|  | -1.548910 | 如图2-12 |

（b）

图2-12 编程设计题3的运行结果

（4）个人所得税。（作业提交系统归档编号：20170330091057 题号4）

**解答：**

1) 解题思路：

1.输入x。

2.定义函数If（Switch）。

3.判断x的值，如果x <= 1000((int)x / 1000=0),转3.1；如果1000<x<= 2000或((int)x / 1000=1),转3.2；如果2000<x<=3000((int)x / 1000=2),转3.3；如果3000<x <= 4000((int)x / 1000=3),转3.4；如果4000<x <= 5000((int)x / 1000=4),转3.4；如果x >5000((int)x / 1000=其他值),转3.5；

3.1 t=0;转4.

3.2 t = (x - 1000)\*0.05; 转4.

3.3 t = (x - 2000)\*0.1 + (2000 - 1000) \* 0.05; 转4.

3.4 t = (x - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000) \* 0.05; 转4.

3.5 t = (x - 4000)\*0.2 + (4000 - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000) \* 0.05; 转4.

3.6 t = (x - 5000)\*0.25 + (5000 - 4000)\*0.2 + (4000 - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000) \* 0.05; 转4.

4.输出t。

5.结束。

2）程序清单

<1>if语句

#include<stdio.h>

#include<stdlib.h>

double If(double);

int main()

{

double x,tax;

while (1)

{

scanf("%lf", &x);

if (x != 0)

{

tax = If(x);

printf("%lf\n", tax);

}

else break;

}

system("pause");

}

double If(double x)

{

double t;

if (x <= 1000)t = 0;

else if (x <= 2000)t = (x - 1000)\*0.05;

else if (x <= 3000)t = (x - 2000)\*0.1 + (2000 - 1000) \* 0.05;

else if (x <= 4000)t = (x - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000) \* 0.05;

else if (x <= 5000)t = (x - 4000)\*0.2 + (4000 - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000) \* 0.05;

else t = (x - 5000)\*0.25 + (5000 - 4000)\*0.2 + (4000 - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000) \* 0.05;

return t;

}

<2>switch语句

#include<stdio.h>

#include<stdlib.h>

double Switch(double);

int main()

{

double x,tax;

while (1)

{

scanf("%lf", &x);

if (x != 0)

{

tax = Switch(x);//tax = If(x);

printf("%lf\n", tax);

}

else break;

}

system("pause");

}

double Switch(double x)

{

double t; int a;

a = (int)x / 1000;

switch (a)

{

case 0:t = 0; break;

case 1:t = (x-1000)\*0.05; break;

case 2:t = (x - 2000)\*0.1 + (2000 - 1000) \* 0.05; break;

case 3:t = (x - 3000)\*0.15 + (3000 - 2000)\*0.1 +

(2000 - 1000) \* 0.05; break;

case 4:t = (x - 4000)\*0.2 + (4000 - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000- 1000) \* 0.05; break;

default:t = (x - 5000)\*0.25 + (5000 - 4000)\*0.2 + (4000 - 3000)\*0.15 + (3000 - 2000)\*0.1 + (2000 - 1000)\* 0.05; break;

}

return t;

}

3）测试数据

（a） 随机输入数据，如表2-4所示。

表2-4 编程设计题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| x |
| 用if语句 | 2500 | 100.000000 | 如图2-14 |
| 4000 | 300.000000 |
| 0 | 程序结束运行 |
| 用switch语句 | 1000 | 0.000000 | 如图2-15 |
| 3500 | 225.000000 |
| 5500 | 625.000000 |

（b） 对应测试测试用if语句的运行结果如图2-14所示。

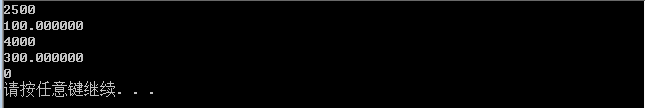


图2-14 编程设计题4的测试用if语句的运行结果

对应测试测试用switch语句的运行结果如图4-2所示。

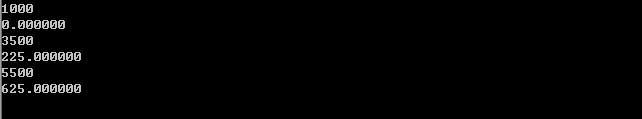


图2-15 编程设计题4的测试用switch语句的运行结果

（5）空格处理。（作业提交系统归档编号：20170330091057 题号5）

**解答：**

1） 算法流程如图2.16所示。

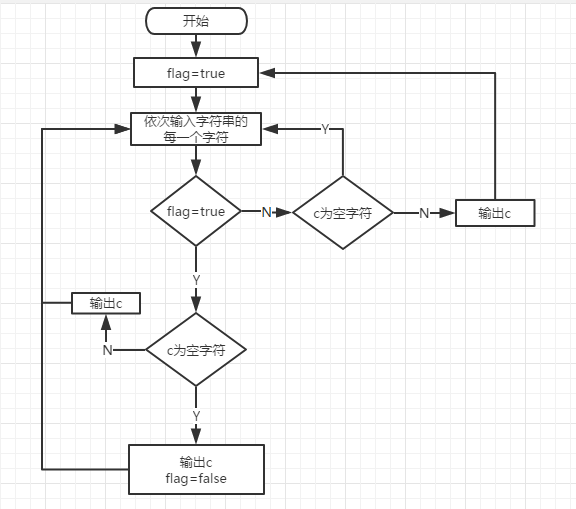


图2-16 编程设计题5的程序流程图

2）程序清单

#include<stdio.h>

#include<stdbool.h>

int main()

{

int N;

scanf("%d", &N);

getchar();

for (int k = 0; k<N; k++)//控制输入N组数据

{

int i = 0;

bool flag = true;

char c;

while ((c = getchar()) != '\n')//循环输入单个字符c

{

if (flag == true && c == ' ')//3.1

{

putchar(c);

flag = false;

}

if (flag == true && c != ' ')//3.2

{

printf("%c", c);

}

if (flag == false && c != ' ')//3.3

{

printf("%c", c);

flag = true;

}

else continue;

}

printf("\n");

}

}

3）测试数据

（a） 随机输入字符，如表2-5所示。

表2-5 编程题5的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | 理 论 结 果 | 运 行 结 果 |
| N | c |
| 用例 | 3 | Hello world ! | hello world ! | 如图2-17 |
| Hello world ! | hello world ! |
| Hello world ! | hello world ! |

（b） 对应测试测试用例的运行结果如图2-17所示。

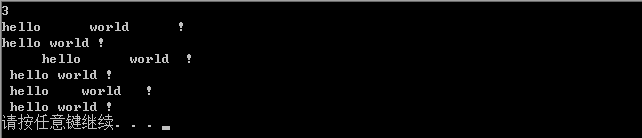


图2-17 编程设计题5的测试用例的运行结果

## 2.3 自设题

**（1）**自设实验题目：打印边长为N(0<N<10)的菱形

**（2）**实验目的：掌握多重循环问题。

**（3）**实验程序：

#include<stdio.h>

#include<stdlib.h>

int main()

{

int N,i,j;

printf("请输入边长N："); scanf\_s("%d", &N);

printf("\n");

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

{

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

printf(" ");

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

printf("%c ", '\*');

printf("\n");

}

for (i = N - 1; i > 0; i--)

{

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

printf(" ");

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

printf("%c ", '\*');

printf("\n");

}

system("pause");

return 0;

}

**（4）**实验用例：

表2-6 自设题的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| N |
| 用例一 | 5 | 边长为5的菱形 | 如图2-18 |
| 用例二 | 7 | 边长为7的菱形 | 如图2-19 |
| 用例三 | 10 | 边长为10的菱形 | 如图2-20 |

（b） 对应测试测试用例的运行结果如图2-18所示。

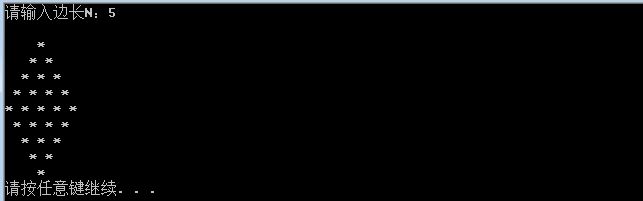


图2-18 自设题测试用例一的运行结果

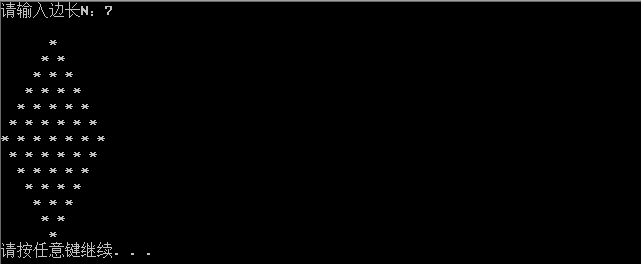
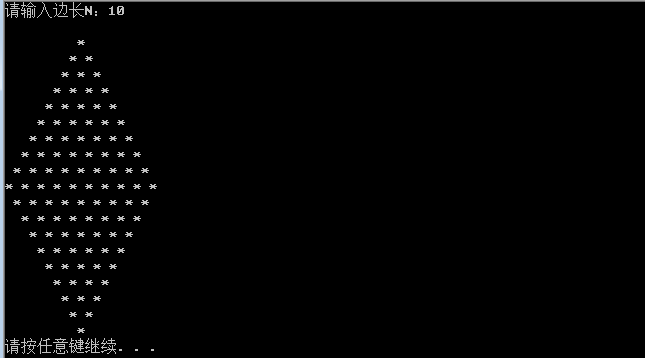
对应测试测试用例的运行结果如图2-19所示。

图2-19 自设题测试用例二的运行结果

对应测试测试用例的运行结果如图2-20所示

图2-20 自设题测试用例三的运行结果

**（5）**实验结论：处理多重循环是最重要的是判断终止条件，是正确循环。

## 2.4 实验小结

**2.4.1** 多重循环中终止条件的判断或者是i值得范围十分重要，需要逻辑清晰，在检测程序是输入边界值进行判断。

**2.4.2** 在循环中应准确掌握return语句，break语句和continue语句，对于循环的运行终止等有很大的帮助。

**2.4.3** 在程序中语句的前后顺序也会有很重要的作用。例如在编写空格处理的过程中，最初程序在输出上有很大的错误，在进行了各种尝试和调试后，发现只需调换几个if语句的顺序即可。

**2.4.4** 递归在编写程序时可以简化程序，但在处理大数据时运行较慢。

**2.4.5**在程序需要外部输入时，须注意回车符是否造成影响，必要时加上getchar语句或者在scanf语句中加入空格以取消回车符对程序的影响。

# 实验3 函数与程序结构实验

## 3.1 实验目的

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

（2）熟悉和掌握不同存储类型变量的使用。

（3）熟悉多文件编译技术。

## 3.2 实验内容

### 3.2.1 源程序改错题

下面是计算s=1!+2!+3!+…+n!的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

1 #include "stdio.h"

2 void main(void)

3 {

4 int k;

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

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

7 }

8 long sum\_fac(int n)

9 {

10 long s=0;

11 int i;

12 long fac;

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

14 fac\*=i;

15 s+=fac;

16 return s;

17 }

**解答：**

1. 错误修改：
2. 第2行，因程序后面直接调用函数sum\_fac而未进行声明，正确形式

为：

long sum\_fac(int);

2）第13行的for循环语句根据逻辑应当加一对括号，正确形式为：

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

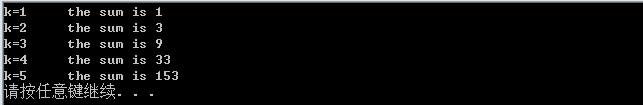
{

fac \*= i;

s += fac;

}

1. 错误修改后运行结果：



### 3.2.2 源程序修改替换题

（1）修改第1题中sum\_fac函数，使其计算量最小。

**解答：**

为使函数计算量最小，使用两个静态变量，替换后的程序如下所示：

#include "stdio.h"

#include<stdlib.h>

long sum\_fac(int);

int main(void)

{

int k;

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

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

system("pause");

}

long sum\_fac(int n)

{

static long s=0;

static long fac=1;

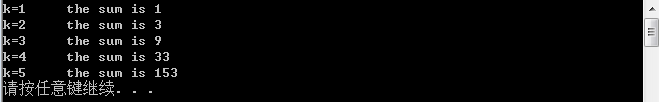
fac\*=n;

s+=fac;

return s;

}

运行结果如下图：



（2）修改第1题中sum\_fac函数，计算。

**解答：**

替换后的程序如下所示：

#include "stdio.h"

long double sum\_fac(int);

int main(void)

{

int k;

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

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

}

long double sum\_fac(int n)

{

static long double s=0;

static long double fac=1;

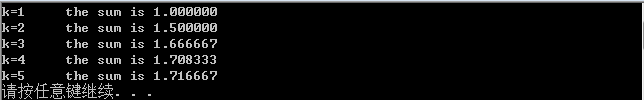
fac\*=1/(long double)n;

s+=fac;

return s;

}

运行结果如下图：



### 3.2.3 跟踪调试题

计算fabonacci数列前n项和的程序如下：

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

void main(void)

{

int i,k;

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

scanf("%d",&k);

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

sum+=fabonacci(i);

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

}

}

long fabonacci(int n)

{

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

return 1;

else

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

}

单步执行程序，观察p,i,sum,n值。

（1）刚执行完scanf("%d",&k);语句，p,i值是多少？

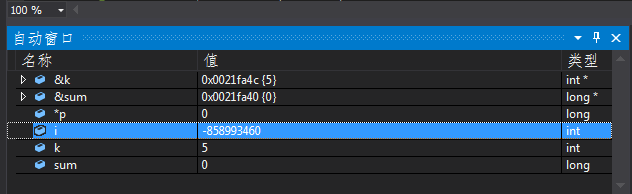
（2）从fabonacci函数返回后光条停留在哪个语句上？

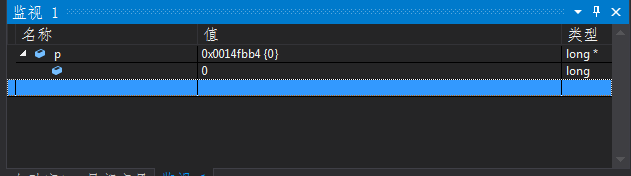
（3）进入fabonacci函数，watch窗口显示的是什么？

（4）当i=3，从调用fabonacci函数到返回，n值如何变化？

**解答**

1. p=0x00014fbb4{0} i=-858993460。如下图所示。





1. 停留在sum+=fabonacci(i)。
2. n的值。
3. 3 2  1 3

### 3.2.4 编程设计题

（1）编程让用户输入两个整数，计算两个数的最大公约数并且输出之（要求用递归函数实现求最大公约数）。同时以单步方式执行该程序，观察递归过程。

**解答：**

1） 算法流程如图3-1所示。

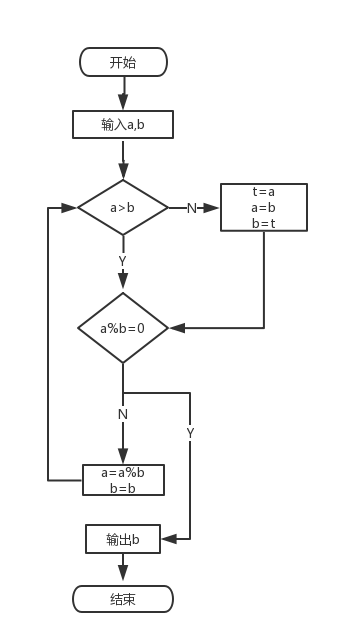


图3-1 编程设计题1的程序流程图

2）程序清单

#include<stdio.h>

int max(int, int);

int main()

{

int a, b;

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

{

if (a != 0)

{

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

}

else break;

}

}

int max(int a, int b)

{

int t;

if (a<b)

{

t = a; a = b; b = t;//交换a,b的值

}

if (a%b == 0)return b;

else return max(a%b, b);

}

3）测试

（a） 测试数据：

表3-1 编程设计题1的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程序输入 | | 理 论 结 果 | 运 行 结 果 |
| a | b |
| 用例1 | 4 | 4 | 4 | 如图3-2 |
| 用例2 | 15 | 5 | 5 | 如图3-2 |
| 用例3 | 27 | 36 | 9 | 如图3-2 |

（b） 对应测试测试用例1用例2用例3的运行结果如图3-2所示。

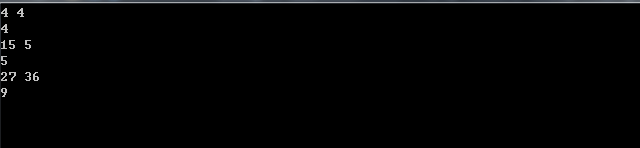


图3-2 编程设计题1的测试用例一用例二用例三的运行结果

（2）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。

编写一个程序证明对于在符号常量BEGIN和END之间的偶数这一猜测成立。例如，如果BEGIN为10，END为20，程序的输出应为：

GOLDBACH'S CONJECTURE:

Every even number n>=4 is the sum of two primes.

