

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 信安2班**

**学 号： U201614836**

**姓 名： 林晓斌**

**指导教师： 祝 建 华**

**报告日期：**

**计算机科学与技术学院**

目 录

[1 表达式和标准输入与输出实验](#_Toc4929)

[1.1 实验目的](#_Toc12617)

[1.2 实验内容](#_Toc15083)

[1.2.1 源程序改错](#_Toc6535)

[1.2.2 源程序修改替换](#_Toc22301)

[1.2.3 程序设计](#_Toc28239)

[1.3实验心得](#_Toc25675)

[2 流程控制实验](#_Toc3404)

[2.1 实验目的](#_Toc30809)

[2.2 实验内容](#_Toc12335)

[2.2.1源程序改错题](#_Toc23591)

[2.2.2源程序修改替换题](#_Toc25219)

[2.2.3编程设计题](#_Toc25236)

[2.2.4 选做题牛顿近似法](#_Toc21283)

[2.3实验心得](#_Toc28832)

[3 函数与程序结构实验](#_Toc18009)

[3.1实验目的](#_Toc1938)

[3.2、实验内容](#_Toc21994)

[3.2.1源程序改错题](#_Toc26953)

[3.2.2源程序修改替换题](#_Toc22179)

[3.3.3跟踪调试题](#_Toc28710)

[3.3.4编程设计题](#_Toc14107)

[3.3.5选做题](#_Toc16174)

[3.3实验心得](#_Toc2510)

[4 编译预处理实验](#_Toc21589)

[4.1实验目的](#_Toc15545)

[4.2实验题目及要求](#_Toc5653)

[4.2.1．源程序改错题](#_Toc21273)

[4.2.2．源程序修改替换题](#_Toc32440)

[4.2.3．跟踪调试题](#_Toc23047)

[4.2.4．编程设计题](#_Toc23299)

[4.3实验心得](#_Toc22273)

[5 数组实验](#_Toc32546)

[5.1 实验目的](#_Toc10142)

[5.2 实验内容及要求](#_Toc11266)

[5.2.1 源程序改错](#_Toc17405)

[5.2.2 源程序完善、修改、替换](#_Toc28803)

[5.2.3 跟踪调试源程序](#_Toc8636)

[5.2.4 程序设计](#_Toc27938)

[5.2.5 选做题](#_Toc9068)

[5.3实验心得](#_Toc22028)

[6 指针实验](#_Toc8415)

[6.1实验目的](#_Toc11966)

[6.2实验内容](#_Toc14028)

[6.2.1源程序改错题](#_Toc21396)

[6.2.2源程序完善、修改、替换题](#_Toc16531)

[6.2.3跟踪调试题](#_Toc18688)

[6.2.4编程设计题](#_Toc13612)

[6.2.5 选做题](#_Toc17723)

[6.3 实验心得](#_Toc10376)

[7 结构与联合实验](#_Toc20647)

[7.1实验目的](#_Toc31972)

[7.2实验题目及要求](#_Toc31309)

[7.2.1表达式求值的程序验证题](#_Toc27623)

[7.2.2源程序修改替换题](#_Toc18696)

[7.2.3编程设计题](#_Toc3459)

[7.2.4 选做题](#_Toc5993)

[7.3 实验心得](#_Toc29414)

[8 文件实验](#_Toc14163)

[8.1 实验目的](#_Toc24073)

[8.2 实验题目及要求](#_Toc25570)

[8.2.1 文件类型的程序验证题](#_Toc8859)

[8.2.2 源程序修改、替换](#_Toc31512)

[8.2.3 程序设计](#_Toc25431)

[8.3实验心得](#_Toc1637)

[9 参考文献](#_Toc14093)

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

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

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 voidmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

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

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

13 /\* for task 2 \*/

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

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

16 s = PI \* r \* r;

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

18 /\* for task 3 \*/

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

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

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

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

}

**解答**：本实验共有9个错误，修改数为10。

第2行#define定义符号常量指令行末尾不加分号。改为：#define PI 3.14159

第3行void main是非标准语句，需改为int main，同时最后加上return 0.

第10行scanf函数格式错误，应为：scanf("%d",&f);

第11行的5与9都是整型数，相除之后结果为0，故应将数字类型转换，改为：c=5.0/9 \*(f-32)

第15行r为双精度变量，故应改为：scanf(“%lf”,&r);

第17行printf语句不需要加地址符&，故改为：printf("\nThe acreage is %.2f\n\n",s);

第20行应取作16进制运算，故改为：scanf("%hx %hx", &k, &p );

newint变量并没有定义，故应在第6行声明它的定义：short p, k ,newint;

第21行转换式子出错，应为：newint = (p&0xff00)|(k&0xff00)>>8;

以下为经过修改后的源程序。

#include <stdio.h>

#include <stdlib.h>

#define PI 3.14159

int main(void)

{

int f;

short p,k,newint;

double c,r,s;

/\*for task1\*/

printf("Input Fahrenheit:");

scanf("%d",&f);

c=5.0/9\*(f-32);

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

/\*for task2\*/

printf("inout the radius r:");

scanf("%lf",&r);

s=PI\*r\*r;

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

/\*for task3\*/

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

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

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

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

return 0;

}

对修改后的程序进行一次简单的数据检验，如图1.1所示。

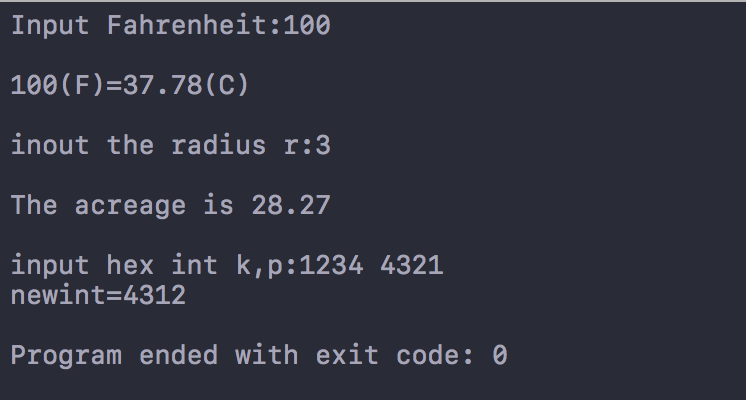


图 1.1 修改后的程序运行结果

1.2.2 源程序修改替换

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

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

t=a ；a=b；b=t；

prinf(“\na=%d,b=%d”,a,b);

}

**解答**：利用按位加运算可以完成，程序如下

#include <stdio.h>

#include <stdlib.h>

int main()

{

int a,b;

printf("Input two integers:");

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

a^=b;

b^=a;

a^=b;

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

return 0;

}

对完成的程序进行一次简单的程序验证，如图1.2所示。

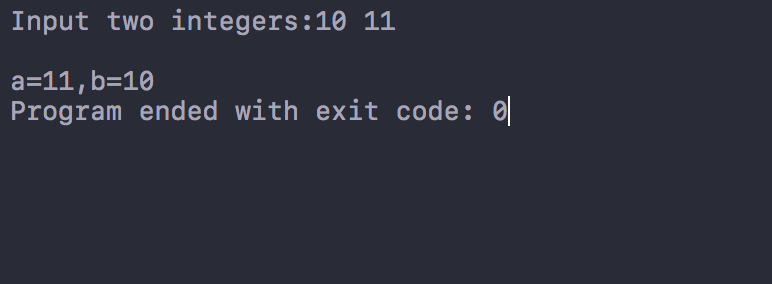


图1.2简单的程序验证

1.2.3 程序设计

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

**解答**:首先判断是否为大写字母，若为大写字母，则将该字母的ASCII码减去32，若不是，则直接输出。转化为流程图（图1.3）

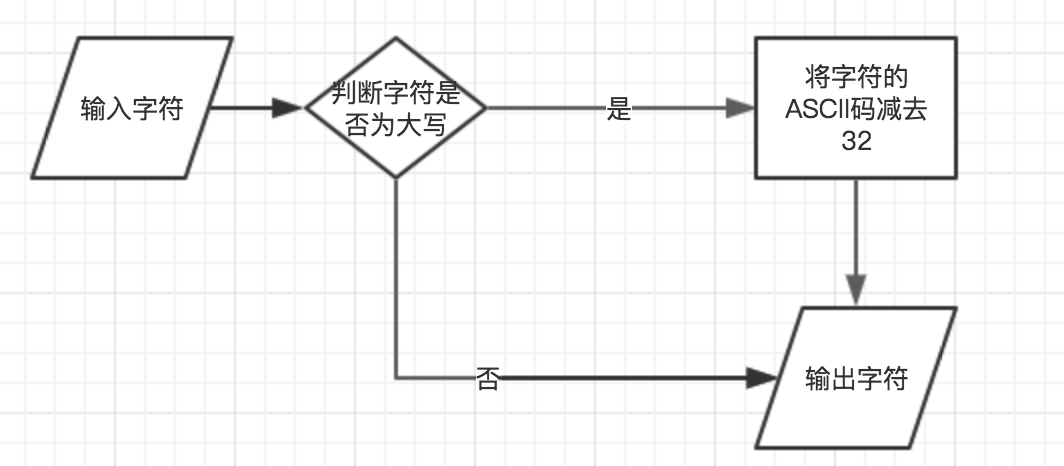


图1.3程序设计第一题的流程图

利用流程图思路写成源程序如下

#include <stdio.h>

#include <stdlib.h>

int main()

{

char x;

printf("Input your char:");

scanf("%c",&x);

if(x>='a'&&x<='z')

{

printf("%c\n",x);

}

else if(x>='A'&&x<='Z')

{printf("%c\n",x+32);}

else printf("please input right char");

return 0;

}

向程序输入小写的“a”，应该输出也是小写的“a”，实验结果（图1.4）与预设一致

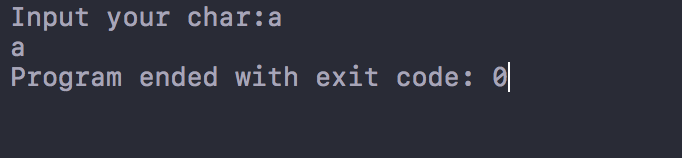


图1.4程序设计第一题的输出结果

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**：解题思路主要通过判断语句完成。

1.输入x，m，n，为了方便分析测试结果，x的输入采用16进制

2.如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，转2.1，否则转3.

2.1 首先x>>m，将要处理的n位移动到最右；

2.2 再将上一步的结果左移16-n位，即：x<<16-n;

2.3 用16进制输出结果并转4.

3. 显示输入错误信息；

4. 结束

依照解题思路写出源程序

#include <stdio.h>

#include <stdlib.h>

int main()

{

unsigned short int x,m,n;

printf("Please enter x(16进制,m(0~15),n(1~16-n):\n" );

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

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

{

x>>=m;

x<<=(16-n);

printf("%hx",x);

}

else printf("Error!\n");

return 0;

}

分三种不同的情况对源程序进行测试，对于输入的n、m值，分为m超边界条件，n超边界条件，m、n均不超出边界条件进行讨论。如表1-1所示

表1-1 编程题2的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 | 运 行 结 果 |
| X | m | N |
| 用例1 | 0100 0110 1000 0000（4680） | 7 | 4 | 计算结果1101 0000 0000 0000 即D000 | D000 或 图1.5 |
| 用例2 | 1101 0101 1000 0011（D583） | 16 | 1 | 输入错误（m值超范围） | 图1.6 |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） | 图1.7 |

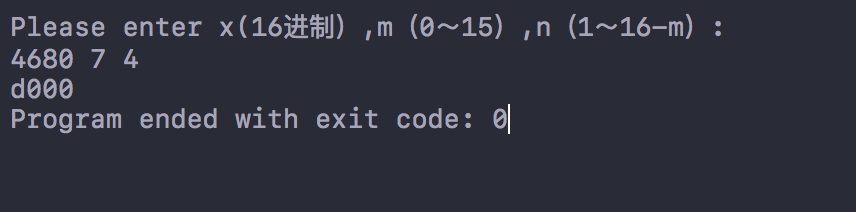


图 1.5 编程题2，m,n均不超出边界条件情况

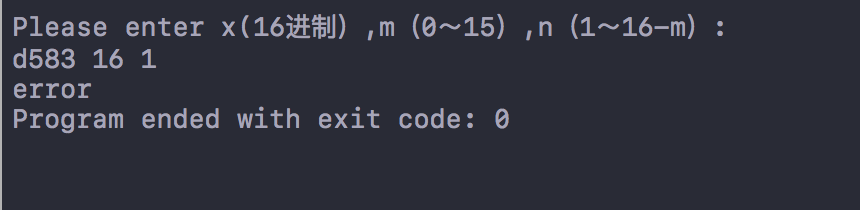


图 1.6编程题2，m超出边界条件情况

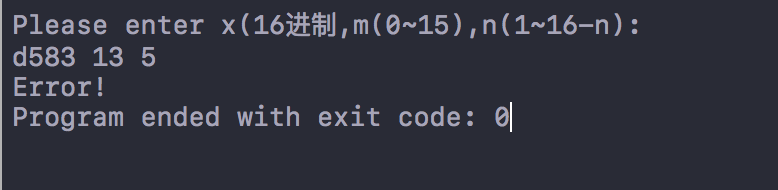


图 1.7编程题2，n超出边界条件情况

3 IP地址通常是4个用句点分隔的小整数（即点分十进制），如32.55.1.102。这些地址在0机器中用无符号长整数表示。编写一个程序，以机器存储的形式读入一个互联网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。例如：整形676879571的二进制表示是00101000 01011000 01011100 11010011.按照8位一组可表示40 88 92 211.由于CPU处理数据的差异，他的顺序是颠倒的，所以最终格式为211.92.88.40。