10=3+7

12=5+7

……

20=3+17

**解答：**

1. 算法流程如图3-3所示。

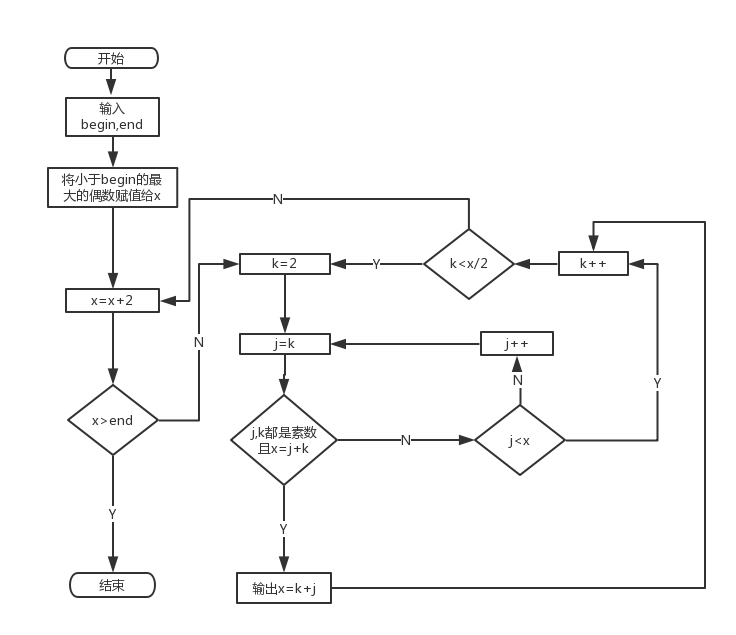


图3-3 编程设计题2的程序流程图

2）程序清单

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

int IsPrime(int m)

{

int k, limit;

if (m == 2)return 1;

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

limit = m / 2;

for (k = 3; k <= limit; k += 2)

if (!(m%k))return 0;

return 1;

}

int main(void)

{

int begin, end, x, k, j;

bool flag;

while (scanf("%d %d", &begin, &end) && begin != 0)

{

x = (begin % 2 == 0) ? begin - 2 : begin - 1;

while (begin >= 0 && end >= begin)

{

x = x + 2;

if (x > end)

{

printf("\n"); break;

}

flag = true;

for (k = 2; k <= x / 2&&flag==true; k++)

{

for (j = k; j <= x&&flag==true; j++)

{

if (IsPrime(j) && IsPrime(k) && x == (j + k))

{

printf("%d=%d+%d\n", x, k, j);flag = false;

}

}

}

}

}

system("pause");

}

3）测试

（a） 测试数据：

表3-2 编程设计题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程序输入 | | 理 论 结 果 | 运 行 结 果 |
| begin | end |
| 用例1 | 2 | 6 | 4=2+2  6=3+3 | 图3-4 |
| 用例2 | 10 | 20 | 10=3+7  12=5+7  14=3+11  16=3+13  18=5+13  20=3+17 | 图3-4 |
| 用例3 | 5 | 9 | 6=3+3  8=3+5 | 图3-4 |

（b） 对应测试测试用例1用例2用例3的运行结果如图3-4所示。

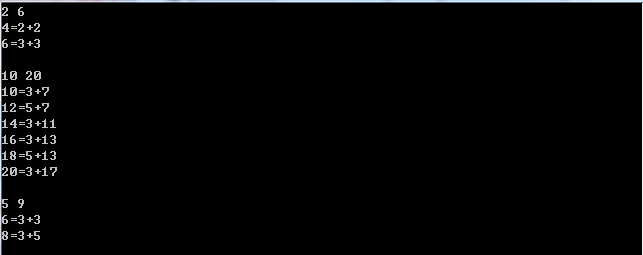


图3-4 编程设计题2的测试用例一用例二用例三的运行结果

### 3.2.5 选做s题

1、设file1.c如下：

#include <stdio.h>

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

void main(void)

{

x=10;

y=20;

ch=getchar();

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

func1();

}

file2.c如下：

extern int x,y; /\* 外部变量的引用性说明 \*/

extern char ch; /\* 外部变量的引用性说明 \*/

void func1(void)

{

x++;

y++;

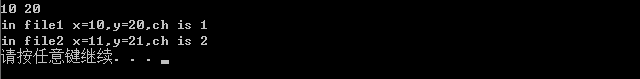
ch++;

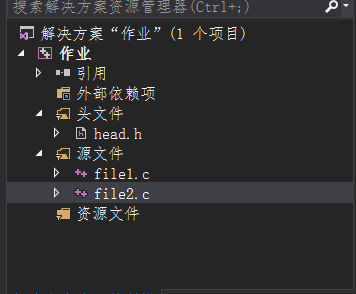
printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

试用TCC进行多文件编译和链接。然后在DOS环境下运行生成的可执行文件。

**解答：**

****



## 3.3 自设题

**（1）** 自设实验题目：编写函数交换两个变量的值

**（2）** 实验目的：通过设计实验程序，理解在值传递中形参的改变不会引起实参的改变。若是需要改变数值则要用到指针传递。

**（3）**实验程序：

#include<stdio.h>

#include<stdlib.h>

void swapdate1(int, int);

void swapdate2(int\*, int\*);

int main()

{

int x, y;

scanf("%d %d", &x, &y);

printf("main x=%d,y=%d\n", x, y);

swapdate1(x, y);

printf("main x=%d,y=%d\n", x, y);

swapdate2(&x, &y);

printf("main x=%d,y=%d\n", x, y);

system("pause");

return 0;

}

void swapdate1(int x, int y)

{

int temp;

temp = x; x = y; y = temp;

printf("swapdate1:x=%d,y=%d\n", x, y);

}

void swapdate2(int \*x, int \*y)

{

int temp;

temp = \*x; \*x = \*y; \*y = temp;

printf("swapdate1:x=%d,y=%d\n", \*x, \*y);

}

**（4）**实验用例：x=10,y=20

对应测试用例的运行结果如图3-5所示。

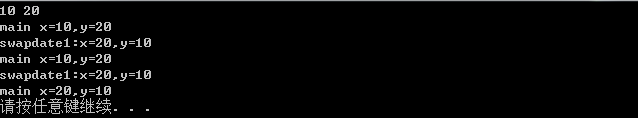


图3-5 自设题的测试用例的运行结果

**（5）**实验结论：

函数参数在按值传递情况下，被调函数的形参作为被调函数的局部变量看待，在内存中分配相应的空间存放主调函数传入的数据，形参和实参有各自独立的存储空间，函数调用时，将实参的值传递给对应的形参，形参的改变不会引起实参的改变。

指针变量从本质上讲就是存放变量地址的一个变量，可以改变指针变量所指向的地址和其指向的地址中所存放的数据。指针传递参数本质上也是值传递的方法，但它所传递的是一个地址值，其特点是可以通过形参改变实参的值。

## 3.4 实验小结

**3.4.1** 在同学的帮助下学会了调试，以前调试程序经常是自己在各种地方加上printf语句进行输出，感觉调试真的是必不可少的一项技能。

**3.4.2** 在本次实验中，多次运用到了递归的思想，感觉递归就是把一个复杂的问题使用同一方法分解为较简单的问题，如果这个的问题依然得不到解决就再次分解，直到问题能被直接处理为止。虽然递归结果清晰代码简洁，但在运行时也感觉到了，递归算法的运行效率较低，耗费的计算时间较长。

**3.4.3** 对于选做题中的多文件编译和链接最初不是很理解，后来再请教同学后

也算是有了初步的了解并完成了实验作业，。

# 实验4 编译预处理实验

## 4.1 实验目的

1. 掌握文件包含、宏定义、条件编译、assert宏的使用；
2. 练习带参数的宏定义、条件编译的使用；
3. 练习assert宏的使用；
4. 使用Turbo C 2.0集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 4.2 实验内容

### 4.2.1 源程序改错题

下面是用宏来计算平方差、交换两数的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

1 #include "stdio.h"

2 #define SUM a+b

3 #define DIF a-b

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

5 void main

6 {

7 int b, t;

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

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

10 printf("\nSUM=%d\n the difference between square of a and square of b

is:%d",SUM, SUM\*DIF);

11 SWAP(a,b);

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

13 }

**解答：**

（1）错误修改：

1) 第4行的宏定义中没有实现a,b的交换，正确形式为：

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

2) 第5行的函数声明中没有说明参数类型，正确形式为：

void main()

3) 第7行的变量声明中缺少a，正确形式为：

int a, b, t;

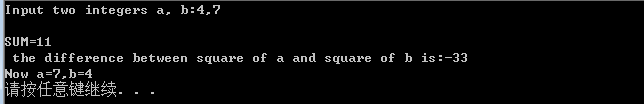
4) 第10行中使用宏定义没有考虑到括号对计算次序正确性的影响，正确形式为：

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

5) 第12行的输出语句中printf应为小写，正确形式为：

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

（2）错误修改后运行结果：



### 4.2.2 源程序修改替换题

下面是用函数实现求三个数中最大数、计算两数之和的程序，在这个源程序中存在若干语法和逻辑错误。

要求：1）对这个例子程序进行调试修改，使之能够正确完成指定任务；

2）用带参数的宏替换函数max，来实现求最大数的功能。

void main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

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

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

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

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

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

}

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

{

int t;

if (x>y)

t=x;

else

t=y;

if (t<z)

t=z;

return t;

}

float sum(float x, float y)

{

return x+y;

}

**解答：**

将头文件和函数声明补充完整，将求三者最大值改为宏定义，替换后的程序如下所示：

#include<stdio.h>

#include<stdlib.h>

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

float sum(float x, float y);

void main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

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

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

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

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

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

system("pause");

}

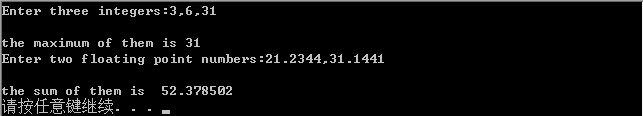
float sum(float x, float y)

{

return x + y;

}

修改后运行结果：



### 4.2.3 跟踪调试题

下面程序利用R计算圆的面积s，以及面积s的整数部分。

#define R

void main(void)

{

float r, s;

int s\_integer=0;

printf ("input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

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

s\_integer= integer\_fraction(s);

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

assert((s-s\_integer)<1.0);

#endif

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

1. 修改程序，使程序编译通过且能运行；

程序如下：

#include<stdio.h>

#include<assert.h>

#include<stdlib.h>

#define R

int integer\_fraction(float x);

void main(void)

{

float r, s;

int s\_integer = 0;

printf("input a number: ");

scanf("%f", &r);

#ifdef R

s = 3.14159\*r\*r;

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

s\_integer = integer\_fraction(s);

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

assert((s - s\_integer)<1.0);

#endif

system("pause");

}

int integer\_fraction(float x)

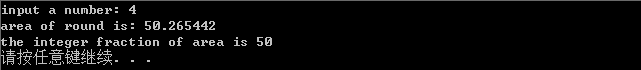
{

int i = x;

return i;

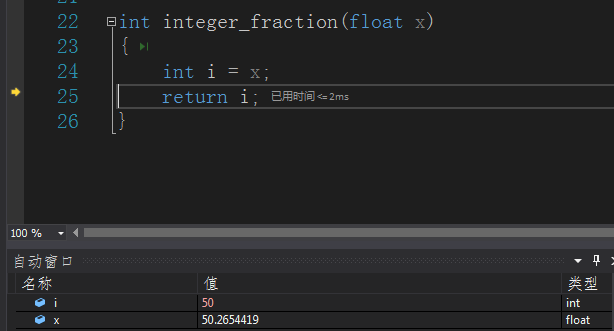
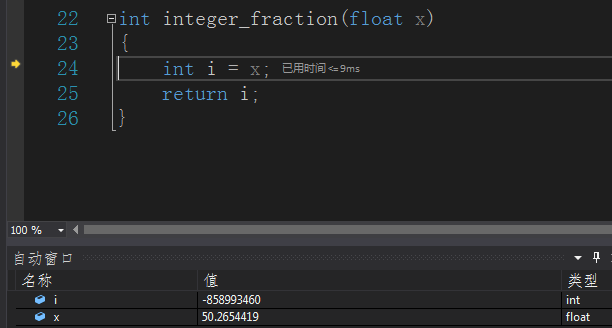
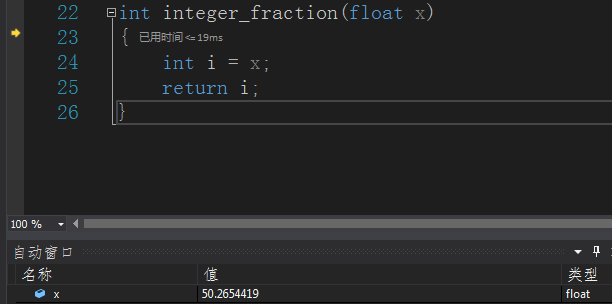
}

修改后运行结果：



1. 单步执行。进入函数decimal\_fraction时watch窗口中x为何值？在返回main时, watch窗口中i为何值？

X=50.2654419 i=50



1. 排除错误，使程序能正确输出面积s值的整数部分，不会输出错误信息assertion failed。

**解答：**

#define R

#include<stdio.h>

#include <assert.h>

int integer\_fraction(float x);

int main(void)

{

float r = 0.0, s = 0.0;

int s\_integer = 0;

printf("input a number: ");

scanf("%f", &r);

#ifdef R

s = 3.14159\*r\*r;

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

s\_integer = integer\_fraction(s);

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

assert((s - s\_integer)<1.0);

#endif

}

int integer\_fraction(float x)

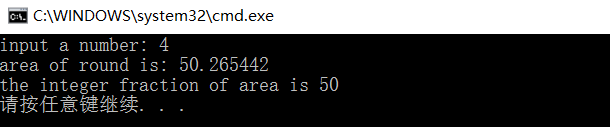
{

int i = (int)x;

return i;

}

运行结果如图所示：



### 4.2.4 编程设计题

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

**解答：**

1） 算法流程如图4.1所示。

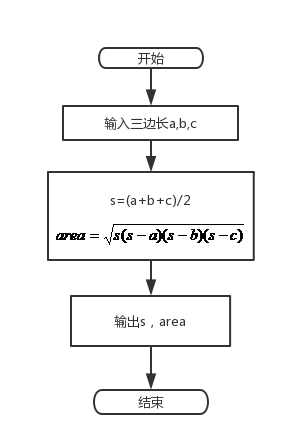


图4-1 编程题1的程序流程图

1. 源程序清单

#include<stdio.h>

#include<stdlib.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()

{

int a, b, c;

while (scanf("%d %d %d", &a, &b, &c) != EOF)

{

printf("%lf %lf\n", (double)s(a, b, c), area((double)s(a, b, c), a, b, c));

}

return 0;

system("pause");

}

3）测试

（a） 测试数据：

表4-1 编程题1的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 | 运 行 结 果 |
| a | b | c |
| 用例1 | 4 | 4 | 4 | 6.000000 6.928203 | 截图4-2 |
| 用例2 | 3 | 2 | 2 | 3.500000 1.984313 | 截图4-2 |
| 用例3 | 3 | 4 | 5 | 6.000000 6.000000 | 截图4-2 |

（b） 对应测试测试用例1用例2用例3的运行结果如图4-2所示。

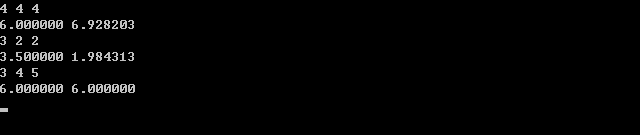


图4-2 编程题1的测试用例一用例二用例三的运行结果

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

**解答：**

1） 算法流程如图4.3所示。

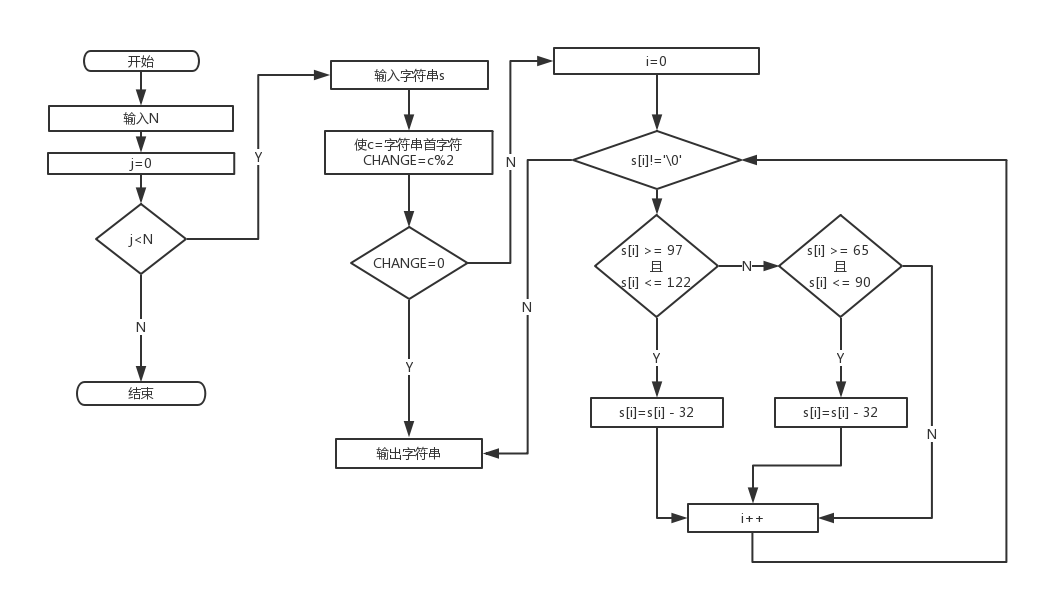


图4-3 编程题2的程序流程图

2)源程序清单

#include<stdio.h>

#include<stdlib.h>

#define CHANGE 1//#define CHANGE 0

int main()

{

int N;

scanf("%d", &N);

getchar();

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

{

char c;

int i = 0;

char s[100];

fgets(s, 100, stdin);

//while ((c = getchar()) != '\n')s[i++] = c; s[i] = '\0';

c = s[0];

#if CHANGE

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

{

if (s[i] >= 97 && s[i] <= 122)

printf("%c", s[i] - 32);

else if (s[i] >= 65 && s[i] <= 90)

printf("%c", s[i] + 32);

else printf("%c", s[i]);

}

#else

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

{

printf("%c", s[i]);

}

#endif

}

system("pause");

}

3）测试

（a） 测试数据：

表4-2 编程题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | 理 论 结 果 | 运 行 结 果 |
| N | s |
| 用例1 | 3 | HELLO,WORLD! | Hello,world! | 截图4-4 |
| 用例2 | abcd!@#$%^&\* | ABCD!@#$%^&\* | 截图4-4 |
| 用例3 | Uqwe,./ | uQWE,./ | 截图4-4 |

（b） 对应测试测试用例1用例2用例3的运行结果如图4-4所示。

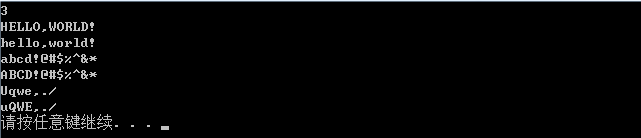


图4-4 编程题2的测试用例一用例二用例三的运行结果

## 4.3 自设题

**（1）**自设实验题目：多种方式输出字符指针。

**（2）**实验目的：掌握字符指针的输出方式。

**（3）**实验程序：

#include <stdio.h>

#include <string.h>

int main()

{

char str[] = "Hello,world!";

char \*pstr = str;

int len = strlen(str), i;

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

{

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

}printf("\n");

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

{

printf("%c", pstr[i]);

}printf("\n");

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

{

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

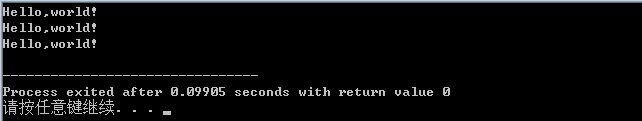
}printf("\n");

return 0;

}

**(4)**实验用例：Hello,world!

输出结果如下图:



**（5）**实验结论：设有声明 int x[10],\*pstr=x;则

第i号元素的地址可以写为：&x[i],x+I,pstr+I;

第i号元素的数值可以写为：x[i],\*(x+i),\*(pstr+i).

## 4.4 实验小结

**4.4.1** 通过本次实验，初步了解掌握了编译预处理的一些内容，但对可变参数宏定义和通用类型宏还不是很熟悉。

**4.4.2** 在完成实验要求的过程中，注意到了括号对编译宏定义的影响之大，以及++运算在宏定义中的运算若不运用函数可能会导致错误，总结来说宏定义比较适合经常使用的简短的表达式，或者是短小可重复的代码段，这些情况下使用宏定义可以简短程序，使程序运行更快。

# 实验5 数组实验

## 5.1 实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

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

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

## 5.2 实验内容

### 5.2.1 源程序改错

下面是用来将数组a中元素按升序排序后输出的源程序。分析源程序中存在的问题，并对源程序进行修改，使之能够正确完成任务。

源程序

1 #include<stdio.h>

2 int main(void)

3 {

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

5 void sort(int [],int);

6 int i;

7 sort(a[0],10);

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

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

10 printf("\n");

11 return 0;

12 }

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

14 {

15 int i, j, t;

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

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

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

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

20 }

**解答：**

（1）错误修改：

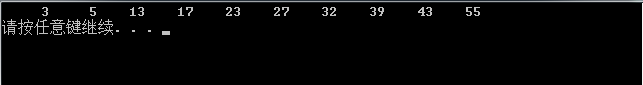
1) 第7行调用函数时参数类型错误，正确形式为：

sort(a, 10);

2）第18行逻辑错误，按源程序运行则会按降序输出，正确形式为：

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

（2）错误修改后运行结果：

****

### 5.2.2 源程序完善、修改、替换

(1) 下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。请在源程序中的下划线处填写合适的代码来完善该程序。

源程序：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M];

/\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

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

a[i] = i + 1;

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

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

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

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

b[M-i] = j? : ;

/\* 将报数为N的人的编号存入数组b \*/

if(j)

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

;

}

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

printf(“%6d”, b[i]);

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

return 0;

}

两处分别完善为：b[M - i] = j ? a[j - 1] : a[i-1];

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

完善后的运行结果如图所示：



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

解答：

1) 解题思路：

1.定义一个数组a[M]，并将其全部元素赋值为1，表示所有人都在圈内；

2.i=0,count=M,k=0;

3.循环i++;

3.1如果a[i]=0即表示这个人已经出圈，如果这个人在数组

的最后一个即i=M-1，则i=-1，以使调回数组头部，转3；

3.2 k++;

3.3 如果数到N即k=N,则令a[i]=0,表示这个人出圈，并使

k=0,count--;表示重头开始计数，且圈内总人数减1；

3.4 当所有人都出圈即count=0时，转4；

3.5 当到达数组末尾时，跳到数组头部，即i=-1；

4. 结束

2）程序清单

#include<stdio.h>

#include<stdlib.h>

#define M 10

#define N 3

int main()

{

int i, k=0, count=M, a[M];

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

a[i] = 1;//a数组的每个元素初值为1，表示所有人都在圈内

for (i = 0;; i++)

{

if (a[i] == 0)//说明这个人已经不再圈里

{

if (i == M - 1)

i = -1;

continue;

}

k++;

if (k == N)

{

a[i] = 0;

printf("%6d", i + 1);

k = 0;

count--;

}

if (count == 0)

break;

if (i == M - 1)//如果到大数组末尾，就跳到数组头部

i = -1;

}

printf("\n");

system("pause");

}

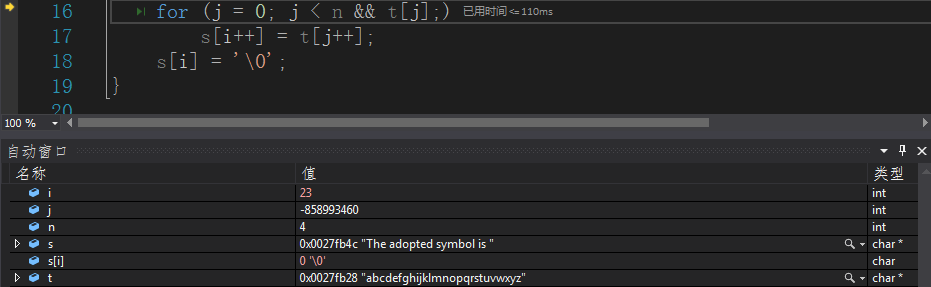
（b） 修改后的程序运行结果如图：

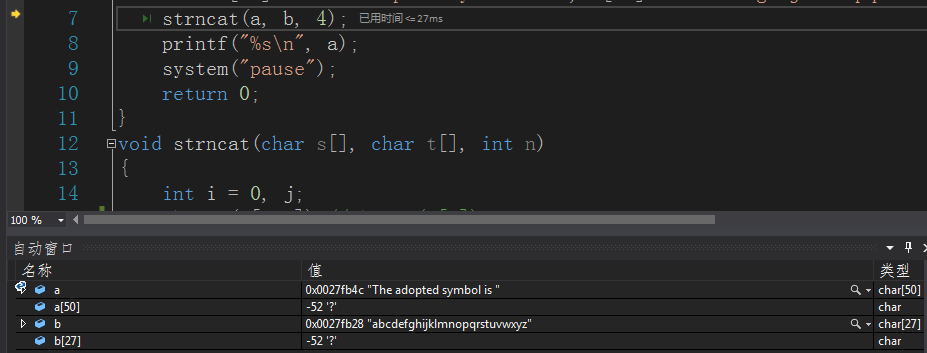


### 5.2.3 跟踪调试源程序

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

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





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

源程序：

#include<stdio.h>

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

int main(void)

{

char a[50]="The adopted symbol is ",

b[27]="abcdefghijklmnopqrstuvwxyz";

strncat(a, b, 4);

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

return 0;

}

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

{

int i = 0, j;

while(s[i++]) ;

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

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

s[i] = '\0';

}

标记圆点处改为while(s[i])i++;

修改后的运行结果如下图：



### 5.2.4 编程设计题

编写并上机调试运行能实现以下功能的程序。

（1）编写一个程序,从键盘读取数据，对一个3×4矩阵进行赋值，求其转置矩阵，然后输出原矩阵和转置矩阵。

**解答：**

1） 算法流程如图5.1所示。

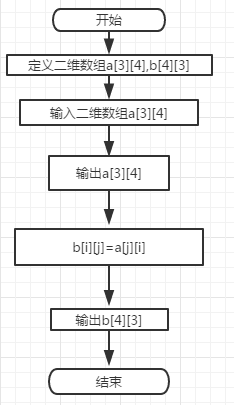


图5-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a[3][4],b[4][3];

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

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

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

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

{

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

printf("%5d", a[i][j]);

printf("\n");

}

printf("\n");

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

for (int j = 0; j < 3; j++)

b[i][j] = a[j][i];

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

{

for (int j = 0; j < 3; j++)

printf("%5d", b[i][j]);

printf("\n");

}

system("pause");

}

3）测试

（a） 测试数据：

表5-1 编程题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| a[3][4] |
| 用例 | 1 2 3 4  5 6 7 8  9 10 11 12 | 1 2 3 4  5 6 7 8  9 10 11 12  1 5 9  2 6 10  3 7 11  4 8 12 | 截图5-2 |

（b） 对应测试测试用例的运行结果如图5-2所示。

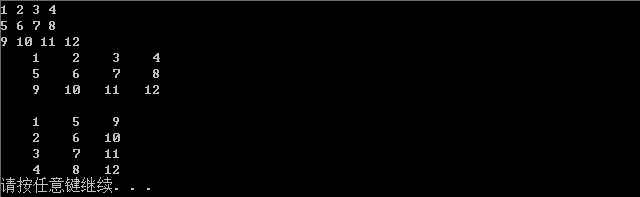


图5-2 编程题1的测试用例的运行结果

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

解答：

1） 算法流程如图5.3所示。

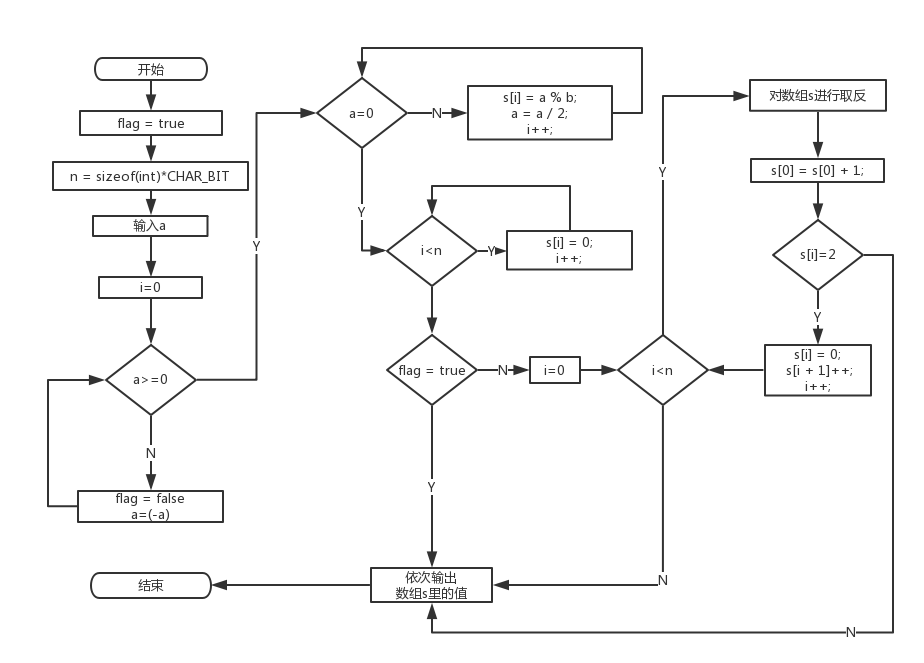


图5-3 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

#include<limits.h>

#include<stdlib.h>

#include<stdbool.h>

void fac(int);

int main()