**解答**：本题依据mask的遮盖作用可以将IP地址逐位提出然后再将其反向重新组装，既得到答案。

#include <stdio.h>

#include <stdlib.h>

int main()

{

unsigned long IP;

unsigned short one,two,three,four;

printf("Please enter your IP:");

scanf("%lu",&IP);

one=(IP&0xFF000000)>>24;

two=(IP&0xFF0000)>>16;

three=(IP&0xFF00)>>8;

four=(IP&0xFF);

printf("%hu.%hu.%hu.%hu",four,three,two,one);

return 0;

}

对源程序进行测试，整形676879571的二进制表示是00101000 01011000 01011100 11010011.按照8位一组可表示40 88 92 211.由于CPU处理数据的差异，他的顺序是颠倒的，所以最终格式为211.92.88.40。故输出应为211.92.88.40，结果如图1.8.

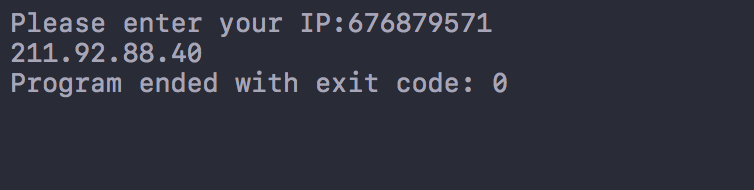


图 1.8 编程题3的输出结果

1.3实验心得

这次c上机实验给了我很多心得体会。其中，最为深刻的是程序的界面友好很关键。众所周知如果你不加以阐释，c语言的运行界面就是一片漆黑，别人根本不知道要输入什么数字，也不知道这些数字会被用来做什么。所以，一个良好的提示界面就显得尤为重要。以及，适当地在自己的代码后加上一点注释，有助于别人理解你的程序，自己以后打开也不会迷茫。

还有就是细心极为重要，这次改错的许多错误也是我经常犯的，比如语句打不打分号的问题等等。对于位运算，以前一直认为其作用不大，今天做了实验题之后我才真正了解到了位运算的简便以及高效，包括mask以及按位加的作用，位运算虽然比较费脑筋但是比起简单的循环确实是高效了许多。

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。

#include <stdio.h>

void main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d",n);

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

s=s\*i;

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

}

**解答：**以上程序有4处错误，其中第1个错误属于数值范围问题，n的阶乘往往很大，故用int作为其变量的类型不是很合适，将s改为long型更为合适。第二个为格式问题，scanf语句中需要使用地址符为变量赋值。第三个为for语句格式错误，for语句括号中需要用“；”隔开，不能省略也不能替换。第四个错误为return语句位置错误，return语句应在函数结束之前出现。下面列出修改后的源程序。

#include <stdio.h>

int main()

{

int i,n;//s为n！的值，int类型过于小，应改为long

long s=1;

printf("Please enter n:");

scanf("%d",&n);//scanf语句错误，缺少地址符&

for(i=1;i<=n;i++)//for 语句错误，中间应用；隔开

s=s\*i;

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

putchar('\n');

return 0;

}

//return 0 (return 函数应该在大括号内)

对程序输入n=10，则此时输出的结果应该是10！=3268800，程序测试如图2.1，与预测结果一致。

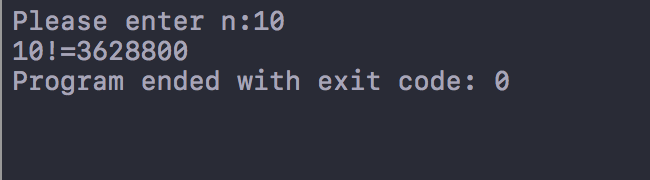


图2.1改错题修改后的程序测试

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。

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

**解答（1）**：相同的循环内容用不同的循环结构实现需要注意控制循环的条件及不同循环结构的格式。While语句循环结构流程图如图2.2

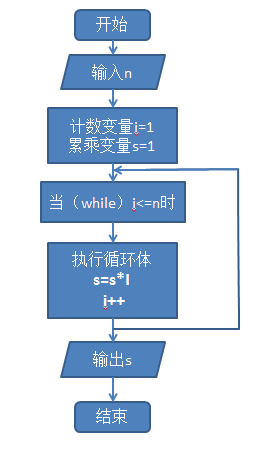


图2.2 while语句流程图

对第一题用while语句替换for语句

#include <stdio.h>

int main()

{

int i=1,n;

long s=1;

printf("Please enter n:");

scanf("%d",&n);

while (i<=n) {

s\*=i;

i++;

}

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

putchar('\n');

return 0;

}

do while语句流程图如图2.3

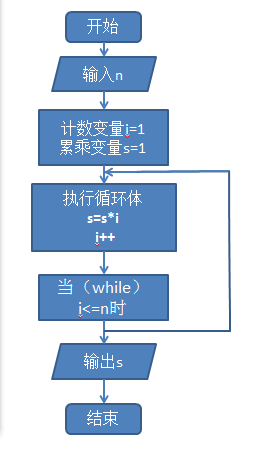


图2.3 do while语句流程图

对第一题的程序使用do while替换for语句

#include <stdio.h>

int main()

{

int i=1,n;

long s=1;

printf("Please enter n:");

scanf("%d",&n);

do {

s\*=i;

i++;

}while (i<=n);

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

putchar('\n');

return 0;

}

与第一题一样，使用n=10作为测试样例，此时输出的结果应该是10！=3268800，替换为while的程序测试如图2.4，替换为do while的程序测试如图2.5，均与预测结果一致。

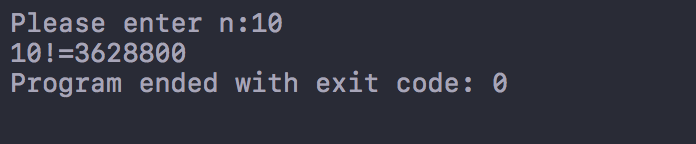


图 2.4 while的程序测试结果

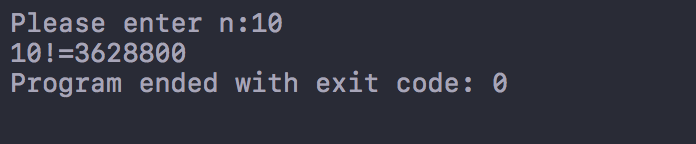


图 2.5 do while的程序测试结果

**解答（2）**：向求n！的循环中加入判断条件判断是否大于s，如果大于s则输出n，否则继续循环。流程图如图2.6。

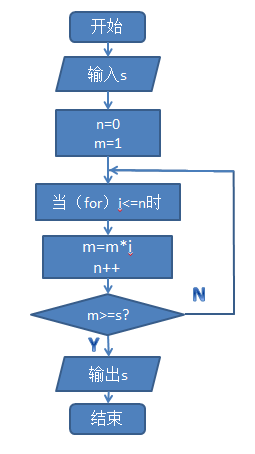


图2.6求最小n的流程图

#include <stdio.h>

int main()

{

int i,n=0;

long s1=1,s2;

printf("Please enter s:");

scanf("%d",&s2);

for (i=1; s1<s2; i++) {

s1\*=i;

n++;

}

printf("满足n!>=s的最小整数n=%d",n);

putchar('\n');

return 0;

}

对程序输入s=40310，因为8！=40320，所以程序应输出n=8，结果如图2.7.

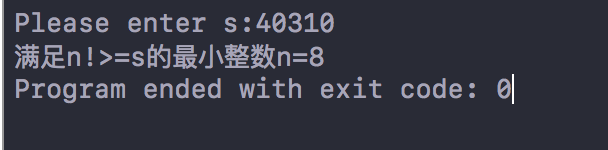


图2.7求最小n的程序测试

2.2.3编程设计题

（1）假设工资税金按以下方法计算：x<1000元，不收取税金；1000<=x<2000,收取5%的税金；2000<=x<3000,收取10%的税金；3000<=x<4000,收取15%的税金；4000<=x<5000,收取20%的税金；x>5000，收取25%的税金。编写一个程序，输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**该问题为数学分段函数问题，可以利用if语句和switch语句实现。用switch语句是需要注意case的设置。If语句流程图如图2.8所示，switch语句流程图如图2.9所示。

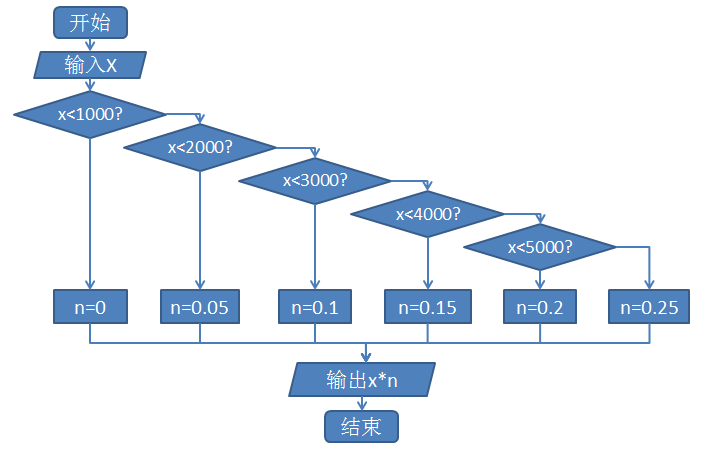


图2.8 if语句流程图

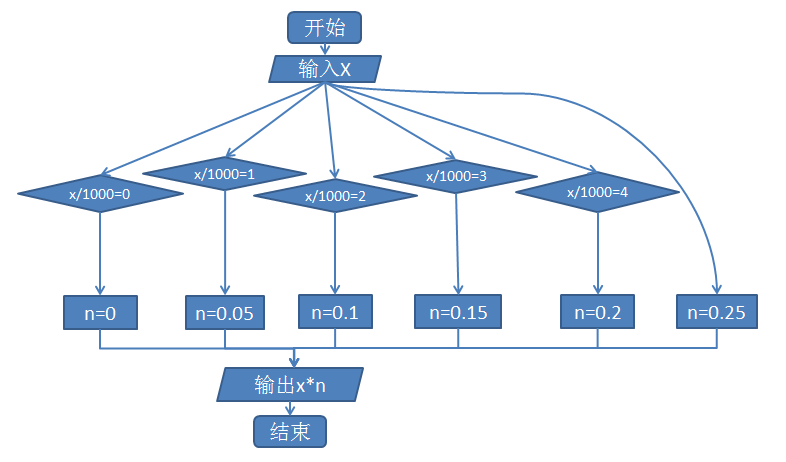


图2.9 switch语句流程图

if型源程序

#include <stdio.h>

int main(void)

{

float x,m;

printf("请输入您的工资金额:");

scanf("%f",&x);

if (x<1000)

m=0;

else if(x<2000)

m=x\*0.05;

else if(x<3000)

m=x\*0.1;

else if(x<4000)

m=x\*0.15;

else if(x<5000)

m=x\*0.2;

else m=x\*0.25;

printf("应收取税金额度：%f",m);

putchar('\n');

return 0;

}

switch型源程序

#include <stdio.h>

int main(void)

{

float x,m;

printf("输入工资金额:");

scanf("%f",&x);

switch ((int)(x/1000)) {

case 0:m=0;break;

case 1:m=x\*0.05;break;

case 2:m=x\*0.1;break;

case 3:m=x\*0.15;break;

case 4:m=x\*0.2;break;

default:m=x\*0.25;

break;

}

printf("应收取税金额:%f",m);

putchar('\n');

return 0;

}

对于该分段函数，选择代表性的100，3500，6000作为输入，输出结果应对应为0，525，1500。输入为100的输出结果见图2.10，输入为3500的输出结果见图2.11，输入为6000的输出结果见2.12。

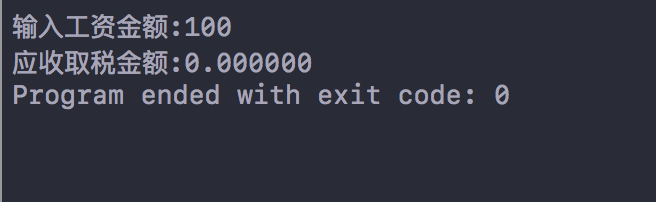


图2.10 工资为100时所收税金

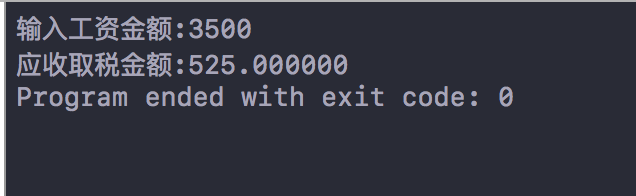


图 2.11工资为3500时所收税金

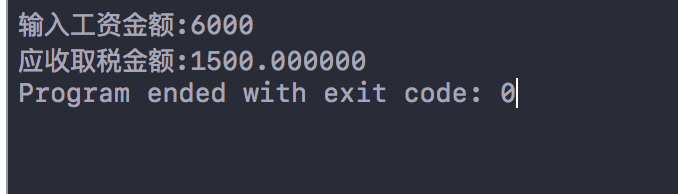


图2.12 工资为6000时所收的税金

（2）编写一个程序，将输入的一行字符复制到输出，复制过程中将一个以上的空格字符用一个空格代替。

**解答**：通过状态机判断不同空格的作用，从而消去一些空格的作用。流程图如图2.13。

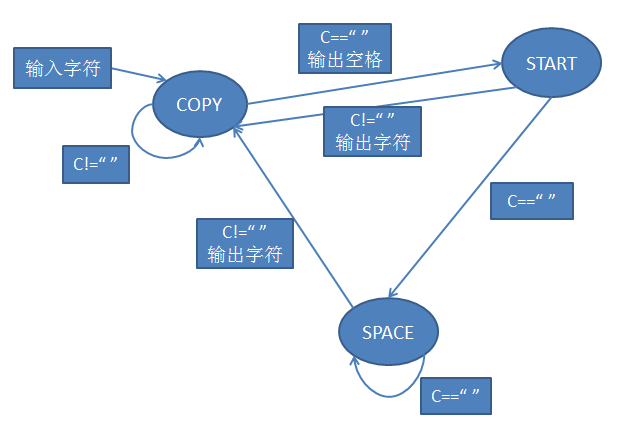


图2.13程序设计题2的流程图

根据流程图编写的程序设计题2的程序如下所示

#include <stdio.h>

enum{copy,start,space};

int main(void)

{

char c;

int state=copy;

printf("please input chars:\n");

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

switch(state) {

case copy:

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

else putchar(c);

break;

case start:

putchar(' ');

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

else {state=copy;

putchar(c);}

break;

case space:

if (c!=' ') {

state=copy;

putchar(c);

}else state=space;

break;

}

return 0;

}

向程序中输入“ 324 as”和“ghsu sada s aaa sd”程序应该输出“ 324 as”和“ghsu sada s aaa sd”，程序测试结果如图2.14所示。

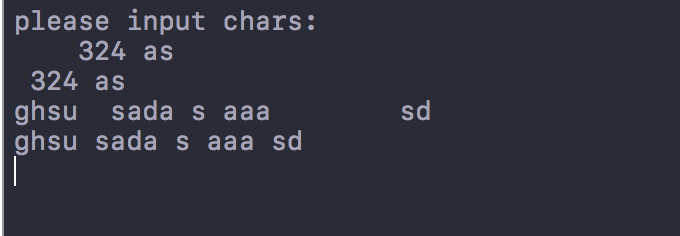


图2.14 将多个空格改为一个的程序结果

（3）编写一个程序，打印如下的杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

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)

**解答**：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。利用不同的变量分别对行、列进行定义利用循环及判断条件可以完成程序。

源代码如下所示。

#include <stdio.h>

int main(void){

int i,j,c,m,n=10;

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

{

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

{ printf(" ");}

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

if(j==0)c=1;

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

printf("%-4d",c);

}

printf("\n");

}

return 0;

}

将程序运行，查看“金字塔”效果。结果如图2.15所示。

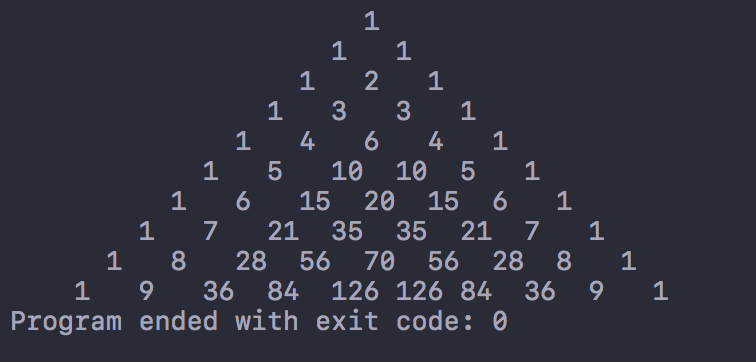


图2.15杨辉三角程序题输出结果

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

**解答**：多次除10取余，每次都分别输出余数，然后除以10并赋给原数重复计算，如流程图2.16所示。

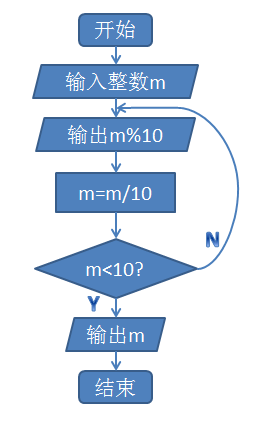


图2.16逆转正整数流程图

根据流程图写出程序如下。

#include <stdio.h>

int main(void)

{

long m,n;

printf("请输入一个正整数:");

scanf ("%ld",&n);

printf("逆转后的整数为：");

while (n>=10) {

m=n%10;

printf("%ld",m);

n/=10;

}

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

return 0;

}

输入1234，则根据题意应输出4321，程序测试结果如图2.17所示。

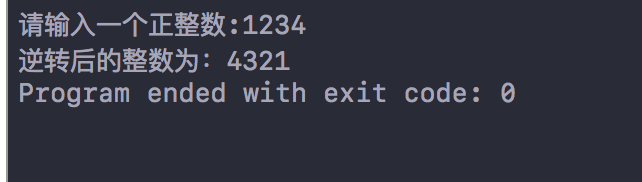


图2.17 逆转正整数

2.2.4 选做题牛顿近似法

编写一个程序，用牛顿迭代法求方程f(x)=3x^2-4x^2-5x+13=0满足精度1e-6的一个近似根，并在屏幕上输出所求近似根。

牛顿迭代法求方程近似根的迭代公式：

x0=a; x(k+1)=xk-f(xk)/f’(xk);

其中，f’(x)是函数f(x)的导函数。牛顿迭代法首先任意设定的一个实数a来作为近似根的迭代初值x0，然后用迭代公式计算下一个近似根x1。如此继续迭代计算x2，x3，...，xn，直到|xn-xn-1|<=精度e，此时值xn即为所求的近似根。

**解答**：利用do while循环可以不断地近似，再通过if语句判断直到最后一项|xn-1-xn|<e时程序终止，输出结果。其中需注意fx的表达形式，应选择效率最高的表达形式即运算量最少的形式。代码如下所示。

#include <stdio.h>

#define a 1e-6

int main(void)

{

double x,d;

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

scanf("%lf",&x);

do{

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

x+=d;

}while (fabs(d)>a);

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

return 0;

}

取1为牛顿迭代法的近似根，程序运行结果如图2.18所示。

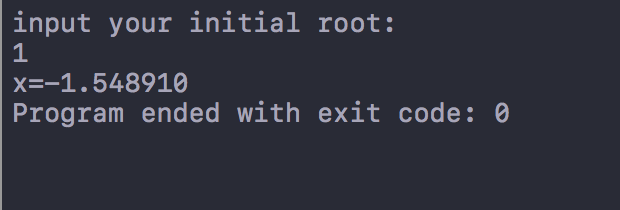


图2.18牛顿迭代法的运行结果

2.3实验心得

通过这次关于流程控制的试验，首先，我意识到了控制流程的方式不止有一种，在多种流程中我们应选择最为高效简便的方案，在写程序之前先思考完方案再开始写代码。感觉与写作文相同，先拟定写作手法再下笔。其次，有些题目看起来思路不是那么的清晰，切勿死磕，应该动动笔，试着画看看流程图，或者带入一些特殊值，尝试找找规律。最后，以前我写代码的时候经常会将循环语句后的“{}”混淆，导致程序出错时检查十分艰辛，流程变得混乱不堪，既不利于自己检查，别人也看得费劲。后来我的一个舍友教了我一个窍门，他让我将“{}”打在关键字下面，并且一一对齐，这样检查的时候就能轻松地分辨括号内对应的是什么函数，也不容易出错。这点我受益匪浅。

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**：本程序共有3处错误。除此之外，还有一处问题，即C99之后函数尽量不要用void型，所以定义函数有一个返回值0。

第2行错误，在引用副函数之前应该先声明副函数。

第12行错误，fac变量没有定义初始值，有可能会导致计算出错，故添加初始值为0。

第15行错误，s+=fac；这条语句应该在for循环语句里面否则不能起到累加的作用。

将以上错误进行修改得到程序如下。

#include "stdio.h"

long sum\_fac(int n);//先声明函数，否则函数无法使用

int main(void)

{

int k;

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

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

return 0;

}

long sum\_fac(int n)

{

long s=0;

int i;

long fac=1;//为fac赋初值，防止计算出错

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

{

fac\*=i;

s+=fac;//该语句应该在循环体里面

}

return s;

}

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**：sum\_fac函数为求1!+2!+3!+…+n!的函数，可以利用静态存储变量来简化运算。将sum\_fac函数修改如下所示。流程图如图3.1所示。

long sum\_fac(int n)

{

static long f=1;

f\*=n;

return f;

}

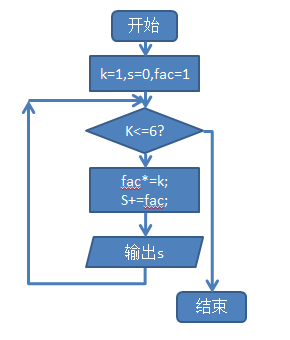


图3.1源程序修改替换第1题流程图

运行结果如图3.2所示。

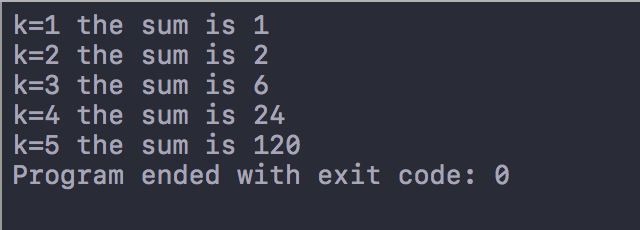


图3.2修改后编程题1的运行结果

（2）修改第1题中sum\_fac函数，计算

**解答**：修改sum\_fac使得n！可以单独求出，然后在主函数中求出它们倒数最后相加即得到。修改后的sum\_fac函数如下所示。流程图如图3.3所示。

long sum\_fac(int n)

{

int i;

long f=1;

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

return f;

}

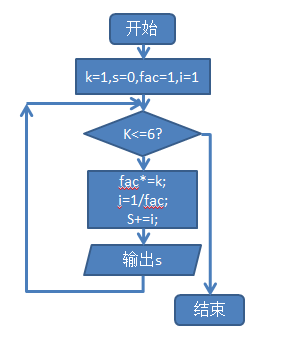


图3.3源程序修改替换题2流程图

运行结果如图3.4所示。

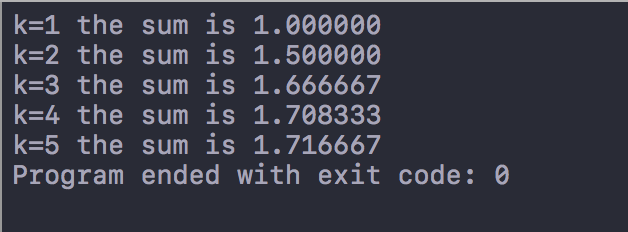


图3.4编程题2 的运行结果

3.3.3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）刚执行完scanf("%d",&k);语句，p值是0\*7fff5fbff798，i值是24630。如图3.5所示。