{

int i=0;

int s[100];

while ((scanf("%d", &s[i])) != EOF)

{

fac(s[i]); i++;

}

system("pause");

}

void fac(int a)

{

int n = sizeof(int)\*CHAR\_BIT;

int i = 0;

int s[100], b = 2;

bool flag = true;

if (a >= 0)flag = true;

else

{

a = 0 - a; flag = false;

}

for (; a != 0; i++)

{

s[i] = a % b;

a = a / 2;

}

for (; i < n; i++)

s[i] = 0;

if (flag == true);

else

{

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

if (s[i] == 1)s[i] = 0;

else s[i] = 1;

s[0] = s[0] + 1;

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

if (s[i] == 2) { s[i] = 0; s[i + 1]++; }

else break;

}

for (i = n - 1; i >= 0; i--)

{

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

}

printf("\n");

}

3）测试

（a） 测试数据：

表5-2 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理论结果 | 运行结果 |
| N |
| 用例1 | 1 | 00000000000000000000000000000001 | 截图5-4 |
| 用例2 | 0 | 00000000000000000000000000000000 |
| 用例3 | -1 | 11111111111111111111111111111111 |

（b） 对应测试测试用例的运行结果如图5-4所示

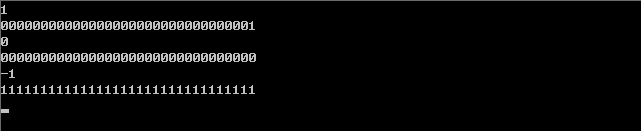


图5-4 编程题2的测试用例的运行结果

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

**解答：**

1) 解题思路：

1.输入n组姓名和成绩；

2.利用冒泡排序将成绩从高到低排序并输出。

3.输入要查找的成绩；

3.1若存在，输出对应的姓名和成绩；

3.2若不存在，输出提示“not found!”。

4. 结束

2）源程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void sort(int t[], char s[][100], int n);

int binarySearch(int a[], int n, int key);

int main()

{

char s[100][100];

int t[100];

int n, i = 0, j = 0; scanf("%d", &n);

getchar();

char c;

for (; i < n; i++)

{

j = 0;

while ((c = getchar()) != ' ')

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

s[i][j] = '\0';

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

getchar();

}

s[i][j] = '\0';

sort(t, s, n);

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

printf("%-20s %d\n", s[k], t[k]);

printf("\n");

int N, key[100]; scanf("%d", &N);

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

{

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

getchar();

}

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

{

if ((binarySearch(t, n, key[i])) == (-1))printf("Not found!\n");

else printf("%-20s %d\n", s[binarySearch(t, n, key[i])], key[i]);

}

system("pause");

return 0;

}

int binarySearch(int t[], int N, int key)

{

int low = 0;

int high = N - 1;

int mid;

int midVal;

while (low <= high)

{

mid = (low + high) / 2;

midVal = t[mid];

if (midVal<key)

high = mid - 1;

else if (midVal>key)

low = mid + 1;

else return mid;

}

return -1;

}

void sort(int t[], char s[][100], int n)

{

int i, j, k, temp1; char temp2[100];

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

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

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

{

temp1 = t[j];

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

t[j + 1] = temp1;

strcpy(temp2, s[j]);

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

strcpy(s[j + 1], temp2);

}

}

3）测试

（a） 测试数据：

表5-3 编程题3的测试数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程序输入 | | | | 理想结果 | 运行结果 |
| 姓名 | 成绩 | 查找  次数 | 查找成绩 |
| 用例 | AA | 45 | 3 | 90 | CC 90  EE 79  BB 78  AA 45  DD 36  CC 90  EE 79  Not found! | 截图5-5 |
| BB | 78 |
| CC | 90 | 79 |
| DD | 36 | 60 |
| EE | 79 |

（b）对应测试测试用例的运行结果如图5-5所示

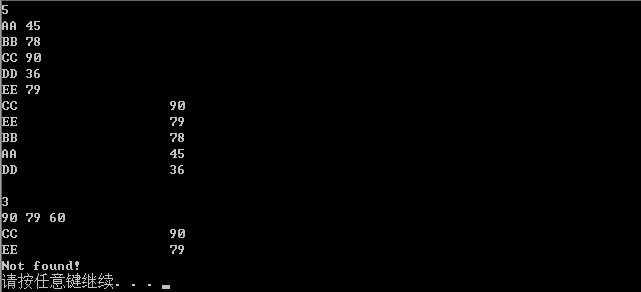


图5-5 编程题3的测试用例的运行结果

### 5.2.5 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：

1. 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中字符穿的的第n个字符的后面。

**解答：**

1. 解题思路：
2. 输入字符串s1,s2和数字n；
3. 调用函数strnins(s,t,n),

2.1用循环得到s中元素的总个数i；

2.2如果i<n，输出“n is too big!!!”,转3；

2.3如果i>=n,从第n个元素开始将t中元素依次赋值给s，输出s;

转到3；

1. 结束。
2. 源程序清单

#include<stdio.h>

#include<stdlib.h>

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

int main()

{

int n,i=0;

char c,s1[100], s2[100];

printf("输入字符串s1:");

while ((c=getchar()) != '\n')s1[i++]=c;

s1[i] = '\0';

i = 0;

printf("输入字符串s2:");

while ((c = getchar()) != '\n')s2[i++]=c;

s2[i] = '\0';

printf("输入n:");

scanf("%d", &n);

strnins(s1, s2, n);

system("pause");

}

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

{

int i = 0, j = 0, k;

while (s[i] != '\0')i++;

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

if (i<n)

printf("n is too big!!!\n");

else

{

for (i = n - 1, k = 0; i < j + n - 1 ; i++, k++)s[i] = t[k];

s[i] = '\0';

printf("结果为：");

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

}

}

3）测试

（a） 测试数据：

表5-4 选做题1的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 | 运 行 结 果 |
| s1 | s2 | n |
| 用例1 | asdfghjkl | zxcvbnm | 3 | aszxcvbnm | 截图5-6 |
| 用例2 | asd | qwe | 5 | n is too big!!! | 截图5-7 |

（b） 对应测试测试用例1的运行结果如图5-6所示。



图5-6 选做题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5-7所示。

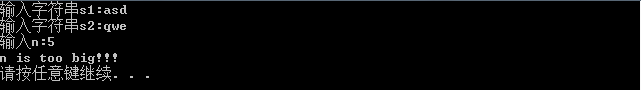


图5-7 选做题1的测试用例二的运行结果

（2）编写一个实现八皇后问题的程序，即：在8\*8放个国际象棋盘上放置八个皇后，任意两个皇后不能位于同一行、同一列或者同一斜线（正斜线或反斜线）上，并输出所有可能的方法。

**解答：**

1. 解题思路:
2. 定义一个数组c,存放的是皇后放在每一行的第几个位置。
3. 将第一个皇后放在一行一列；
4. 将第二个皇后放在二行一列，判断是否符合题意，不符合则放于二行二列：一次类推；如果自一到八都无法满足，则返回上一级；
5. 输出每一种可能性的结果。
6. 结束。
7. 源程序清单

#include <stdio.h>

#include<stdlib.h>

int c[20], n = 8, cnt = 0;

void print()

{

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

{

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

{

if (j == c[i]) printf( "1 ");

else printf( "0 ");

}

printf("\n");

}

printf("\n");

}

void search(int r)

{

if (r == n)

{

print();

cnt++;

return;

}

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

{

c[r] = i;

int ok = 1;

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

if (c[r] == c[j] || r - j == c[r] - c[j] || r - j == c[j] - c[r])

{

ok = 0;

break;

}

if (ok) search(r + 1);

}

}

int main()

{

search(0);

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

system("pause");

return 0;

}

3）测试

（a） 测试数据：n=8

（b） 对应测试测试用例的运行结果如图5-8所示。

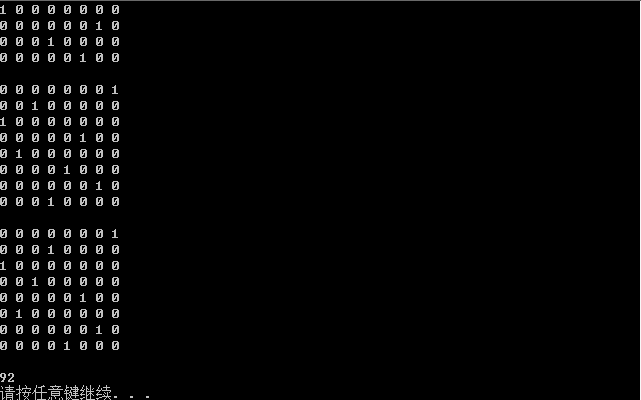


图5-8 选做题2的测试用例的运行结果

## 5.3 自设题

（1）自设实验题目：读入一行次，然后按词把整行倒置输出。例如：输入How are you，则输出you are How。使用递归。

（2）实验目的： 熟悉使用递归，对递归有一定的理解。

（3）实验程序：

#include <stdio.h>

#include<stdlib.h>

void reverse();

int main()

{

printf("input:\n");

reverse();

system("pause");

return 0;

}

void reverse()

{

int i, j;

char c, ch[10];

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

{

c = getchar();

if (c == ' ' || c == '\t' || c == '\n')

break;

ch[i] = c;

}

if (c != '\n')

{

reverse();

}

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

{

putchar(ch[j]);

}

printf(" ");

}

（4）实验用例：



（5）实验结论：之前对递归的理解在某些地方存在着一定的错误，自己的理解可以说是扁平化的，递归很有利于简化程序。

## 5.4 实验小结

**5.4.1** 在解决约瑟夫问题时遇到了很多问题，感觉自己的思维比较局限，在思考问题是难以有突破。这个问题在后面的八皇后问题中也体现出来了，也看了一些回溯法的讲解，最后还是没能独立完成八皇后问题，而是去寻求同学的帮助。一方面通过约瑟夫环和八皇后问题的解决，又一次感受到编程的魅力，同时也感受到了算法学习的重要性。

**5.4.2**体会到了交流的重要性，自己在闷头写程序的时候，思维很容易陷入一些奇怪的死循环中，或者就是程序冗长而繁琐，所以也应在自己独立思考的同时，多于他人交流。

# 实验6 指针实验

## 6.1 实验目的

1．熟练掌握指针的说明、赋值、使用。

2．掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

3．熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

4．掌握指针函数与函数指针的用法。

5．掌握带有参数的main函数的用法。

## 6.2 实验内容

### 6.2.1 源程序改错题

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

1 #include "stdio.h"

2 void main(void)

3 {

4 float \*p;

5 scanf("%f",p);

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

7 }

**解答：**

（1）错误修改：

1) 第2行的主函数的返回值类型应该为int型，正确形式为：

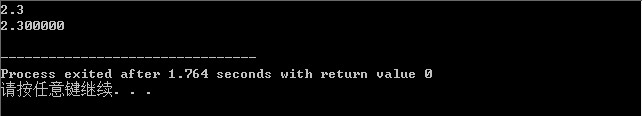
int main(void)

2) 第4行的没有为指针分配空间，可使用malloc函数为指针分配空间，同时加上头文件#include<stdlib.h>，正确形式为：

float \*p;

p=(float \*)malloc(sizeof(float));

（2）错误修改后运行结果：

****

### 6.2.2 源程序完善、修改、替换题

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

#include "stdio.h"

#include "string.h"

void main(void)

{

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

int choice, i;

do {

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

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

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

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

scanf("%d", &choice);

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

switch (choice) {

case 1:

p = strcpy;

break;

case 2:

p = strcat;

break;

case 3:

goto down;

}

getchar();

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

i = 0;

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

i = 0;

result = (a, b);

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

down:

;

}

**解答：**

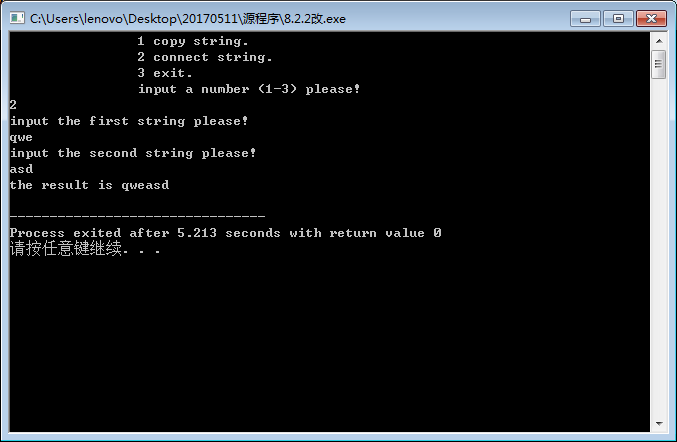
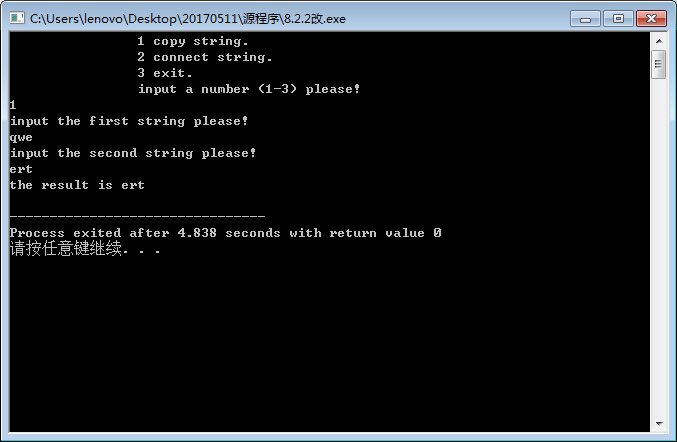
三处分别补充为：

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

gets(a);

gets(b);

运行的结果截图如下：



（2）为了使程序不受scanf、getchar、gets等函数输入后回车符的影响，请修改第（1）题程序，按要求输出下面结果：（（输入）表示该数据是键盘输入数据）

1 copy string.

2 connect string.

3 exit.

input a number (1-3) please!

2 （输入）

input the first string please!

the more you learn, （输入）

input the second string please!

the more you get. （输入）

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

**解答：**

考虑到fgets会读入换行符，则改为通过循环输入字符串，或者使用gets语句，替换后的程序如下所示：

#include "stdio.h"

#include "string.h"

int main(void)//void main(void)

{

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

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

int choice,i;

do{

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

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

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

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

scanf("%d",&choice);

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

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

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

i=0;

for(;i<80;i++)

{

a[i]=getchar();

if(a[i]=='\n')break;

}

a[i]='\0';

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

i=0;

for(;i<80;i++)

{

b[i]=getchar();

if(b[i]=='\n')break;

}

b[i]='\0';

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

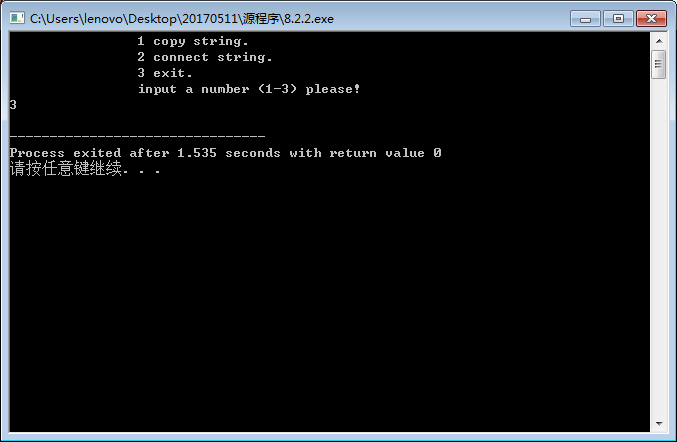
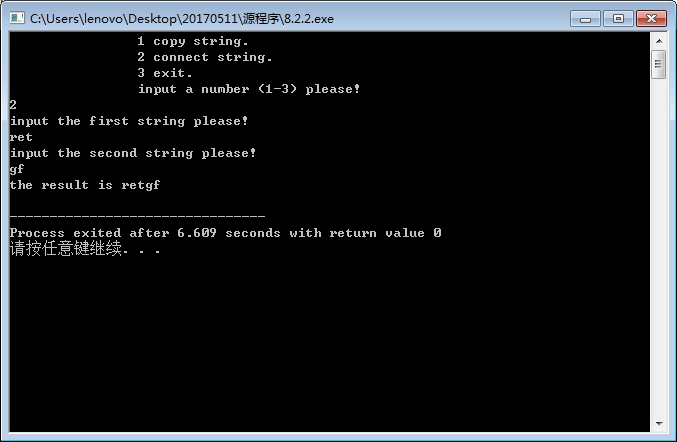
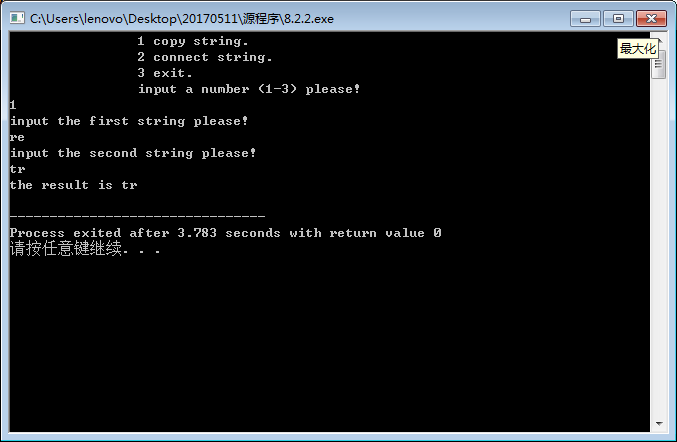
printf("the result is %s\n",result);

down:

;

}

运行结果的截图如下：



### 6.2.3 跟踪调试题

#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？

88

（2）排除错误，使程序输出结果为：

there is a boat on the lake.

**解答：**

#include "stdio.h"

#include<stdlib.h>

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

void main(void)

{

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

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

}

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

{

char \*p=s;

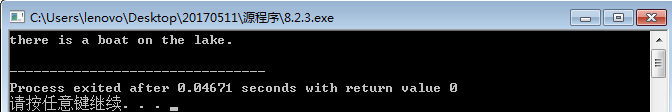
while(\*p++=\*t++)

;

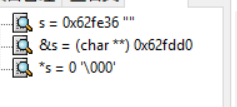
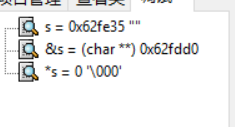
return (s);

}

运行结果的截图为：



（3）选做：由于watch窗口中只显示s所指串的值，不显示s中存储的地址值，怎样才能观察到s值的变化呢？



### 6.2.4 编程设计题

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

1） 算法流程如图6-1所示。

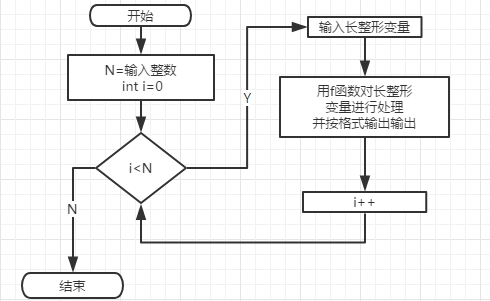


图6-1 编程设计题1的程序流程图

2）源程序清单

#include<stdio.h>

void f(long x);

int main()

{

long x[100];

int N;scanf("%d",&N);

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

{

scanf("%ld",&x[i]);

}

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

{

f(x[i]);

printf("\n");

}

}

void f(long x)

{

char mask=0x0f;

char \*p=(char\*)&x+3;

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

{

printf("%X ",((\*p)>>4)&mask);

printf("%X",(\*p)&mask);

if(i!=3)printf(" ");

p--;

}

}

3）测试

（a） 测试数据：

表6-1 编程设计题1的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | 理 论 结 果 | 运 行 结 果 |
| N | x |
| 用例 | 3 | 90 | 0 0 0 0 0 0 5 A | 截图6-2 |
| -23 | F F F F F F E 9 |
| 0 | 0 0 0 0 0 0 0 0 |

（b） 对应测试测试用例的运行结果如图6-2所示。

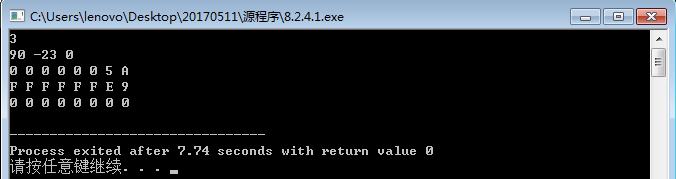


图6-2 程序设计题1的测试用例的运行结果

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

1） 算法流程如图6-3所示。

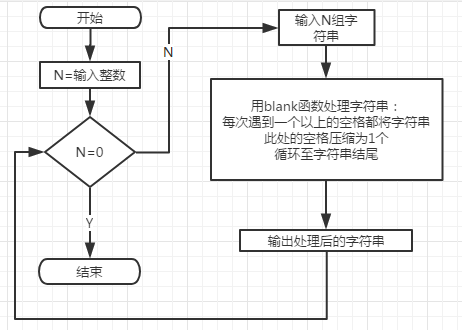


图6-3 编程设计题2的程序流程图

2）源程序清单

#include<stdio.h>

#include<string.h>

#define M 80

void blank(char\*p[100],int N);

int main()

{

while(1)

{

int N;

scanf("%d",&N);

if(N==0)break;

getchar();

int i;

char b[100][M];

char \*p[100];

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

{

fgets(b[i],80,stdin);

p[i]=b[i];

}

blank(p,N);

}

}

void blank(char \*p[100],int n)

{

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

{

int j=0;

for (j = 0; p[i][j]!='\0'; j++)

{

if(p[i][j]==' '&&p[i][j+1]==' ')

{

int k=j+1,temp=j+1;

while(p[i][k++]==' ');

k--;j++;

for(;p[i][k]!='\0';k++,j++)

p[i][j]=p[i][k];

p[i][j]='\0';

j=temp;

}

}

if(p[i][0]!='\n')printf("%s", p[i]);

}

printf("\n");

}

3）测试

（a） 测试数据：

表6-2 编程设计题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | 理论结果 | 运行结果 |
| N | b |
| 用例1 | 2 | asd fgh jkl | asd fgh jkl | 截图6-4 |
| qwe rty | qwe rty |
| 用例2 | 3 | xi da da | xi da da |
| (空行) | (没有输出) |
| we’d stop trying. | we’d stop trying. |
| 用例3 | 0 |  | 结束 |

（b） 对应测试测试用例的运行结果如图6-4所示。

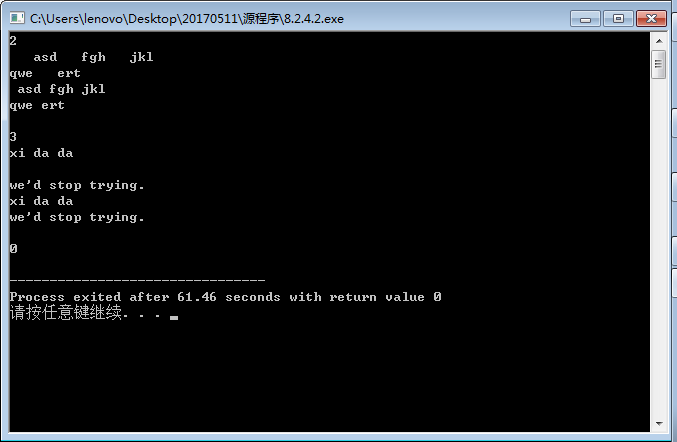


图6-4 程序设计题2的测试用例的运行结果

（3）编写一个程序，输入n个整数，排序后输出，排序的原则由命令行可选参数-d决定，并且有参数-d时按递减顺序排序，否则按递增顺序排序。要求将排序算法定义成函数，利用指向函数的指针使该函数实现递增或递减排序。

1） 算法流程如图6-5所示。

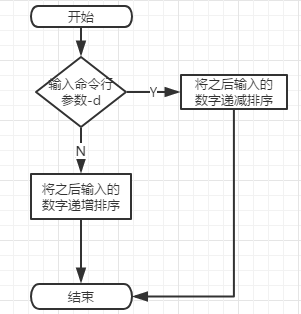


图6-5 编程设计题3的程序流程图

2）源程序清单

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define n 4

void s\_sort(int a[], int x, int y);

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

{

int flag;

if (argc > 2) printf("error\n");

else

{

if (argc == 2)

{

if(strcmp(argv[1], "-d"))

printf("error\n");

else flag = 1;

}

else flag = 0;

}

int i;

int a[n];

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

{

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

}

void(\*p)(int a[], int x, int y);

p = s\_sort;

p(a, n, flag);

return 0;

}

void s\_sort(int a[], int x, int y)

{

int i, j, t;

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

{

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

{

if ((y) ? (a[j] > a[i]) : (a[j] < a[i]))

{

t = a[j];

a[j] = a[i];

a[i] = t;

}

}

}

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

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

printf("\n");

}

3）测试

（a） 测试数据：

表6-3 编程设计题3的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | 理论结果 | 运行结果 |
| 命令行参数 | 数字 |
| 用例1 | -d | 34 45 2 5 | 45 34 5 2 | 截图6-6 |
| 用例2 | （无） | 34 45 2 5 | 2 5 34 45 |

（b） 对应测试测试用例1用例2的运行结果如图6-6所示。



图6-6 程序设计题3的测试用例1用例2的运行结果

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

a.计算每个学生各门课程平均成绩；

b.计算全班每门课程的平均成绩；

c.分别统计低于全班各门课程平均成绩的人数；

d.分别统计全班各门课程不及格的人数和90分以上（含90分）的人数。

在调用函数中输出上面各函数的计算结果。（要求都用指针操作，不得使用下标操作。）

1. 算法流程如图6-7所示。

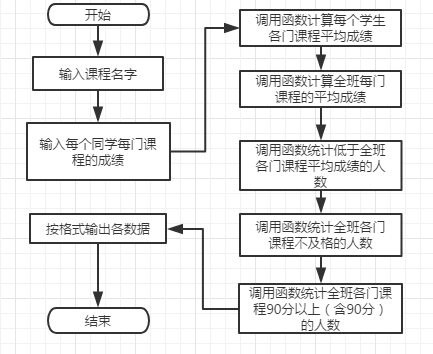


图6-7 编程设计题4的程序流程图

2）源程序清单

#include<stdio.h>

#include<stdlib.h>

#define N 5

#define M 5

float avg1[N], sum1[N];//每个同学的平均分

float avg2[M], sum2[M];//每门课程的平均分

char name[N][10];

char course[M][10];

int score[N][M];

void avg\_stu(char name[N][10],int score[N][M]);

void avg\_course(char course[M][10],int score[N][M]);

void num\_low\_avg(char name[N][10], char course[M][10], int score[N][M]);

void num(char name[N][10], char course[M][10], int score[N][M]);

int main()

{

int i = 0, j = 0;

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

scanf("%s ", \*(course + i));

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

{

scanf("%s ", \*(name + i));

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

scanf("%d", \*(score + i) + j);

}

avg\_stu(name, score);

avg\_course(course, score);

num\_low\_avg(name, course, score);

num(name, course, score);

system("pause");

}

void avg\_stu(char name[N][10], int score[N][M])

{

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

{

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

{

sum1[i] = sum1[i]+ \*(\*(score + i) + j);

}

avg1[i] = sum1[i] / M;

}

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

{

printf("Average score of %s is %.2f\n",\*(name + i), avg1[i]);

}

}

void avg\_course(char course[M][10], int score[N][M])

{

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

{

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

{

sum2[i] = sum2[i] + \*(\*(score + j) + i);

}

avg2[i] = sum2[i] / M;

}

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

{

printf("Average score of %s is %.2f\n", \*(course + i), avg2[i]);

}

}

void num\_low\_avg(char name[N][10], char course[M][10], int score[N][M])

{

int count[M] = {0};

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

{

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

{

if (score[j][i] < avg2[i])count[i]++;

}

}

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

printf("Number of students lower than avg of %s is %d\n", \*(course + i), count[i]);

}

void num(char name[N][10], char course[M][10], int score[N][M])

{

int count1[M] = { 0 }, count2[M] = { 0 };

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

{

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

{

if (score[j][i] < 60)count1[i]++;

if (score[j][i] > 90)count2[i]++;

}

}

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

printf("Number of students %s fail is %d\n", \*(course + i), count1[i]);

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

printf("Number of students %s perfect is %d\n", \*(course + i), count2[i]);

}

3）测试

（a） 测试数据：

表6-4 编程设计题4的测试数据

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | | | | | 理论结果 | 运行结果 |
| 用例1 | 姓名  课程 | Zhang | Li | Wang | Han | Gan | 依次输出每位同学的平均成绩、每门课程的平均成绩、低于各门课程平均成绩的人数、各门课程不及格的人数、各门课程90分以上（含90分）的人数。 | 截图6-8 |
| A | 87 | 99 | 77 | 87 | 95 |
| B | 45 | 98 | 100 | 48 | 75 |
| C | 85 | 68 | 95 | 47 | 59 |
| D | 50 | 89 | 100 | 85 | 20 |
| E | 87 | 68 | 87 | 89 | 100 |