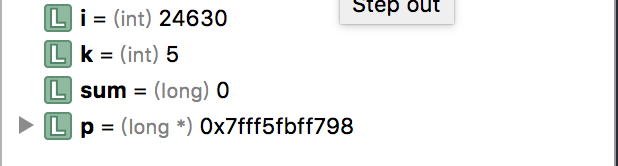


图3.5刚执行完scanf("%d",&k);语句，p,i值

（2）从fabonacci函数返回后光条停留在sum+=fabonacci(i);语句上，如图3.6所示。

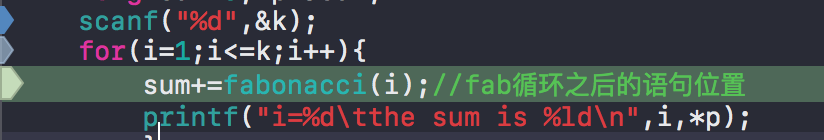


图3.6从fabonacci函数返回后的语句

（3）进入fabonacci函数，watch窗口显示的是n的值。

（4）当i=3，从调用fabonacci函数到返回，n值变化为3，2，3，1，3。

3.3.4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。

（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）利用辗转相除法求两个数的最大公约数。流程图如图3.7所示。

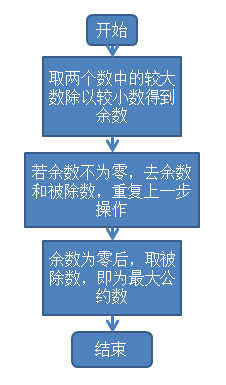


图3.7编程设计题1中辗转相除法流程图

根据流程图设计的程序代码如下所示。

#include <stdio.h>

int last (int m,int n){

int z,r;

if (m>=n&&m%n==0) z=n;

else if (m<=n&&m%n==0) z=m;

else

{

r=m%n;

m=n;

n=r;

z=last(m,n);

}

return z;

}

int main (void){

int m,n;

printf("please input two integers:\n");

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

printf("两整数的最大公约数为：%d\n",last(m,n));

}

程序测试样例为148，35以及46，23，其输出结果应为1，23。程序测试结果如图3.8和3.9所示。

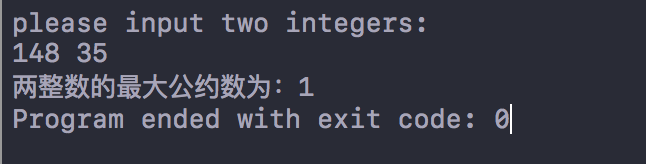


图3.8程序设计测试1

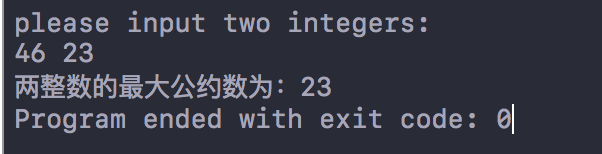


图3.9程序设计测试2

（2）首先先写出对于给定的偶数判断是否满足哥德巴赫猜想的式子。然后控制循环条件使其能在begin与end给定的数的范围中执行。判断是否满足哥德巴赫猜想流程图如图3.10。

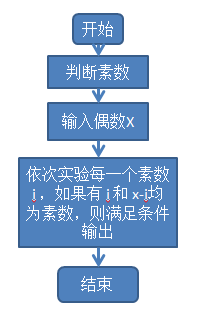


图3.10判断是否满足哥德巴赫猜想

由流程图写出程序如下。

#include <stdio.h>

int prime(int n)

{

int k;

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

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

return 1;

}

int main ()

{

int y,n,m,i;

printf("input your numbers range:");

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

for (i=n; i>=n&&i<=m; i++)

{

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

{

if (prime(y)&&prime(i-y))

{

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

break;

}

}

}

return 0;

}

程序测试样例为begin10，end20。输出如图3.11所示，均满足哥德巴赫猜想。

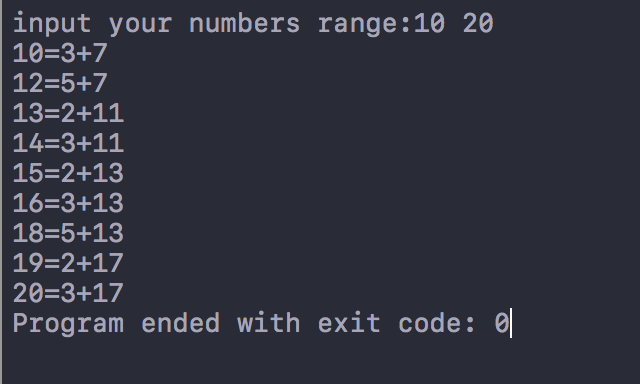


图3.11编程题2样例测试结果

3.3.5选做题

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**：由于我使用的IDE不是windows系统支持的，所以无法在dos环境下运行生成可运行文件。不过我在project中仍可运行上述程序，运行结果如图3.12所示，其中样例为“a”.

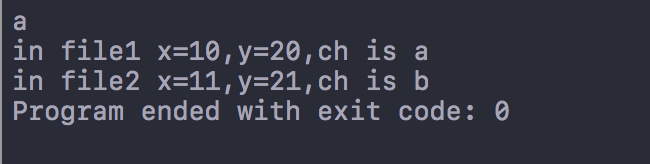


图3.12多文件编译的运行结果

3.3实验心得

首先，我觉得我在本次实验中对于副函数的认识更加全面，以前总是有“丢三落四”的习惯，常常在主函数前忘了定义副函数。而且对于副函数的计算量控制的也不是很好，源程序修改替换题1为我敲响了警钟，要追求更加简洁，计算量更小的程序。

其次，之前因为对Xcode这个IDE不熟悉，每次进行debug的时候就会莫名其妙进入汇编语言模式，弄得我丈二和尚摸不着头脑，专门请教了助教老师，他帮我把汇编模式关闭了，也算是个不大的收获吧。在写冰雹数的时候深刻体会到Debug对于调试程序错误时十分有效，可以定位到程序错误的点，这是最大的收获。

4 编译预处理实验

4.1实验目的

掌握文件包含、宏定义、条件编译、assert宏的使用；

练习带参数的宏定义、条件编译的使用；

练习assert宏的使用；

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实验题目及要求

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

5void 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);

}

**解答**：本实验共有4个错误，与以往一样函数的类型最好不要用void，在这里我们改为int，并添加返回值return 0；。第1个错误在第2.3行代码，定义宏时应该严格定义字符串防止产生歧义。第2个错误在第4行代码，定义宏swap并不能将两个数值互换，没能达到要求。第3个错误在第7行，t为多余变量，应该去掉。第4个错误在第9行代码，scanf语句中为a,b赋值的%d之间不需要加“,”。修改后的代码如下。

#include "stdio.h"

#define SUM ((a)+(b))

#define DIF ((a)-(b))

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

int main()

{

int a,b;

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

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

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

SWAP(a,b);

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

return 0;

}

利用a=2、b=3两个数字进行测试，输出结果应为平方差为-5，a=3，b=2。测试结果如图4.1。

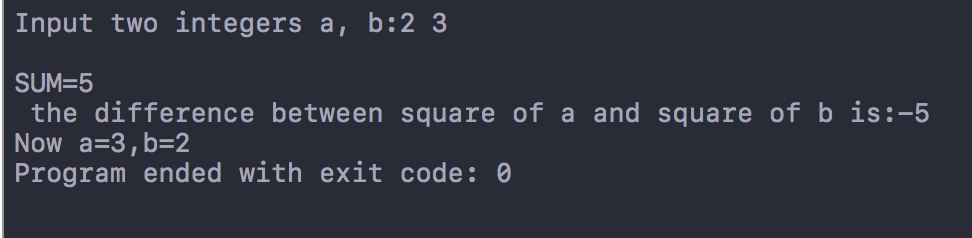


图4.1程序改错题测试

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。

1void main(void)

2{

3 int a, b, c;

4 float d, e;

5 printf("Enter three integers:");

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

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

8

9printf("Enter two floating point numbers:");

10scanf("%f,%f",&d,&e);

11printf("\nthe sum of them is %f\n",sum(d,e));

12}

13

14int max(int x, int y, int z)

15{

16 int t;

17 if (x>y)

18 t=x;

19 else

20 t=y;

21 if (t<z)

22 t=z;

23 return t;

24}

25float sum(float x, float y)

26{

27 return x+y;

28}

**解答**：本实验共有2个错误。第1个错误出现在第1行之前，如果副函数定义在主函数之后，则在主函数中使用副函数时应该在主函数之前先声明副函数。第2个错误出现在第6、10行的scanf语句中，%d之间与%f之间不用“，”隔开，应删去“，”。

定义带参数的宏max使得x，y，z三个形参进行比较后输出最大值，可利用“ ？ ； ”语句进行判断并输出。完成后程序如下。

#include "stdio.h"

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

double sum(double x, double y);

int main(void)

{

int a, b, c;

double 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("%lf %lf",&d,&e);

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

return 0;

}

double sum(double x, double y)

{

return x+y;

}

利用两组数据进行测试，第1组为a=1,b=3,c=7,d=1.568,e=3.534第2组为a=2,b=2,c=2,e=1.222,f=1.222。第1组的测试预测结果应该为max=7，sum=5.102000。程序测试结果如图4.2。第2组的测试预测结果应该为max=2，sum=2.4440000.程序测试结果如图4.3。

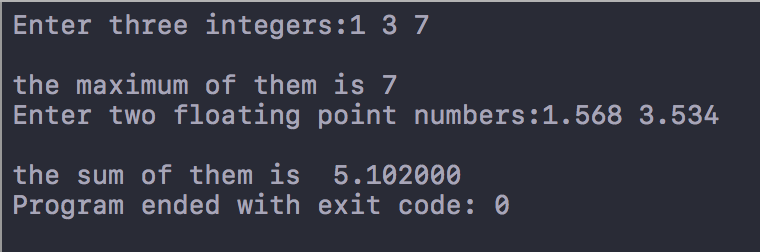


图4.2第1组数据测试结果

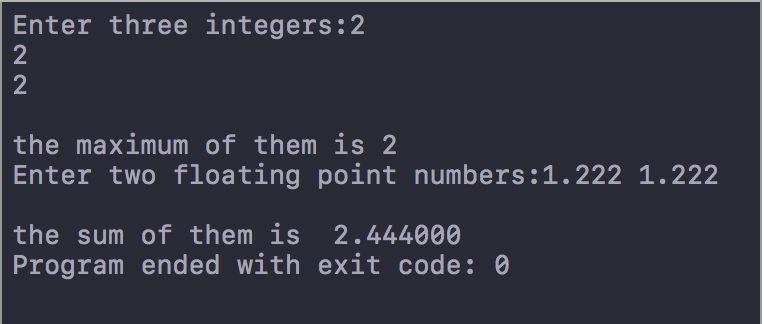


图4.3第2组数据测试结果

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；

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

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

解答：该程序缺少头文件，并且在主函数之前没有声明定义在其后的副函数。修改后程序如下。

#include "stdio.h"//

#include "assert.h"//

#define R

int integer\_fraction(float x);//

int main(void)//

{

float r, s;

int s\_integer=0;

printf("input a number:");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

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

s\_integer= integer\_fraction(s);

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

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

#endif

return 0;

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

该程序单步执行进入函数decimal\_fraction时watch窗口中x为78.5397491（如图4.4），返回main时, watch窗口中i为78（如图4.5），此时的i为x截取小数部分的取值，因为i的数据类型为int，而x的数据类型为float。所以i=x可以做到截取小数部分的功能。

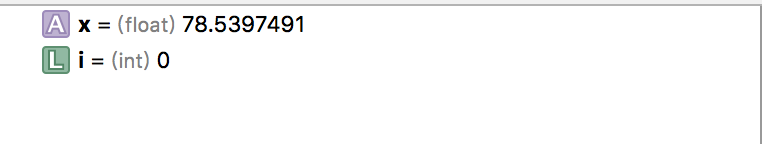


图4.4进入副函数时的x、i取值



图4.5返回主函数时的x、i取值

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**：定义带参数的宏只需将公式套入即可。判断三角形是否存在时需要判断是否满足任意两边之和大于第三边的条件，若不满足则输出error，满足则输出需要的值。本程序需要注意数据的范围，最好使用double。

#include <stdio.h>

#include <math.h>

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

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

int main ()

{

double a,b,c;

printf("请输入三角形的三边：");

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

if (((a+b)>c)&&((b+c)>a)&&((a+c)>b))

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

else printf("Error!\n");

}

测试数据分为三组。第一组三边为3、4、5，第二组三边为2、2、5，第三组三边为2、2、4。第一组的输出应为area=6，s=6，其余两组应为error！程序测试如图4.6、4.7、4.8.