（b） 对应测试测试用例的运行结果如图6-8所示。

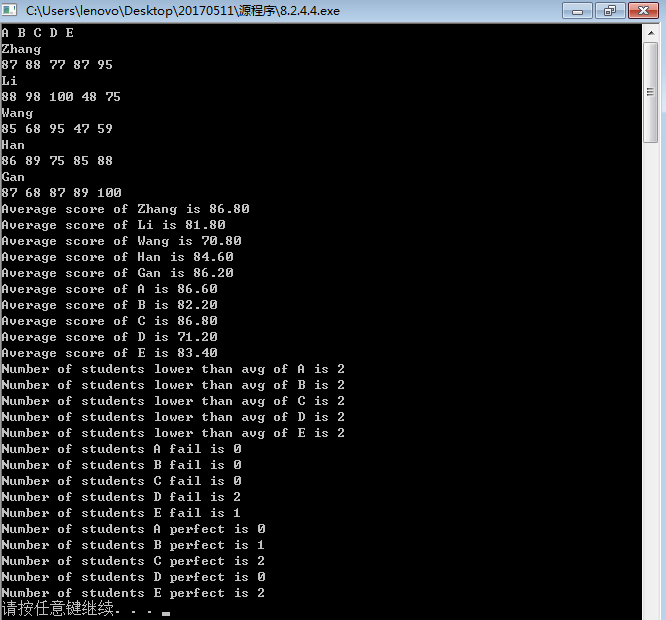


图6-8 程序设计题4的测试用例的运行结果

### 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 + 98765432109876543210.0123456789

**解答：**

1) 解题思路：讲数字一位一位存入数组进行运算。

2）程序清单

#include<stdio.h>

#include<stdlib.h>

#define M 10

#define N 20

int getnumber()

{

char c; int m;

L: c = getchar();

if (c == '.') return -1;

switch (c)

{

case '0':m = 0; break;

case '1':m = 1; break;

case '2':m = 2; break;

case '3':m = 3; break;

case '4':m = 4; break;

case '5':m = 5; break;

case '6':m = 6; break;

case '7':m = 7; break;

case '8':m = 8; break;

case '9':m = 9; break;

default: goto L;

}

return m;

}

int main()

{

int a[N + M + 1], b[N + M + 1], i;

bool flag = 0;

printf("请输入第一个数：");

for (i = M + N; i >= 0; i--)

{

a[i] = getnumber();

}

printf("请输入第二个数：");

for (i = M + N; i >= 0; i--)

{

b[i] = getnumber();

}

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

{

if (a[i] == -1) continue;

a[i] += b[i];

if (flag)

a[i]++;

flag = 0;

if (a[i] >= 10)

{

a[i] -= 10;

flag++;

}

}

printf("运算结果为：");

for (i = M + N; i >= 0; i--)

{

if (a[i] == -1) printf(".");

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

}

}

3）测试

（a） 测试数据：12345678912345678912.1234567891

98765432109876543210.0123456789

（b） 对应测试测试数据的运行结果如图6-9所示。

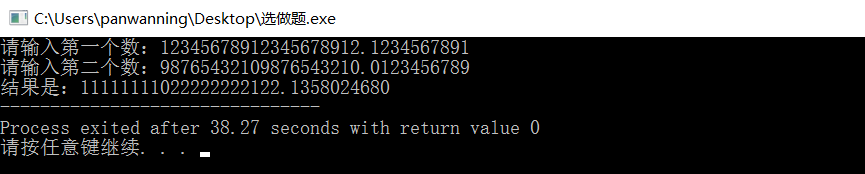


图6-9 选做题1的测试数据的运行结果

（2）编写使用复杂声明char \*(\*p[2])(const char \*,const char \*);的程序。

提示：p中元素可为strcmp、strstr等函数名。

## 6.3 自设题

**（1）** 自设实验题目： 设x,y为三维向量，z为3\*3矩阵。编写一个函数，利用指针变量作为一维数组参数，计算两个向量x，y的矢量积（向量的叉乘）并存放到z中。在调用函数中显示计算结果。要求都用指针操作，不得使用下标操作。

**（2）** 实验目的： 准确使用指针。

**（3）**实验程序：

#include<stdio.h>

void fun(int \*, int \*, int \*, int \*);

int main()

{

int x[3], y[3], z[3][3], r[3], \* pz;

int i,j;

pz = \*z;

printf("input x:");

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

scanf("%d", x+i);

printf("input y:");

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

scanf("%d", y+i);

fun(x, y, pz, r);

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

printf("%3c ", \*(pz + j));

printf("\n");

for(i=1; i<3; i++)

{

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

printf("%3d ", \*(pz + 3\*i + j));

printf("\n");

}

printf("Result is: (%d)i + (%d)j + (%d)k\n", \*r, \*(r+1), \*(r+2));

return 0;

}

void fun(int \*x, int \*y, int \*pz, int \*r)

{

int i, j;

j = 'i';

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

\*(pz + i) = j++;

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

\*(pz + 3 + i) = \*(x + i);

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

\*(pz + 6 + i) = \*(y + i);

\*(r + 0) = (\*(x + 1)) \* (\*(y + 2)) - (\*(x + 2)) \* (\*(y + 1));

\*(r + 1) = (\*(x + 2)) \* (\*(y)) - (\*(x)) \* (\*(y + 2));

\*(r + 2) = (\*(x)) \* (\*(y + 1)) - (\*(x + 1)) \* (\*(y));

}

**（4）**实验用例：

表6-5 自设题的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | 理论结果 | 运行结果 |
| x | Y |
| 用例1 | 1 2 3 | 4 5 6 | -3i+6j-3k | 截图6-9 |
| 用例2 | 1 2 3 | 2 4 6 | 0i+0j+0k | 截图6-10 |

对应测试测试用例1的运行结果如图6-9所示。

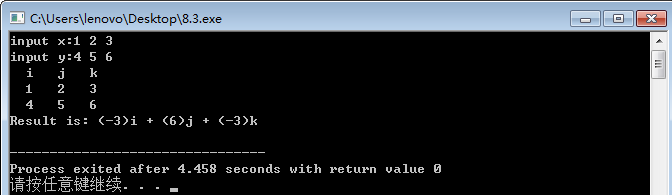


图6-9 自设题的测试用例1的运行结果

对应测试测试用例2的运行结果如图6-10所示。

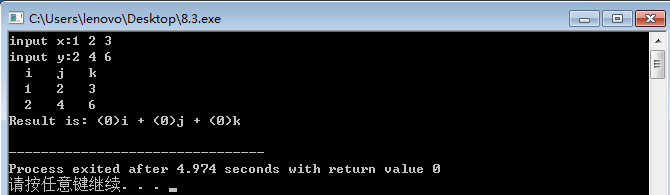


图6-10 自设题的测试用例2的运行结果

**（5）**实验结论：指针的使用须注意\*的两义，可以为乘，也可以为间访符，应注意使用括号。

## 6.4 实验小结

**6.4.1**本次实验一方面感受到了指针的便捷，另一方面也感受到了正确使用指

针需要注意太多的细节，在写程序时，不管是否使用了指针，都应该更加认真细

致。

**6.4.2** 如果是使用指针错误使程序无法运行，编译器是不会报错的，这次实

验也让我发现了自己对编译器报错功能的依赖，总是在报错之后才发现一些比较

愚蠢的错误，以后应该在写程序时更严密。

# 实验7 结构与联合实验

## 7.1 实验目的

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

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

3．了解字段结构和联合的用法。

## 7.2 实验内容

### 7.2.1 表达式求值的程序验证题

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

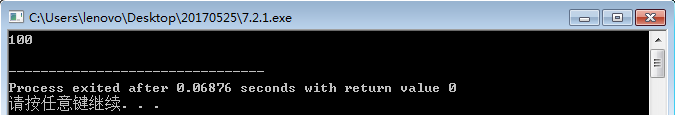
}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

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 | 100 | 100 |
| 2 | p++,p->c | B | B |
| 3 | \*p++->t,\*p->t | U x | U x |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

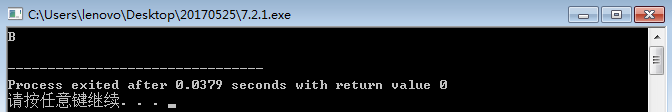
1）(++p)->x：

p先增1指向a[1]，然后访问p所指元素a[1]的x成员，结果为100。



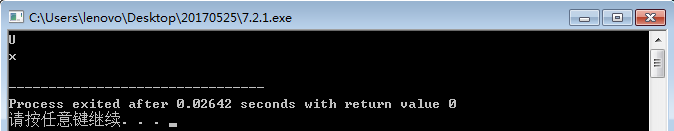
2）p++,p->c：

p先增1指向a[1]，然后访问p所指元素a[1]的c成员，结果为B。



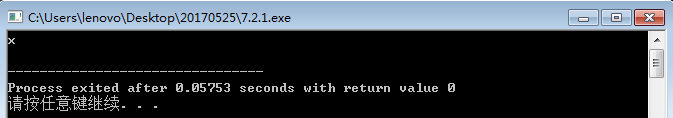
3）\*p++->t,\*p->t：

先访问p所指元素a[0]的t成员，结果为U，然后p自增1，指向a[1]，接着执行后一个语句将访问元素a[1]的t成员，结果为x。



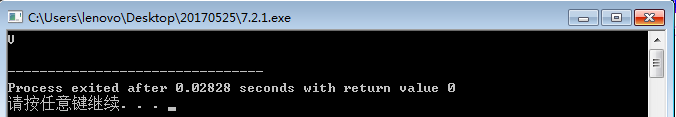
4）\*(++p)->t：

P先增1，然后间访p所指元素a[1]的t成员，结果为x。



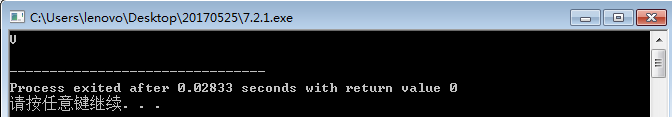
5）\*++p->t：

p所指元素a[0]的t成员，然后t自增,间访t[1],结果为V。



6）++\*p->t：

间访p指向的元素a[0]的t成员，即t[0]，然后t[0]自增，结果为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：

1 #include "stdio.h"

2 #include "stdlib.h"

3 struct s\_list{

4 int data; /\* 数据域 \*/

5 struct s\_list \*next; /\* 指针域 \*/

6 } ;

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

8 void main(void)

9 {

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

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

12 create\_list(head,s); /\* 创建新链表 \*/

13 p=head; /\*遍历指针p指向链头 \*/

14 while(p){

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

16 p=p->next; /\*遍历指针p指向下一结点 \*/

17 }

18 printf("\n");

19 }

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

21 {

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

23 if(p[0]==0) /\* 相当于\*p==0 \*/

24 ;

25 else { /\* loc\_head指向动态分配的第一个结点 \*/

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

27 loc\_head->data=\*p++; /\* 对数据域赋值 \*/

28 tail=loc\_head; /\* tail指向第一个结点 \*/

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

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

31 tail=tail->next; /\* tail指向新创建的结点 \*/

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

33 }

34 tail->next=NULL; /\* 对指针域赋NULL值 \*/

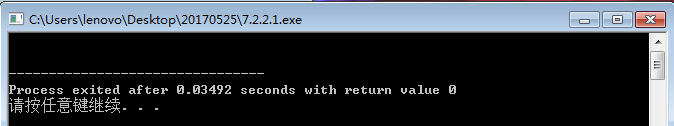
35 }

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

37 }

**解答：**

（1）源程序运行结果如图：



分析原因：对头指针进行的是值传递，对实参没有实际影响。

（2）错误修改：

1) 第7行的形参类型错误，正确形式为：

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

2) 第12行的参数类型错误，正确形式为：

create\_list(&head,s);

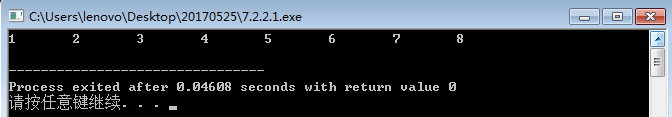
3) 第20行的参数类型错误，正确形式为：

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

4) 第36行应用指向头指针的指针进行操作，正确形式为：

\*headp=loc\_head;

（3）错误修改后运行结果：



（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：**

通过修改create\_list函数，将其建成一个后进先出的链表，替换后的程序如下所示：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

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

int main(void)//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);

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;

struct s\_list \* temp;

if(p[0]==0)

;

else {

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

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

tail=loc\_head;

while(\*p){

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

temp->next=loc\_head;

loc\_head=temp;

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

}

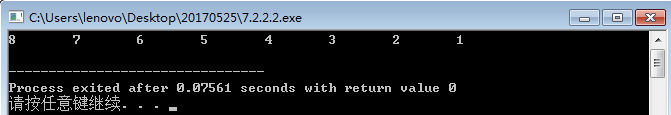
tail->next=NULL;

}

\*headp=loc\_head;

}

修改后运行结果：



### 7.2.3 编程设计题

（1）设计一个字段结构struct bits，它将一个8位无符号字节从最低位向最高位声明为8个字段，各字段依次为bit0, bit1, …, bit7，且bit0的优先级最高。同时设计8个函数，第i个函数以biti(i=0,1,2,…,7)为参数，并且在函数体内输出biti的值。将8个函数的名字存入一个函数指针数组p\_fun。如果bit0为1，调用p\_fun[0]指向的函数。如果struct bits中有多位为1，则根据优先级从高到低依次调用函数指针数组p\_fun中相应元素指向的函数。8个函数中的第0个函数可以设计为：

void f0(struct bits b)

{

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

}

**解答：**

1）算法流程如图7.1所示。

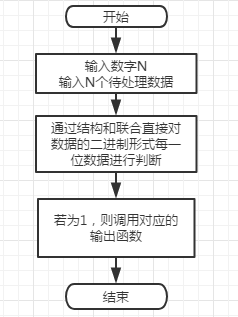


图7-1 编程设计题1的程序流程图

2）源程序清单

#include<stdio.h>

void isr0(void);

void isr1(void);

void isr2(void);

void isr3(void);

void isr4(void);

void isr5(void);

void isr6(void);

void isr7(void);

struct ISR\_BITS{

unsigned int bit0:1;

unsigned int bit1:1;

unsigned int bit2:1;

unsigned int bit3:1;

unsigned int bit4:1;

unsigned int bit5:1;

unsigned int bit6:1;

unsigned int bit7:1;

};

union ISR\_RSG{

unsigned short a;

struct ISR\_BITS b;

};

int main()

{

int N;scanf("%d",&N);

union ISR\_RSG x[10];

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

{scanf("%hu",&x[i].a);}

void (\*p\_fun[8])(void)={isr0,isr1,isr2,isr3,isr4,isr5,isr6,isr7};

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

{

printf("%d:\n",x[i].a);

if(x[i].b.bit0==1) (\*p\_fun[0])();

if(x[i].b.bit1==1) (\*p\_fun[1])();

if(x[i].b.bit2==1) (\*p\_fun[2])();

if(x[i].b.bit3==1) (\*p\_fun[3])();

if(x[i].b.bit4==1) (\*p\_fun[4])();

if(x[i].b.bit5==1) (\*p\_fun[5])();

if(x[i].b.bit6==1) (\*p\_fun[6])();

if(x[i].b.bit7==1) (\*p\_fun[7])();

printf("\n");

}

}

void isr0(void)

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

void isr1(void)

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

void isr2(void)

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

void isr3(void)

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

void isr4(void)

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

void isr5(void)

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

void isr6(void)

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

void isr7(void)