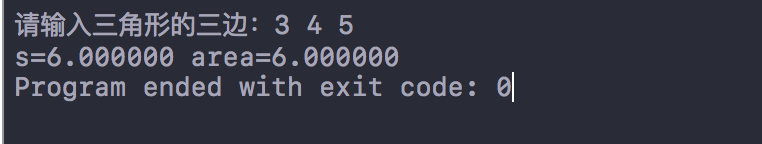


图4.6第一组数据测试

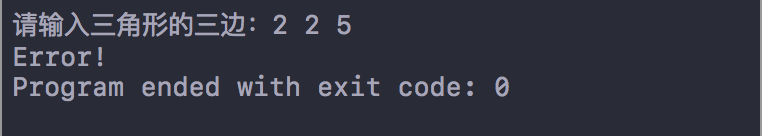


图4.7第二组数据测试

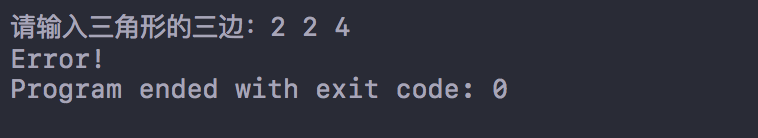


图4.8第三组数据测试

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

**解答**：通过控制宏定义的change的值进而控制输出状态，利用if语句就可以完成。

#include <stdio.h>

#define change 1

int main (){

char c;

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

if(change){

if (c>='A'&&c<='Z') c=c+32;}

printf("%c",c);

}

return 0;

}

进行两组测试，第一组为change=1的情况，第二组为change=0的情况。程序输出如图4.9、4.10.

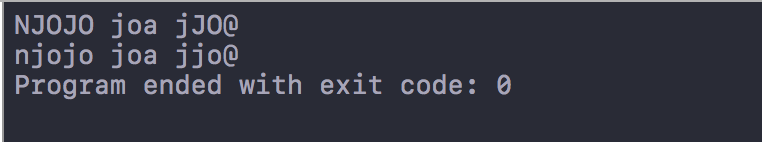


图 4.9change=1时程序测试

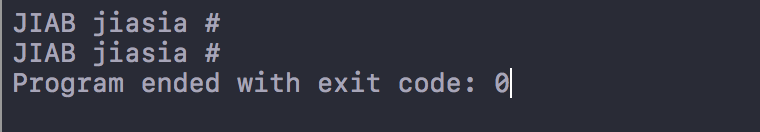


图 4.10change=0时程序测试

4.3实验心得

感觉宏定义对于一个新手来说不是很友好，虽然它具有效率高的优点，但是由于它是预处理程序并不懂c语言的规则，所以很容易产生歧义，要避免歧义必须反复加强运算顺序。但即使这样也有避免不了产生歧义的时候，所以，在我对c语言有更深的理解之前，我觉得我暂时不会大规模地使用宏定义。我觉得目前的阶段学习c语言还是比较注重准确，次要追求效率。再有就是一些比较细节的问题，比如宏名标识符后与做括号之间不能加括号，宏定义后不能加“；”,还需要多多练习。

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 实验内容及要求

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：本程序共有3个错误。第1个错误在程序的第2行，副函数必须在主函数使用前进行声明。第2个错误在程序的第7行，因为a为一位数组所以a[0]并不代表地址而代表一个值，而此处应输入地址故改为a。第3个错误在程序的第19行，三条语句之间应用“；”区别开以表示按顺序执行语句，而不是用“，”。源程序修改后如下所示。

#include<stdio.h>

void sort(int [],int);//

int main(void)

{

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

int i;

sort(a,10);//

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

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

printf("\n");

return 0;

}

void sort(int b[], int n)

{

int i, j, t;

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

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

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

{

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

}

}

程序运行结果如图5.1所示。

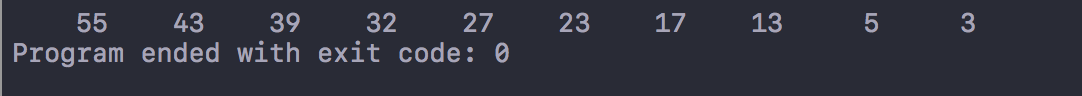


图5.1修改后源程序运行结果

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];

int i, j, k;

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

a[i] = i + 1;

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

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

if(++j > i - 1) j = 0;

b[M-i] = j? a[j-1] : a[i-1] ;

if(j)

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

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

}

for(i = 0;i < M – 1; i++)

printf(“%6d”, b[i]);

printf(“%6d\n”, a[0]);

return 0;

}

解答：按照注解的意思应该要将原数组中对应的数字编号的数剔除，再将剩余的数据重新组成数组然后再进行剔除，直到剩下最后一个编号。

a[j-1]

a[i-1]

a[k]=a[k+1]

(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：#include <stdio.h>

int main()

{

int b[11];

int i,j,k;

for (i=1;i<=10;i++) b[i]=1;

for (i=10,j=0;i>0;i--)

{

k=0;

while (k<3)

{

j++;

if (j>10) j=1;

if (b[j]==1) k++;

}

b[j]=0;

printf("%6d",j);

}

return 0;

}

程序运行结果如图5.2所示。

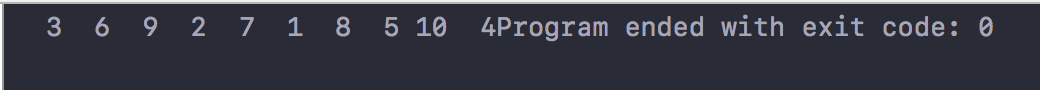


图5.2简化算法后的程序运行结果

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：

1#include<stdio.h>

2void strncat(char [],char [],int);

3int main(void)

4{

5 char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";

6 strncat(a, b, 4);

7printf("%s\n",a);

8 return 0;

9}

10void strncat(char s[],char t[], int n)

11{

12 int i = 0, j;

13while(s[i++]) ;

14for(j = 0; j < n && t[j];)

15s[i++] = t[j++];

16 s[i] = '\0';

17}

解答：（1）当光标落在for语句时，i=23如图5.3。当光条落在strncat函数块结束标记（右花括号 ））所在行时s=“The adopted symbol is” t=”abcdefghijkmnopqrstuvwxyz”如图5.4.

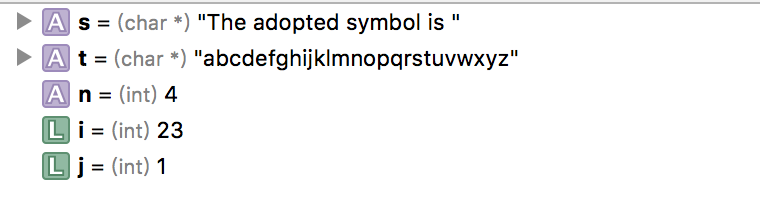


图5.3当光标落在for语句时i的值

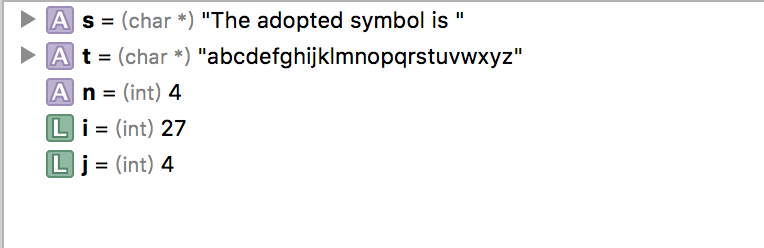


图5.4当光条落在strncat函数块结束标记时s,t的值

(2)第13行代码的原意应该为将字符数组a跳转到最后一位‘/0’,但是代码不对，应该为while(s[i])i++；才能达成原意。

5.2.4 程序设计

1.编写并上机调试运行能实现以下功能的程序。

编写一个程序,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**：这题只需要交换i，j的值，也就是交换数组两个下标的值以实现交换矩阵行列的行列，从而求得其转置矩阵。其中注意数组的起始数字是0而不是1.

#include <stdio.h>

int main()

{

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

int i,j;

for (i=0;i<=2;i++)

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

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

for (i=0;i<=2;i++)

{

for (j=0;j<=3;j++) printf("%4d",a[i][j]);

printf("\n");

}

for (i=0;i<=2;i++)

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

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

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

{

for (i=0;i<=2;i++) printf("%4d",b[j][i]);

printf("\n");

}

return 0;

}

程序测试输入矩阵1234／5678/9100，为一个3x4的矩阵，输出应该为159／261/370/480的一个4x3的矩阵，结果如图5.5所示

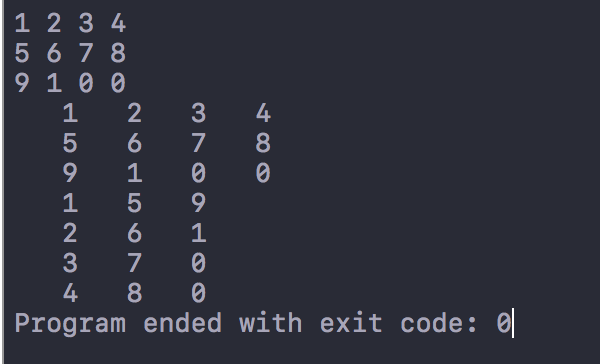


图5.5程序测试结果

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**：📄整数在计算机中使用二进制保存，故要实现此程序效果需进行位运算，将位运算的结果保存到数组中，此为过程一。然后将数组中的值按逆序（内存的保存方法与我们平常的阅读顺序相反）

#include <stdio.h>

#define s 20

int main()

{

int i,a[s];

unsigned n;

printf("Please input an unsigned integer:\n");

scanf("%u",&n);

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

a[i]=n&0x0001;

n>>=1;

}

for(i=s-1;i>=0;i--)printf("%d",a[i]);

return 0;

}

输入12，1进行测试，程序测试结果如图5.5，5.6所示。

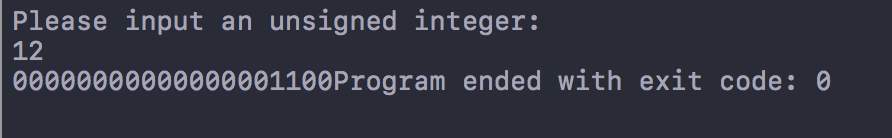


图5.5输入12时的程序测试结果

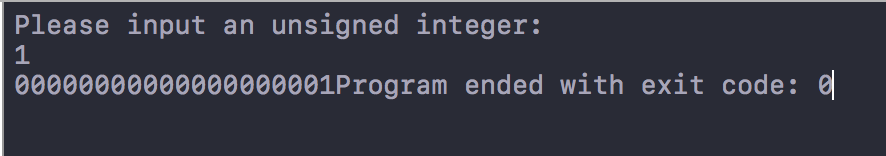


图5.6输入1 时的程序测试结果

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**：首先将学生的名字及成绩输入数组中，然后用选择排序法排列学生的成绩，然后按照学生成绩利用strcpy函数将学生名字复制到相应的名字数组中，最后利用二分查找法将对应的学生姓名及成绩输出。

#include <stdio.h>

#include <string.h>

int b[1000];

int search(int score,int n,int k);

int main()

{

int i,j,n,max,temp,score,k;

char a[1000][100],s[100];

scanf("%d",&n);

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

{

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

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

}

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

{

max=i;

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

if (b[j]>b[max]) max=j;

if (i!=max)

{

temp=b[i];

b[i]=b[max];

b[max]=temp;

strcpy(s,a[i]);

strcpy(a[i],a[max]);

strcpy(a[max],s);

}

}

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

{

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

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

printf("\n");

}

printf("\n");

scanf("%d",&score);

k=search(score,1,n);

if (k==0) printf("not found!");

else printf("%s %d",a[k],b[k]);

return 0;

}

int search(int score,int n,int k)

{

if (score==b[(n+k)/2]) return (n+k)/2;

if (n==k&&score!=b[n]) return 0;

if (score>b[(n+k)/2]) search(score,n,(n+k)/2);

else search(score,(n+k)/2+1,k);

}

按照程序要求输入2/nike 10/mike 20/20结果应为mike 20/nike 10/mike 20，程序结果截图如图5.7所示。

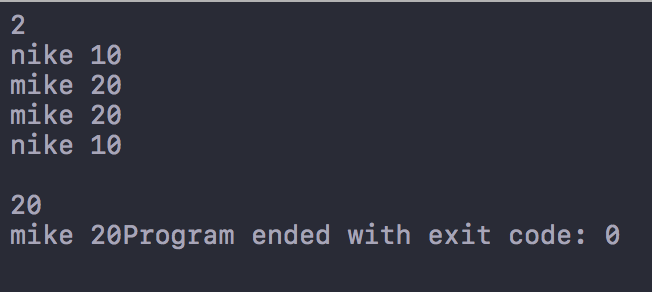


图5.7程序测试结果

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中字符串的第n个字符的后面。

**解答**：要实现题意意思，只需将字符数组s的第n个元素之后的部分用字符数组t代替。运行结果如图5.8

#include<stdio.h>

#include<string.h>

int strnins(char \*s, char \*t, int n)