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

3）测试

（a） 测试数据：

选择用来测试的数据如表7-1所示。

表7-1 编程设计题1的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | 理 论 结 果 | 运 行 结 果 |
| X | m |
| 用例1 | 5 | 56 | The Interrupt Service Routine isr3 is called!  The Interrupt Service Routine isr4 is called!  The Interrupt Service Routine isr5 is called! | 截图7-2 |
| 0 | （无输出） |
| 255 | The Interrupt Service Routine isr0 is called!  The Interrupt Service Routine isr1 is called!  The Interrupt Service Routine isr2 is called!  The Interrupt Service Routine isr3 is called!  The Interrupt Service Routine isr4 is called!  The Interrupt Service Routine isr5 is called!  The Interrupt Service Routine isr6 is called!  The Interrupt Service Routine isr7 is called! |
| 8 | The Interrupt Service Routine isr3 is called! |
| 89 | The Interrupt Service Routine isr0 is called!  The Interrupt Service Routine isr3 is called!  The Interrupt Service Routine isr4 is called!  The Interrupt Service Routine isr6 is called! |

（b） 对应测试测试用例1的运行结果如图7-2所示。

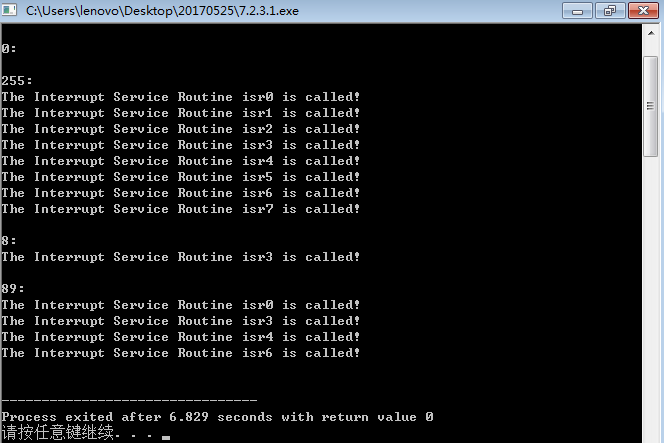


图7-2 编程设计题1的测试用例一的运行结果

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

(1) 输入每个学生的各项信息。

(2) 输出每个学生的各项信息。

(3) 修改指定学生的指定数据项的内容。

(4) 统计每个同学的平均成绩（保留2位小数）。

(5) 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

**解答：**

1）算法流程如图7.3所示。

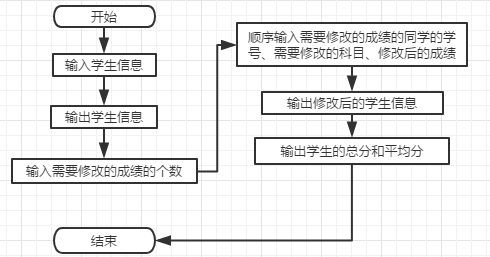


图7-3 编程设计题2的程序流程图

2）源程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/\*定义存入学生信息的结构\*/

struct stu{

char id[15];

char name[20];

float english;

float math;

float phy;

float c;

float sum;

float avg;

struct stu \*next;

};

/\*定义输入的要修改的学生信息的结构\*/

struct stu\_change{

char id[15];

char sub[20];

float score;

};

struct stu \*head=NULL;

struct stu\_change s[10];

void creat(int n);

void show(int n);

void sshow(int n);

void change();

void sum\_and\_avg(int n);

/\*修改学生信息的函数\*/

void change()

{

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

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

{

scanf("%s",s[i].id);

scanf("%s",s[i].sub);

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

}

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

{

struct stu \*p=(struct stu\*)malloc(sizeof(struct stu));

p=head;

while(p)

{

if(strcmp(p->id,s[i].id)==0)

{

if(strcmp(s[i].sub,"English")==0)

{p->english=s[i].score;break;}

else if(strcmp(s[i].sub,"Math")==0)

{p->math=s[i].score;break;}

else if(strcmp(s[i].sub,"Physics")==0)

{p->phy=s[i].score;break;}

else if(strcmp(s[i].sub,"C")==0)

{p->c=s[i].score;break;}

}

p=p->next;

}

}

}

/\*求各学生的总分和平均分的函数\*/

void sum\_and\_avg(int n)

{

struct stu \*p=head;

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

{

p->sum=p->english+p->math+p->phy+p->c;

p->avg=p->sum/4;

p=p->next;

}

}

/\*建立一个单向链表\*/

void creat(int n)

{

struct stu \*p=head;

scanf("%s",head->id);

scanf("%s",head->name);

scanf("%f",&head->english);

scanf("%f",&head->math);

scanf("%f",&head->phy);

scanf("%f",&head->c);

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

{

p->next=(struct stu\*)malloc(sizeof(struct stu));

p=p->next;

scanf("%s",p->id);

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

scanf("%f",&p->english);

scanf("%f",&p->math);

scanf("%f",&p->phy);

scanf("%f",&p->c);

}

p->next=NULL;

}

int main()

{

head=(struct stu\*)malloc(sizeof(struct stu));

int n;

scanf("%d",&n);

creat(n);

show(n);

change();

printf("Alter:\n");

show(n);

sum\_and\_avg(n);

printf("SumAndAvg:\n");

sshow(n);

return 0;

}

/\*输出学生信息的函数\*/

void show(int n)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n","ID","Name",

"English","Math","Physics","C");

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->english);

printf("%-10.2f",p->math);

printf("%-10.2f",p->phy);

printf("%-10.2f\n",p->c);

p=p->next;

}

printf("\n");

}

/\*输出学生总分和平均分的函数\*/

void sshow(int n)

{

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

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->sum);

printf("%-10.2f\n",p->avg);

p=p->next;

}

printf("\n");

}

3）测试

（a） 测试数据：

选择测试的数据如表7-2所示。

表7-2 编程设计题2的测试数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | | 理 论 结 果 | 运 行 结 果 |
| n | 学生信息 | a | 学生信息 |
| 用例1 | 5 | U20140101 ZhangChuanChao 85 86 87 88 | 3 | U20140101  Math  95.6 | 输出学生成绩  输出修改后的学生成绩  输出学生的总成绩和平均分 | 截图  7-4 |
| U20140126 MaiDouDou  99 99 99 99 |
| U20140359  C  100 |
| U20140158 XiaoDouDou  56 85 89 59 |
| U20140312 DaoDaoDog  84 89 65 100 |
| U20140359  English  100 |
| U20140359 XiDaDa  88.8 88.8 88.8 88.8 |

（b） 对应测试测试用例1的运行结果如图7-4所示。

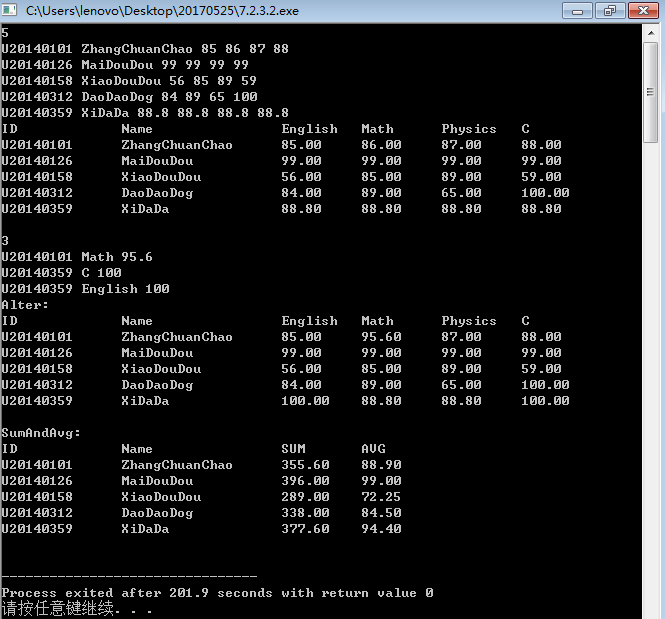


图7-4 编程设计题1的测试用例一的运行结果

### 7.2.4 选做题

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

**解答：**

1) 算法流程如图7.5所示。

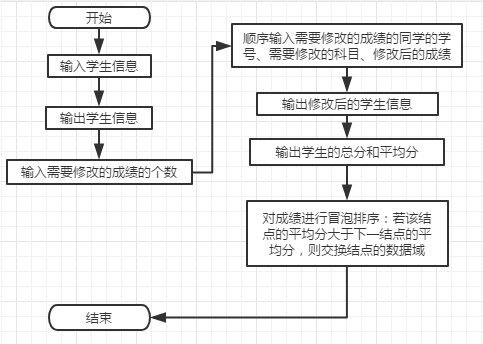


图7-5 选做题1的程序流程图

1. 源程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/\*定义存入学生信息的结构\*/

struct stu{

char id[15];

char name[20];

float english;

float math;

float phy;

float c;

float sum;

float avg;

struct stu \*next;

};

/\*定义输入的要修改的学生信息的结构\*/

struct stu\_change{

char id[15];

char sub[20];

float score;

};

struct stu \*head=NULL;

struct stu\_change s[10];

void creat(int n);

void show(int n);

void sshow(int n);

void ssshow(int n);

void change();

void sum\_and\_avg(int n);

void sort(struct stu \*head);

/\*修改学生信息的函数\*/

void change()

{

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

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

{

scanf("%s",s[i].id);

scanf("%s",s[i].sub);

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

}

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

{

struct stu \*p=(struct stu\*)malloc(sizeof(struct stu));

p=head;

while(p)

{

if(strcmp(p->id,s[i].id)==0)

{

if(strcmp(s[i].sub,"English")==0)

{p->english=s[i].score;break;}

else if(strcmp(s[i].sub,"Math")==0)

{p->math=s[i].score;break;}

else if(strcmp(s[i].sub,"Physics")==0)

{p->phy=s[i].score;break;}

else if(strcmp(s[i].sub,"C")==0)

{p->c=s[i].score;break;}

}

p=p->next;

}

}

}

/\*用交换结点数据域的方法升序排序的函数\*/

void sort(struct stu \*head)

{

struct stu \*p1=head,\*p2;

int len=0,i,j;

char \*temp1=(char\*)malloc(15);

char \*temp2=(char\*)malloc(20);

float temp3;float temp4;float temp5;float temp6;float temp7;float temp8;

while(p1){len++;p1=p1->next;}

for(i=0,p1=head;i<len-1;i++,p1=p1->next)

for(j=i+1,p2=p1->next;j<len;j++,p2=p2->next)

if(p1->avg>p2->avg)

{

strcpy(temp1,p1->id);strcpy(p1->id,p2->id);strcpy(p2->id,temp1);

strcpy(temp2,p1->name);strcpy(p1->name,p2->name);

strcpy(p2->name,temp2);

temp3=p1->english;p1->english=p2->english;p2->english=temp3;

temp4=p1->math;p1->math=p2->math;p2->math=temp4;

temp5=p1->phy;p1->phy=p2->phy;p2->phy=temp5;

temp6=p1->c;p1->c=p2->c;p2->c=temp6;

temp7=p1->sum;p1->sum=p2->sum;p2->sum=temp7;

temp8=p1->avg;p1->avg=p2->avg;p2->avg=temp8;

}

}

/\*求各学生的总分和平均分的函数\*/

void sum\_and\_avg(int n)

{

struct stu \*p=head;

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

{

p->sum=p->english+p->math+p->phy+p->c;

p->avg=p->sum/4;

p=p->next;

}

}

/\*建立一个单向链表\*/

void creat(int n)

{

struct stu \*p=head;

scanf("%s",head->id);

scanf("%s",head->name);

scanf("%f",&head->english);

scanf("%f",&head->math);

scanf("%f",&head->phy);

scanf("%f",&head->c);

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

{

p->next=(struct stu\*)malloc(sizeof(struct stu));

p=p->next;

scanf("%s",p->id);

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

scanf("%f",&p->english);

scanf("%f",&p->math);

scanf("%f",&p->phy);

scanf("%f",&p->c);

}

p->next=NULL;

}

int main()

{

head=(struct stu\*)malloc(sizeof(struct stu));

int n;

scanf("%d",&n);

creat(n);

show(n);

change();

printf("Alter:\n");

show(n);

sum\_and\_avg(n);

printf("SumAndAvg:\n");

sshow(n);

sort(head);

printf("Sort:(交换数据域)\n");

ssshow(n);

return 0;

}

/\*输出学生信息的函数\*/

void show(int n)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n",

"ID","Name","English","Math","Physics","C");

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->english);

printf("%-10.2f",p->math);

printf("%-10.2f",p->phy);

printf("%-10.2f\n",p->c);

p=p->next;

}

printf("\n");

}

/\*输出学生总分和平均分的函数\*/

void sshow(int n)

{

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

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->sum);

printf("%-10.2f\n",p->avg);

p=p->next;

}

printf("\n");

}

/\*输出排序后学生信息的函数\*/

void ssshow(int n)

{

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

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f\n",p->avg);

p=p->next;

}

printf("\n");

}

3）测试

（a） 测试数据：

选择测试的数据如编程设计题2的测试数据表7-2所示。

（b） 对应测试测试用例1的运行结果如图7-6所示。

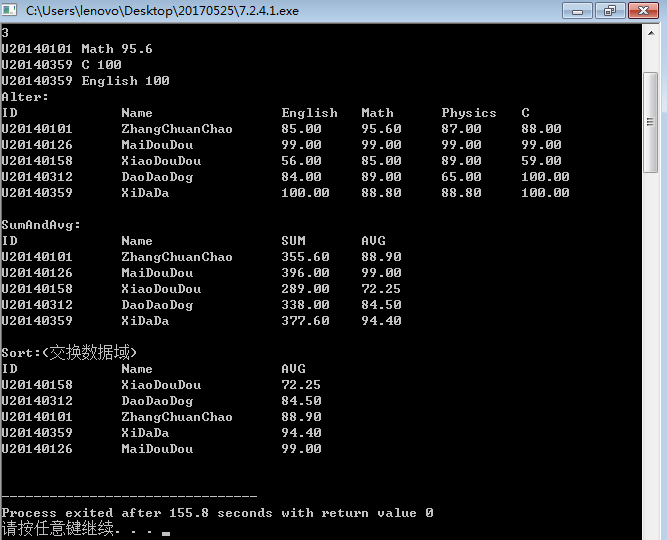


图7-6 选做题1的测试用例一的运行结果

（2）对选做题第（1）题，进一步写出用交换结点指针域的方法升序排序的函数。

**解答：**

1) 算法流程如图7.5所示。

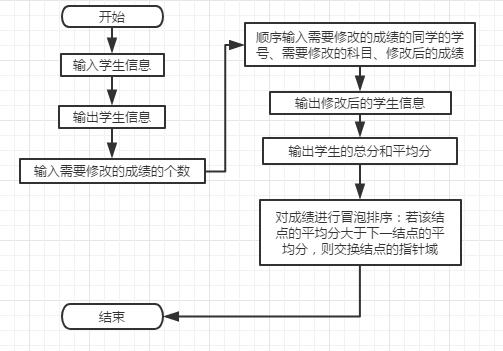


图7-7 选做题1的程序流程图

1. 源程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/\*定义存入学生信息的结构\*/

struct stu{

char id[15];

char name[20];

float english;

float math;

float phy;

float c;

float sum;

float avg;

struct stu \*next;

};

/\*定义输入的要修改的学生信息的结构\*/

struct stu\_change{

char id[15];

char sub[20];

float score;

};

struct stu \*head=NULL;

struct stu\_change s[10];

void creat(int n);

void show(int n);

void sshow(int n);

void change();

void sum\_and\_avg(int n);

/\*修改学生信息的函数\*/

void change()

{

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

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

{

scanf("%s",s[i].id);

scanf("%s",s[i].sub);

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

}

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

{

struct stu \*p=(struct stu\*)malloc(sizeof(struct stu));

p=head;

while(p)

{

if(strcmp(p->id,s[i].id)==0)

{

if(strcmp(s[i].sub,"English")==0)

{p->english=s[i].score;break;}

else if(strcmp(s[i].sub,"Math")==0)

{p->math=s[i].score;break;}

else if(strcmp(s[i].sub,"Physics")==0)

{p->phy=s[i].score;break;}

else if(strcmp(s[i].sub,"C")==0)

{p->c=s[i].score;break;}

}

p=p->next;

}

}

}

/\*用交换结点指针域的方法升序排序的函数\*/

void sort(struct stu \*\*headp)

{

struct stu \*prior1,\*prior2,\*p1,\*p2,\*t;

int i=0;

p1=(struct stu\*)malloc(sizeof(struct stu));

p1->next=\*headp;

(\*headp)=prior1=p1;

for(p1=prior1->next;p1->next!=NULL;prior1=p1,p1=p1->next)

for(p2=p1->next,prior2=p1;p2!=NULL;prior2=p2,p2=p2->next)

if(p1->sum>p2->sum)

{

t=p2->next;

prior1->next=p2;

prior2->next=p1;;

p2->next=p1->next;

p1->next=t;

t=p1;p1=p2;p2=t;

}

p1=(\*headp);

(\*headp)=(\*headp)->next;

free(p1);

}

/\*求各学生的总分和平均分的函数\*/

void sum\_and\_avg(int n)

{

struct stu \*p=head;

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

{

p->sum=p->english+p->math+p->phy+p->c;

p->avg=p->sum/4;

p=p->next;

}

}

/\*建立一个单向链表\*/

void creat(int n)

{

struct stu \*p=head;

scanf("%s",head->id);

scanf("%s",head->name);

scanf("%f",&head->english);

scanf("%f",&head->math);

scanf("%f",&head->phy);

scanf("%f",&head->c);

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

{

p->next=(struct stu\*)malloc(sizeof(struct stu));

p=p->next;

scanf("%s",p->id);

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

scanf("%f",&p->english);

scanf("%f",&p->math);

scanf("%f",&p->phy);

scanf("%f",&p->c);

}

p->next=NULL;

}

int main()

{

head=(struct stu\*)malloc(sizeof(struct stu));

int n;

scanf("%d",&n);

creat(n);

show(n);

change();

printf("Alter:\n");

show(n);

sum\_and\_avg(n);

printf("SumAndAvg:\n");

sshow(n);

return 0;

}

/\*输出学生信息的函数\*/

void show(int n)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n","ID","Name",

"English","Math","Physics","C");

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->english);

printf("%-10.2f",p->math);

printf("%-10.2f",p->phy);

printf("%-10.2f\n",p->c);

p=p->next;

}

printf("\n");

}

/\*输出学生总分和平均分的函数\*/

void sshow(int n)

{

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

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->sum);

printf("%-10.2f\n",p->avg);

p=p->next;

}

printf("\n");

}

/\*输出排序后学生信息的函数\*/

void ssshow(int n)

{

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

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f\n",p->avg);

p=p->next;

}

printf("\n");

}

3）测试

（a） 测试数据：

选择测试的数据如编程设计题2的测试数据表7-2所示。

（b） 对应测试测试用例1的运行结果如图7-8所示。

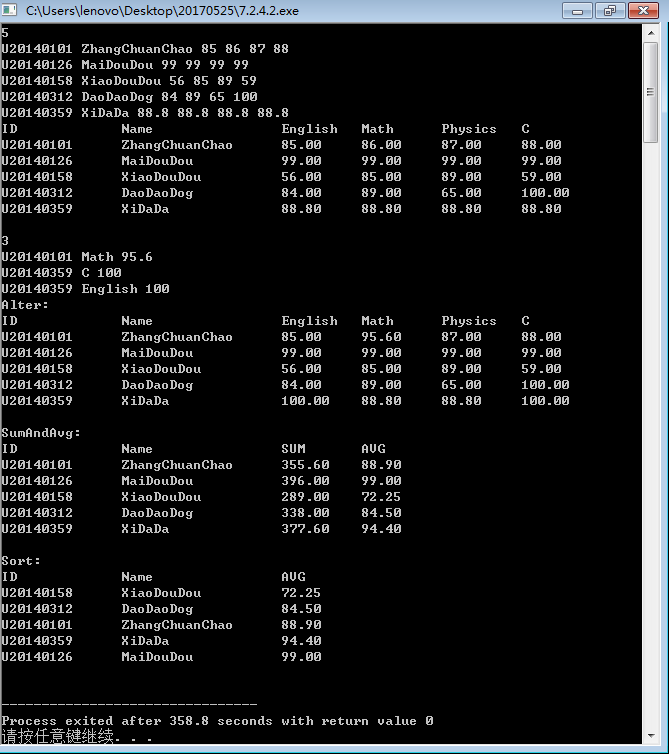


图7-8 选做题2的测试用例一的运行结果

（3）采用双向链表重做编程设计题第（2）题。

## 7.3 自设题

**（1）**自设实验题目： 输入学生信息，并在其中加入另一个学生的信息。

**（2）**实验目的： 练习插入结点。

**（3）**实验程序：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/\*存储学生信息的结构\*/

struct stu{

char id[15];

char name[20];

float english;

float math;

float phy;

float c;

struct stu \*next;

};

struct stu \*head=NULL;

void creat(int n);

void show(int n);

struct stu\* insert(char\* a);

/\*增加结点的函数\*/

struct stu\* insert(char\* a)

{

struct stu \* current=head,\*after,\*other;

while((strcmp(current->id,a))!=0&&current!=NULL)

current=current->next;

if(!current)return NULL;

after=current->next;

other=(struct stu\*)malloc(sizeof(struct stu));

printf("对新成员赋值：\n");

scanf("%s",other->id);

scanf("%s",other->name);

scanf("%f",&other->english);

scanf("%f",&other->math);

scanf("%f",&other->phy);

scanf("%f",&other->c);

if(after)

{

other->next=after;

current->next=other;

}

else

{

other->next=NULL;

current->next=other;

}

return current;

}

/\*创建单向链表\*/

void creat(int n)

{

struct stu \*p=head,\*\*headp=&head;

scanf("%s",head->id);

scanf("%s",head->name);

scanf("%f",&head->english);

scanf("%f",&head->math);

scanf("%f",&head->phy);

scanf("%f",&head->c);

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

{

p->next=(struct stu\*)malloc(sizeof(struct stu));

p=p->next;

scanf("%s",p->id);

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

scanf("%f",&p->english);

scanf("%f",&p->math);

scanf("%f",&p->phy);

scanf("%f",&p->c);

}

p->next=NULL;

}

int main()

{

head=(struct stu\*)malloc(sizeof(struct stu));

int n;

scanf("%d",&n);

creat(n);

show(n);

char a[15];

printf("插入位置的上一个学生的学号：");

scanf("%s",a);

insert(a);

show(n+1);

return 0;

}

/\*按格式输出学生信息的函数\*/

void show(int n)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n",

"ID","Name","English","Math","Physics","C");

struct stu \*p=head;

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

{

printf("%-15s",p->id);

printf("%-20s",p->name);

printf("%-10.2f",p->english);

printf("%-10.2f",p->math);

printf("%-10.2f",p->phy);

printf("%-10.2f\n",p->c);

p=p->next;

}

printf("\n");

}

**（4）**实验用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程序输入 | | 理论结果 | 运 行 结 果 |
| n | 学生信息 |
| 用例1 | 3 | U123 ZHANG  90 90 90 90 | 输出学生成绩  输出修改后的学生成绩  输出学生的总成绩和平均分 | 截图7-9 |
| U321 LI  80 80 80 80 |
| U345 WANG  100 100 100 100 |
| 插入的学生信息 | 前一位学号为U123  U178 LIU  90 90 100 100 |

运行结果如图

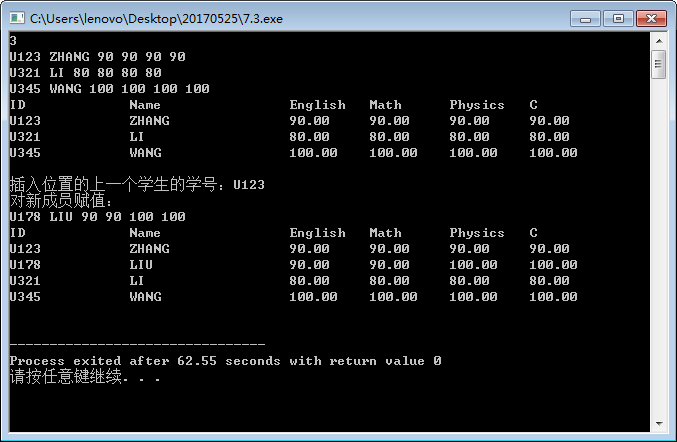


图7-9 自设题的测试用例一的运行结果

**（5）**实验结论：在插入结点时插入时，注意指针的改变有没影响别的指针也跟着变化而带来的指针指错了位置，在删除结点时要记得使用free。

## 7.4 实验小结

**7.4.1**在使用单向链表的时候应注意对头指针的使用。在编写稍长一点的代码比如本次实验中的学生信息管理系统，需要注意在不同函数中对数据进行处理后对实参的影响，同时要注意各变量的作用域的差别。

**7.4.2**关于双向链表，许多理解还不够到位，需要多加了解和练习。

# 实验8 文件实验

## 8.1 实验目的

1．熟悉文本文件和二进制文件在磁盘中的存储方式；

2．熟练掌握流式文件的读写方法。

## 8.2 实验题目及要求

### 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);

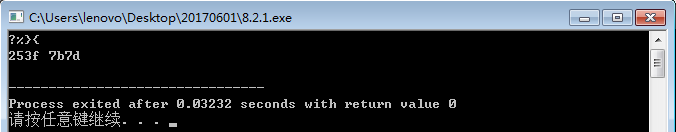
return 0;

}

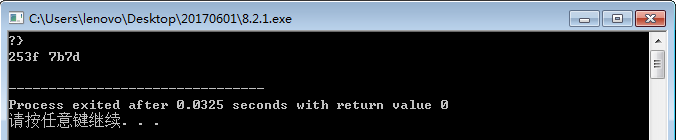
1. 请思考程序的输出结果，然后通过上机运行来加以验证。

ASCII字符编码表中可以查到，3f对应的是？,25对应的是%，7d对应的是}，7b对应的是{。因此第一行输出应为：?%{}.

第二行是直接输出a和b的十六进制形式，即为：253f 7b7d.

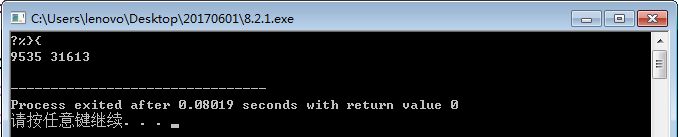


1. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？



因为short的字节长度为2，char的字节长度为1,所以在读取时只读取的一半。

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



%hx是指输出十六进制形式，%d是指输出十进制的整数形式。

### 8.2.2 源程序修改替换题

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. }

**解答：**

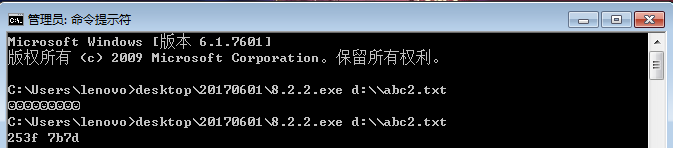
（1）错误修改：

1) 第16行的while循环条件少一组括号，正确形式为：

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

（2）错误修改后运行结果：

（第一行输出为修改前的运行结果，第二行为修改后的运行结果）



（2）用输入输出重定向freopen改写上述源程序中的main函数。

**解答：**

替换后的程序如下所示：

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if(freopen(argv[1],"r",stdin)==NULL){ /\*使stdin指向目标文件\*/

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

exit(-1);

}

while((ch=getchar())!=EOF) /\* 从stdin中读字符 \*/

putchar(ch);

fclose(stdin); /\* 关闭stdin \*/

return 0;

}



### 8.2.3 编程设计题

1. 从键盘输入一行英文句子，将每个单词的首字母换成大写字母，然后输出到一个磁盘文件“test”中保存。

**解答：**

1） 算法流程如图8.1所示。

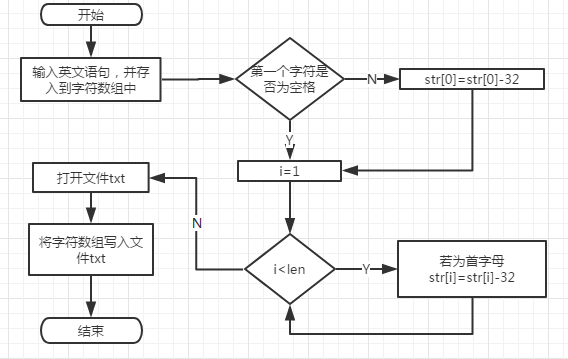


图8-1 编程设计题1的程序流程图

1. 源程序清单

#include<stdio.h>

#include<stdlib.h>

int main()

{

char str[100];

FILE \*fp;

int i,len=0;

while((str[len]=getchar())!='\n')

len++;

if(str[0]!=' ')str[0]=str[0]-32;

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

if(str[i-1]==' '&&str[i]!=' '&&str[i]!='\n')

str[i]=str[i]-32;

fp=fopen("d:\\test.txt","w+");

i=0;

while(str[i]!='\n')

{

fputc(str[i],fp);

i++;

}

fclose(fp);

}

3）测试

（a） 测试数据：

选择两组测试数据，一组以空格开头，一组以字母开头。如表8-1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| str |
| 用例1 | my name is ann. | My Name Is Ann. | 截图8-1 |
| 用例2 | this is c class. | This Is C Class. | 截图8-2 |

表8-1 编程设计题1的测试数据

（b） 对应测试测试用例1的运行结果如图8-1所示。

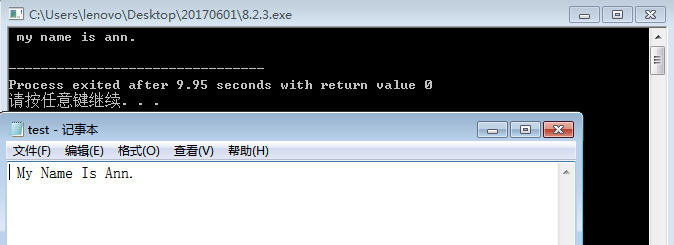


图8-1 编程设计题1的测试用例一的运行结果

对应测试测试用例2的运行结果如图8-2所示。

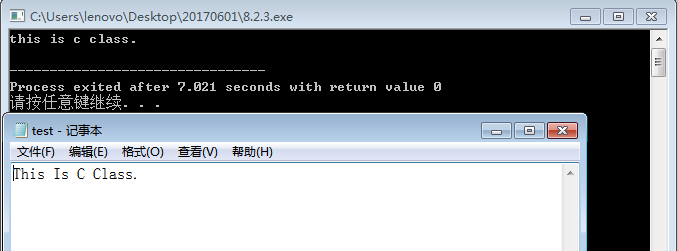


图8-2 编程设计题1的测试用例二的运行结果

## 8.3 实验小结

**8.3.1**文件实验中遇到的一大困难是对重定向和标准输入stdin、标准输出stdout的理解，其次是对文件指针的理解和应用。

# 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