{

int i, l = strlen(s);

char a[100];

if (n<0 || n>l) return 0;

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

{

a[i] = s[n + i];

}

a[i] = '\0';

for (i = 0; t[i] != '\0'; i++)

{

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

}

n += i;

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

{

s[n + i] = a[i];

}

s[n + i] = '\0';

return 1;

}

int main()

{

char t[50], s[50];

int n, i, m;

scanf("%d", &m);

getchar();

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

{

gets(s);

gets(t);

scanf("%d", &n);

getchar();

strnins(s, t, n);

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

}

return 0;

}

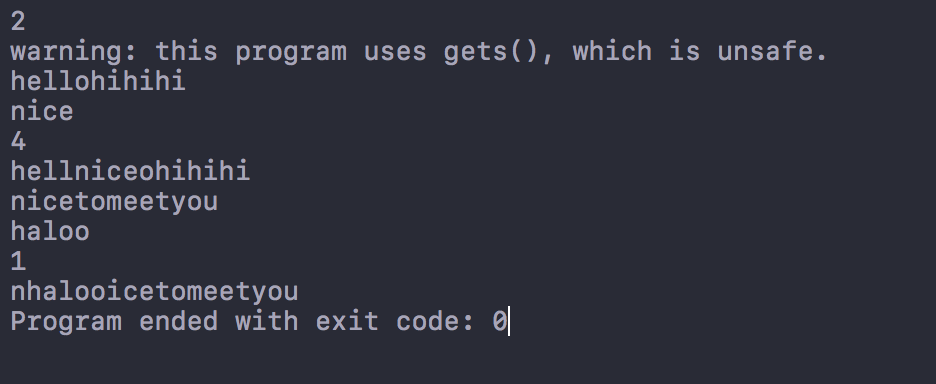


图5.8运行结果

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

**解答**：八皇后复杂比较复杂，大致分为几个模块解决问题。第一个模块是显示程序，它可以使八皇后可能的情况用坐标表示出来。第二模块是检查模块，检查其n-1行内是否能摆放成功。第三模块是试放模块，将八皇后尝试放在不同的行，用递归方法向下不停地放。运行结果如图5.9

#include <stdio.h>

#include <math.h>

#include "stdlib.h"

#define max 8

int i,a[max],sum=0;

void show(){

int i;

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

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

}

printf("\n");

sum++;

}

int check(int n){

int i;

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

if ((a[i]==a[n])||(abs(a[i]-a[n])==(n-i))) return 1;

}

return 0;

}

void put(int n){

int i;

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

{

a[n]=i;

if (!check(n))

{

if (n==max-1) show();

else put(n+1);

}

}

}

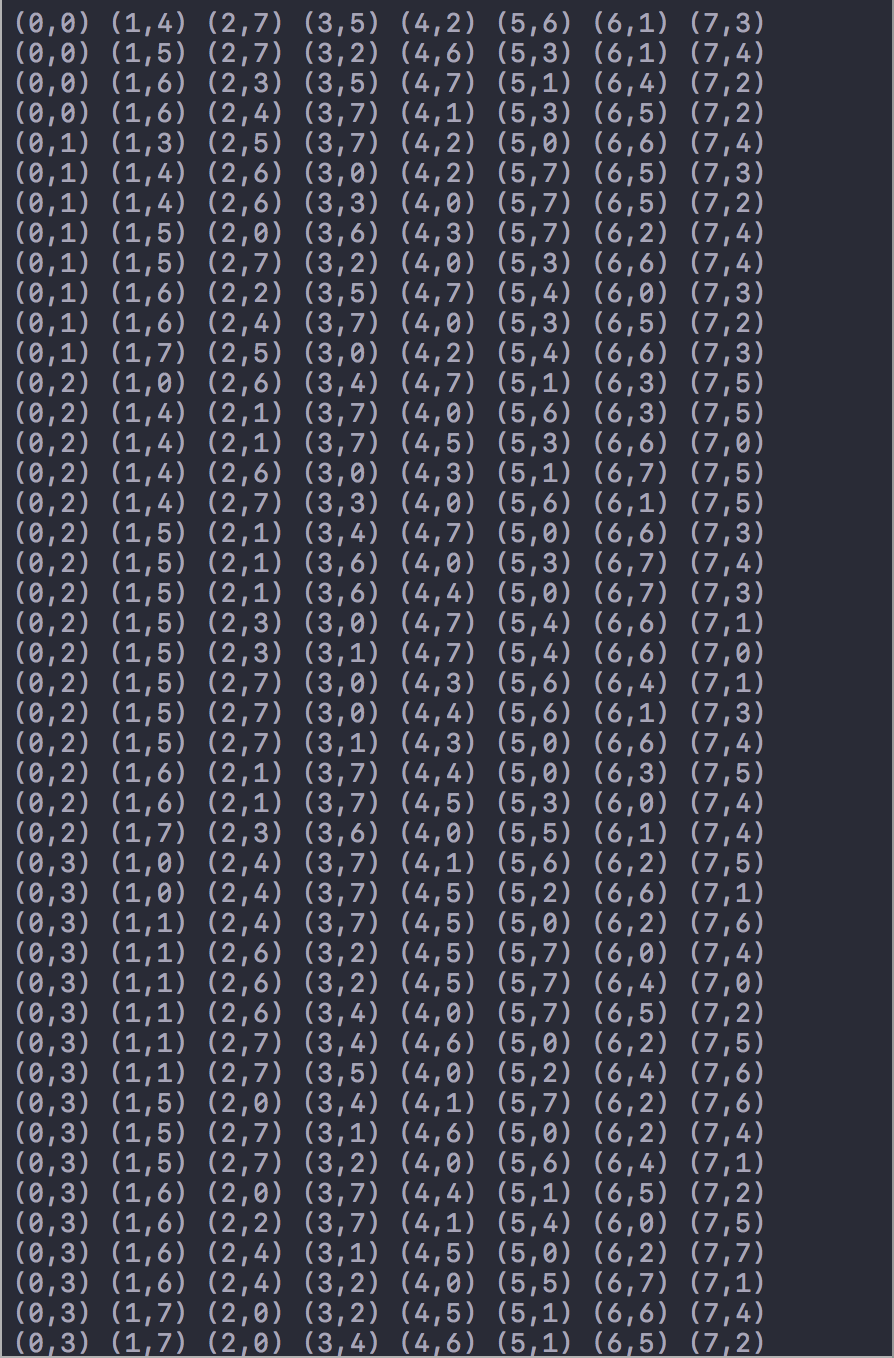
int main() {

put(0);

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

return 0;

}



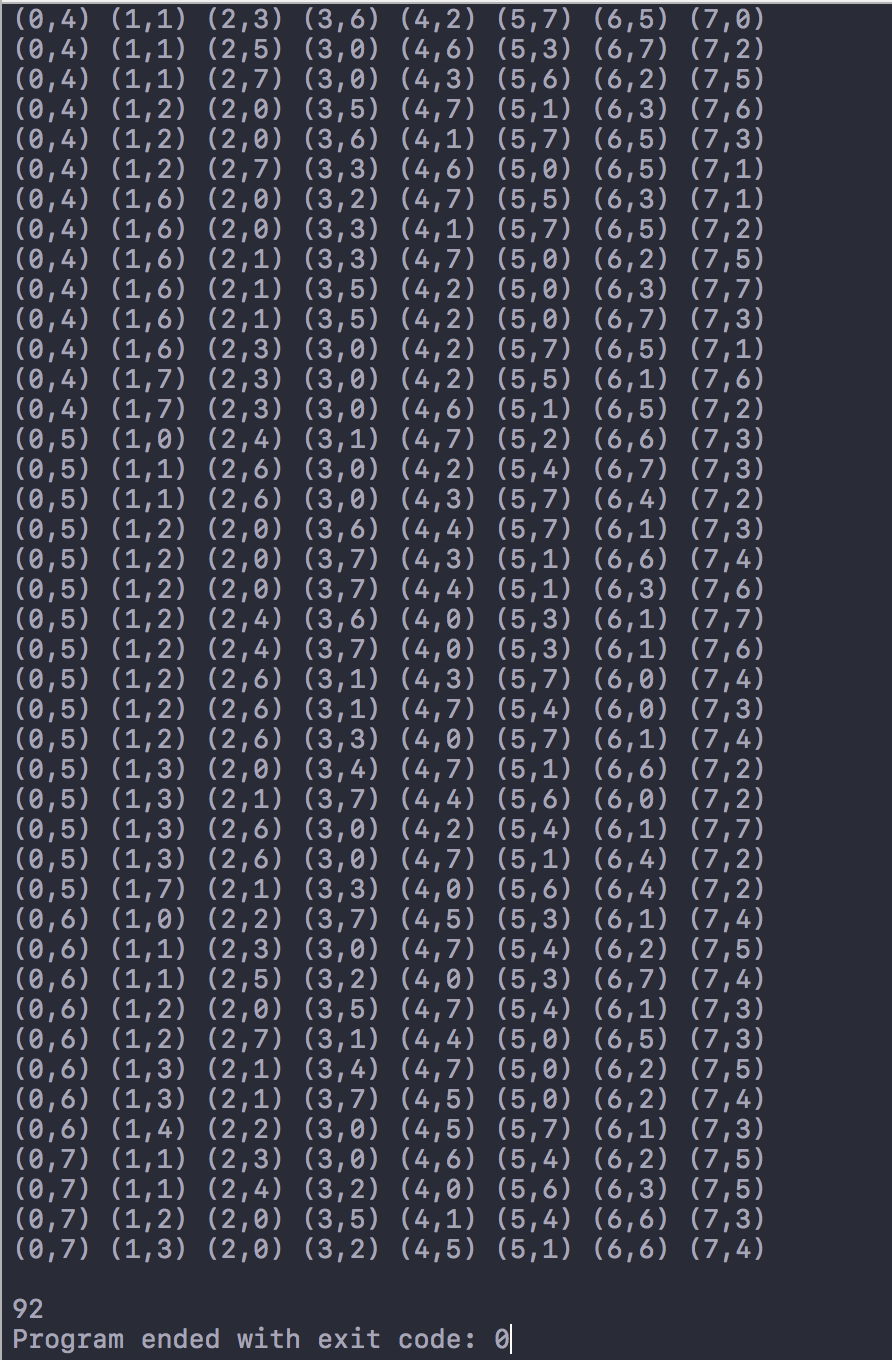


图5.9八皇后运行结果

5.3实验心得

首先感觉此次程序设计题的难度明显上升了，数组的运用比我想象的要灵活，比如最后的成绩查询。再然后就是困扰我许久的数组的地址问题，一般我们用的是&加上要指向的数组名及其所有下标，但是这样只能指向一个数组的值。如果我们想指向多维数组的一列，可以写成“数组名[n]”的形式，表示第n行的所有数组的首地址。再就是八皇后的问题，一开始以为特别复杂，后来仔细想想其实也还好，主要是判断的方法和递归这两个点一定要写对。但其实我的八皇后还有瑕疵，92种情况其实包含着一些镜像的情况，需要再讨论几种对称的情况。

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。

#include "stdio.h"

void main(void)

{

float \*p;

scanf("%f",p);

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

}

**解答**：本题创建了一个float指针却没有给出指针的指向的地址，属于指针悬浮，很容易出问题，应给出指针指向的地址。原程序改为如下。

#include "stdio.h"

void main(void)

{

float i;

float \*p=&i;

scanf("%f",p);

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

}

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)

{

1

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;

2

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

i=0;

3

result= 4 (a,b);

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

down:

;

}

**解答：**1 char \*(\*p)(char \*a,char \*b) = NULL;

2 gets(a);

3 gets(b);

4 p

（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：**#include "stdio.h"

#include "string.h"

void main(void)

{

char \*(\*p)(char \*a,char \*b) = NULL;

char a[800],b[800],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>3);

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

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

i=0;

gets(a);

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

i=0;

gets(b);

result= p(a,b);

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

down: ;

}

运行结果:输入：2

the more you learn,

the more you get

预测结果应为the more you learn, the more you get 测试截图如图6.1所示。

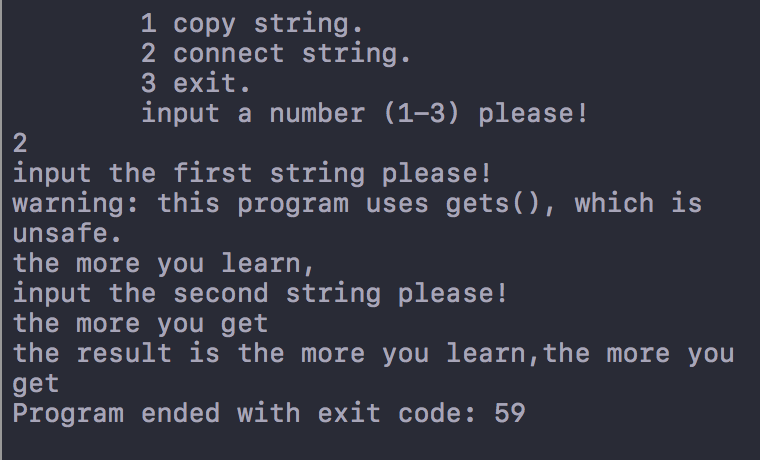


图6.1测试截图

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？

（2）排除错误，使程序输出结果为：

there is a boat on the lake.

（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？

**解答**：（1）进入strcpy时的s值如图6.2，返回main时的s值如图6.3.

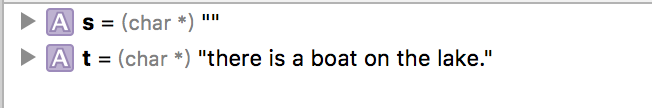


图6.2进入strcoy时的s值

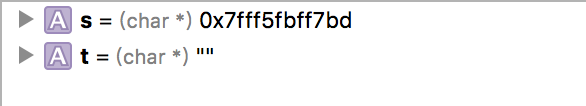


图6.3返回main时的s值

（2）原代码副函数中返回的是已经自加过的s的地址，故我们需要获取s的首地址。运行结果如图6.4所示。

#include "stdio.h"

#include "stdlib.h"

char \*strcp(char \*s,char \*t);

int main(void)

{

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

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

return 0;

}

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

{

char \*p=s;

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

return (p);

}

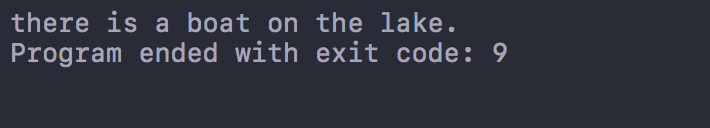


图6.4修改后的程序运行结果

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。

**解答**：将输入的数字用16进制存储，然后用逻辑尺依次取出高4位以及低4位，并且用16进制表示出来。

#include <stdio.h>

int main(void)

{

long x,k;

scanf("%ld",&x);

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

char uphalf,lowhalf;

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

lowhalf=(\*p)&0x0f;

if(lowhalf<10)

lowhalf+='0';

else

lowhalf=(lowhalf-10)+'A';

uphalf=(\*p>>4)&0x0f;

if(uphalf<10)

uphalf+='0';

else

uphalf=(uphalf-10)+'A';

p++;

printf("%c %c ",uphalf,lowhalf);

}

return 0;

}

运行结果如图6.5所示。

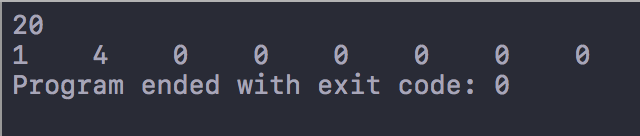


图6.5输入测试数据为20的运行结果

（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。

**解答**：利用数组指针将数组中的空格压缩，若遇到连续两个空格则直接跳过，遇到一个空格再输出一个空格，相当于用新的数组覆盖旧数组。运行结果如图6.6

#include <stdio.h>

#define N 80

void change (char \*p[],int n);

int main(void)

{

int i,n;

printf( "input the number(s) of the line(s):\n" );

scanf("%d",&n);

n=n+1;

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

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

gets(a[i]);

p[i]=a[i];

}

change(p,n);

return 0;

}

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

{

int i,j,k=0;

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

{

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

{

if (p[i][j]==' ')

{

if (p[i][j+1]!=' ')p[i][k++]=' ';

}

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

}

p[i][k]=0;

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

}

}

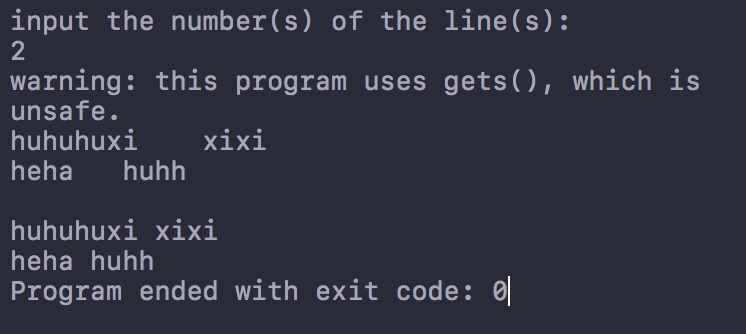


图6.6运行结果

（3）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。）

解答：将学生成绩以及各门科目定义为一个二维数组，然后通过数组指针将其带入分别编写的各个函数中，通过switch语句选择所需的操作。运行结果如图6.7

#include <stdio.h>

#define N 2

#define M 3

int \* average\_students (int \*pa,int \*sa)

{

int k,s;

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

{

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

\*(sa+k)+=\*(pa+s\*N+k);

\*(sa+k)=\*(sa+k)/M;

}

return sa;

}

int \* average\_courses (int \*pa,int \*sa)

{

int k,s;

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

{

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

\*(sa+k)+=\*(pa+k\*N+s);

\*(sa+k)=\*(sa+k)/N;

}

return sa;

}

int number\_notpass (int\*pa)

{

int c[M]={0,0,0},k,s,n=0;

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

{

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

c[k]+=\*(pa+k\*N+s);

c[k]=c[k]/10;

}

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

{

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

{

while (c[k]>\*(pa+k\*N+s)) n++;

}

}

return n;

}

int \* number\_notpass\_course(int \*pa,int \*sa){

int k,s;

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

{

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

{

if (60>\*(pa+k\*N+s)) (\*sa)++;

if (90<\*(pa+k\*N+s)) (\*(sa+1))++;

}

}

return sa;

}

int main(void){

int a[M][N]={0,0,0,0,0,0},i,j,\*pa,\*sa,\*la,\*fa,

c[M]={0,0,0},d[N]={0,0},f[2]={0,0};

char x;

pa=&a[0][0];

sa=&c[0];

la=&d[0];

fa=&f[0];

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

{

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

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

}

printf("please input the operation you want\n");

scanf("%c",&x);

scanf("%c",&x);

switch (x){

case 'a':{average\_students(pa,la);

for(i=0;i<N;i++)printf("%d\n",\*(la+i));}break;

case 'b':{average\_courses(pa,sa);

for(i=0;i<M;i++)printf("%d\n",\*(sa+i));}break;

case 'c':printf("%d\n",number\_notpass(pa));break;

case 'd':{number\_notpass\_course(pa,fa);

for(i=0;i<2;i++)printf("%d\n",\*(fa+i));}break;

}

return 0;

}

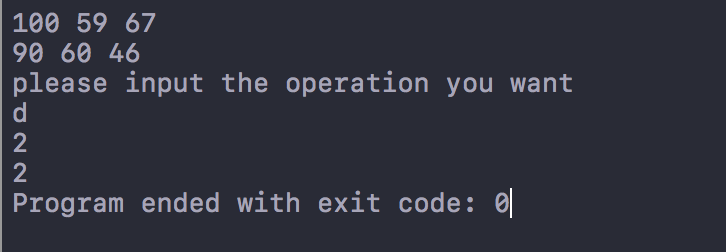


图6.7当选择d项时的运行结果

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。

**解答**：将超长小数以及整数利用字符数组保存，并利用分冶的思想将超长位数分为许多个简单的个位算法。运行结果如图6.11所示

#include <stdio.h>

#define n 20

#define m 10

int main(void) {

int i,d[m+n],e[m+n];

char k,fake,a[m+n],b[m+n];

int c[m+n];

char\*p=a;

char\*q=b;

int\*g=d,\*h=e;

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

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

k=getchar();

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

scanf("%c",p+n+i);

fake=getchar();

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

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

k=getchar();

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

scanf("%c",q+n+i);

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

{

\*(g+i)=(\*(p+i)-'0');

\*(h+i)=(\*(q+i)-'0');

if ((\*(g+i)+\*(h+i))>=10)

{

\*(c+i)=(\*(g+i)+\*(h+i))%10;

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

}

else

\*(c+i)=\*(g+i)+\*(h+i);

}

for (i=0; i<n; i++) printf("%d",\*(c+i));

printf(".");

for (i=0; i<m; i++) printf("%d",\*(c+n+i));

printf(“\n”);

return 0;

}

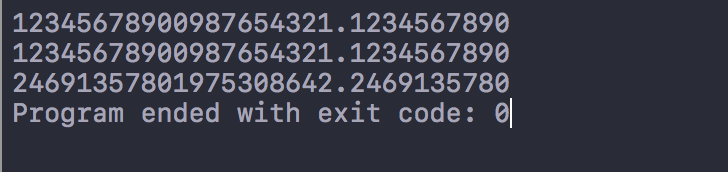


图6.11超长加法运行结果

（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。

**解答**：复杂申明为返回值为指针的指针数组函数，可以利用其进行函数名（p）的替换。程序运行结果如图6.12所示。

#include<stdio.h>

#include<string.h>

int main(void)

{

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

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

p[1]=strcpy;

p[0]=strcat;

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

gets(a);

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

gets(b);

result = p[0](a, b);

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

result = p[1](a, b);

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

return 0;

}

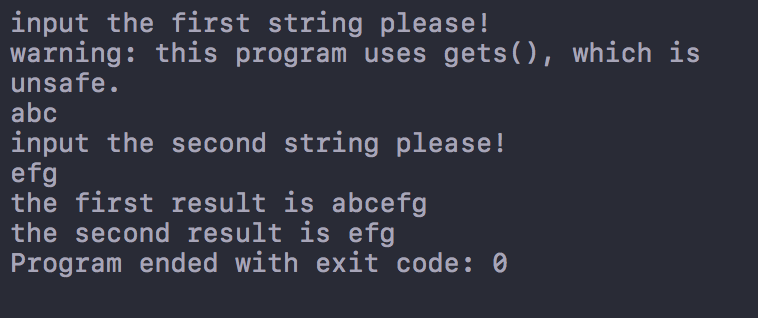


图6.12复杂声明的运行结果

6.3 实验心得

指针是c语言中一个比较重要的知识点，它可以与许多其他模块的知识联合，所以学好指针我觉得是十分有必要的。指针看似困难，但我觉得经过一定的训练之后我已经逐渐掌握了指针使用的要点，同时对于之前的数组等内容也理解得更加深刻了。用好指针也可以使程序更加简洁清晰。现阶段我觉得自己对指针的复杂声明还不够熟练，希望在以后的训练当中自己能加强对指针的复杂声明的训练。

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实验题目及要求

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 | x | x |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | v | v |
| 6 | ++\*p->t | v | v |

7.2.2源程序修改替换题

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

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

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

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

void main(void)

{

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

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

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

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

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

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

{

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

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

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

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

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

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

tail=tail->next; /\* tail指向新创建的结点 \*/

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

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

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

}

**解答**：本题的错误在于create\_list函数中，函数声明的第一个形参应该为struct s\_list结构类型的指针，并且其中还包含着headp的指针形参。故函数声明应改为void create\_list (struct s\_list \*（\*headp）,int \*p);但此题还可以进一步精简，精简后的程序如下。运行结果如图7.1所示

#include <stdio.h>

#include <stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

struct s\_list \* create\_list(int \*p)

{

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

if(p[0]==0)

;

else{

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

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

tail=loc\_head;

while(\*p){

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

tail=tail->next;

tail->data=\*p++;

}

tail->next=NULL;

}

return loc\_head;

}

int main()

{

struct s\_list \*p;

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

p=create\_list(s);

while(p){

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

p=p->next;

}

printf("\n");

return 0;

}

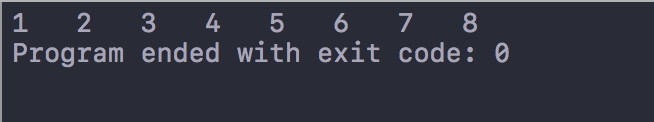


图7.1精简后的运行结果

（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：

#include <stdio.h>

#include <stdlib.h>

struct s\_list

{

int data;

struct s\_list \*next;

};

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

{

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

if(p[0]==0)

;

else

{

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

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

loc\_head->next=NULL;

while(\*p)

{

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

n->data=\*p++;

n->next=loc\_head;

loc\_head=n;

}

}

\*headp=loc\_head;

}

int main()

{

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

head=NULL;

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

create\_list(&head,s);

p=head;

while(p)

{

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

p=p->next;

}

printf("\n");

return 0;

}

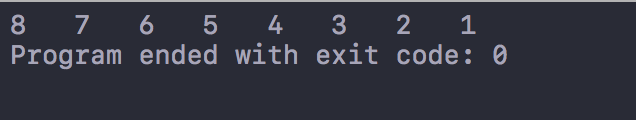


图7.2后进先出运行结果

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：

#include<stdio.h>

struct BITS

{

unsigned short bit0 : 1;

unsigned short bit1 : 1;

unsigned short bit2 : 1;

unsigned short bit3 : 1;

unsigned short bit4 : 1;

unsigned short bit5 : 1;

unsigned short bit6 : 1;

unsigned short bit7 : 1;

unsigned short resv : 8;

};

union REG

{

unsigned short all;

struct BITS bit;

};

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");

}

int main()

{

union REG uIsrReg;

unsigned short number[51];

void(\*p\_isr[8])(void);

p\_isr[0] = isr0;

p\_isr[1] = isr1;

p\_isr[2] = isr2;

p\_isr[3] = isr3;

p\_isr[4] = isr4;

p\_isr[5] = isr5;

p\_isr[6] = isr6;

p\_isr[7] = isr7;

int n, t;

scanf("%d", &n);

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

scanf("%hu", &number[t]);

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

{

uIsrReg.all = number[t];

printf("%hu:\n", uIsrReg.all);

if (uIsrReg.bit.bit0==1)

p\_isr[0]();

if (uIsrReg.bit.bit1 == 1)

p\_isr[1]();

if (uIsrReg.bit.bit2 == 1)

p\_isr[2]();

if (uIsrReg.bit.bit3 == 1)

p\_isr[3]();

if (uIsrReg.bit.bit4 == 1)

p\_isr[4]();

if (uIsrReg.bit.bit5 == 1)

p\_isr[5]();

if (uIsrReg.bit.bit6 == 1)

p\_isr[6]();

if (uIsrReg.bit.bit7 == 1)

p\_isr[7]();

printf("\n");

}

return 0;

}

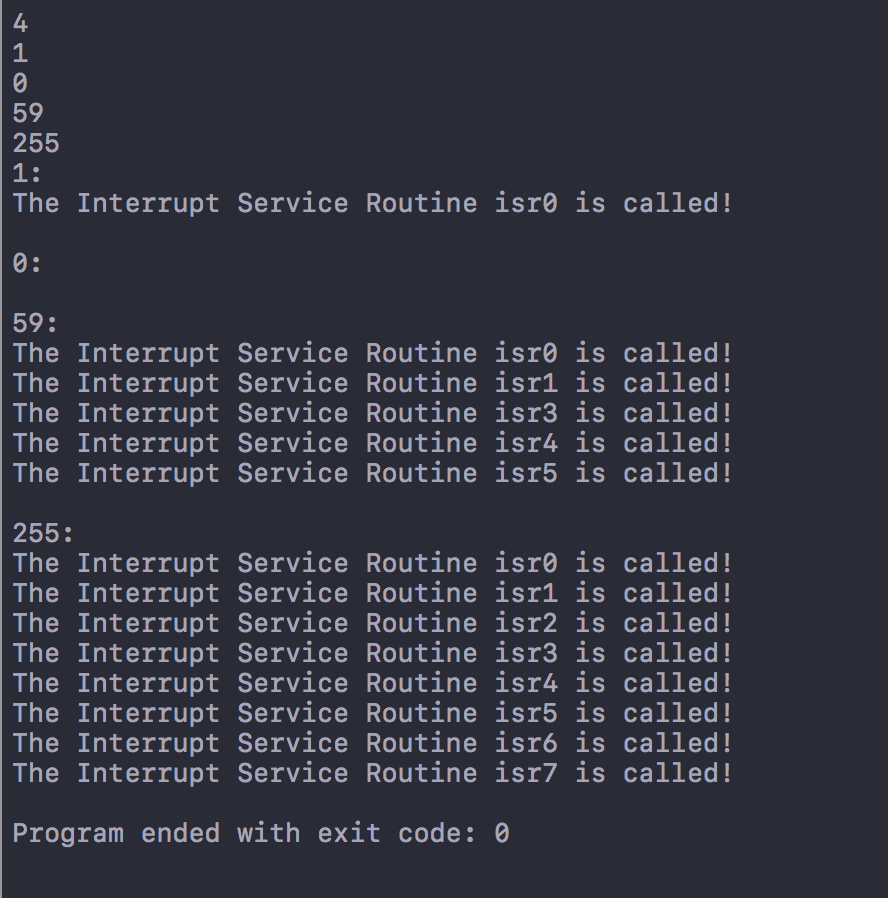


图7.3程序设计1运行结果

（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。

选修题：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**：本题与选修题第二题一起共享一个程序，选修题我选择用冒泡法进行排列。运行结果如图7.4

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct s\_list

{

char ID[15];

char Name[20];

float English;

float Math;

float Physics;

float C;

float SUM;

float AVG;

struct s\_list \*next;

};

void swap(struct s\_list \*p1, struct s\_list \*p2)

{

struct s\_list \*tmp1 = p1->next, \*tmp2 = p2->next;

struct s\_list tmp;

tmp = \*p1;

\*p1 = \*p2;

\*p2 = tmp;

p1->next = tmp1;

p2->next = tmp2;

}

void SelctSort(struct s\_list \*loc\_head)

{

struct s\_list \* p1, \*p2, \*index;

for (p1 = loc\_head; p1->next != NULL; p1 = p1->next)

{

index = p1;

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

if (p2->AVG < index->AVG)

index = p2;

if (index != p1)

swap(index, p1);

}

}

int main()

{

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

struct s\_list \*current;

struct s\_list \*p;

int i,n;

float rescore;

char reid[15],sub[15];

scanf("%d",&n);

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

scanf("%s %s %f %f %f %f",loc\_head->ID,loc\_head->Name,&loc\_head->English,&loc\_head->Math,&loc\_head->Physics,&loc\_head->C);

tail=loc\_head;

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

{

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

tail=tail->next;

scanf("%s %s %f %f %f %f",tail->ID,tail->Name,&tail->English,

&tail->Math,&tail->Physics,&tail->C);

}

tail->next=NULL;

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

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

for(current=loc\_head; current!=NULL; current=current->next)

{

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

current->ID,current->Name,current->English,current->Math,current->Physics,current->C);

}

printf("\n");

scanf("%d",&n);

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

{

scanf("%s %s %f",reid,sub,&rescore);

p=loc\_head;

for(;;)

{

if(strcmp(p->ID,reid)==0)

{

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

p->English=rescore;

else if(strcmp(sub,"Math")==0)

p->Math=rescore;

else if(strcmp(sub,"Physics")==0)

p->Physics=rescore;

else if(strcmp(sub,"C")==0)

p->C=rescore;

break;

}

else

p=p->next;

}

}

printf("Alter:\n");

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

for(current=loc\_head; current!=NULL; current=current->next)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",current->ID,current->Name,current->English,current->Math,current->Physics,current->C);

}

printf("\n");

printf("SumAndAvg:\n");

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

for(current=loc\_head; current!=NULL; current=current->next)

{

current->SUM=+current->English+current->Math+current->Physics+current->C;

current->AVG=current->SUM/4;

printf("%-15s%-20s%-10.2f%-10.2f\n",current->ID,current->Name,current->SUM,current->AVG);

}

printf("\n");

SelctSort(loc\_head);

printf("Sort:\n");

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

for(current=loc\_head; current!=NULL; current=current->next)

{

printf("%-15s%-20s%-10.2f\n",current->ID,current->Name,current->AVG);

}

printf("\n");

return 0;

}

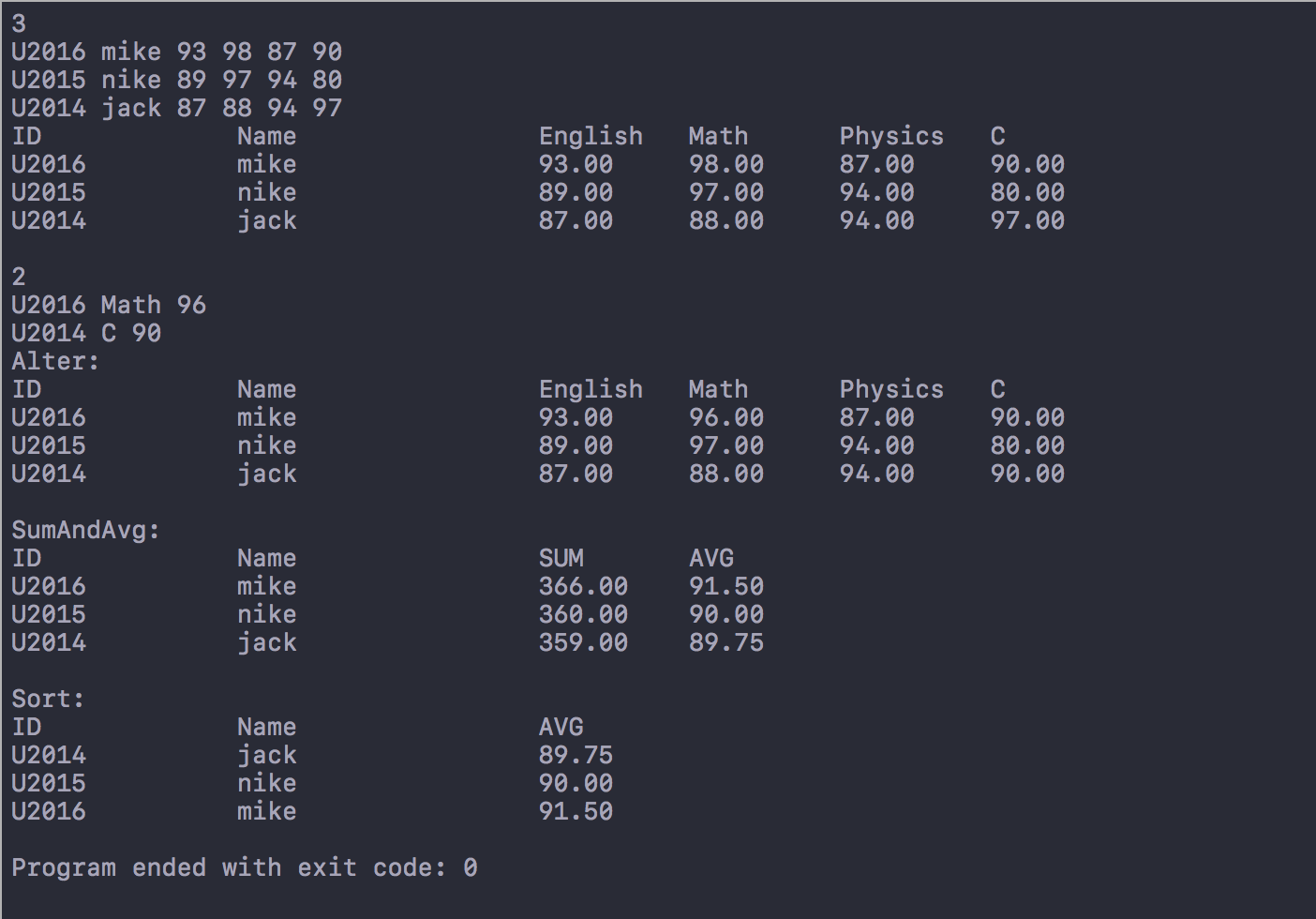


图7.4程序运行结果

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。

（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**：选做部分程序如前面题中解答所示。

7.3 实验心得

结构算是c语言高级应用的入门，以往我们需要建立一个多维数组去实现类似于成绩处理的功能，如今使用结构使得各个数据更加有机地结合，形成一个个小整体，有利于我们对其中数据进行操作。我认为学好这一部分的内容十分重要，特别是链表的实现，对于下学期的数据结构的学习想必大有裨益。做完实验之后，再回头去看自己写的程序，发现有许多冗余之处，比如没能很好地利用typedef函数，整个程序里都是struct xx，不美观也不优雅。正如俗话说：reviewing the old, can serve as a teacher. 以后也得多多回头看看以前的程序啊。

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。

答：第一行将输出以十六进制对应的25 3f 7b 7d ASCII码值对应的字符；

第二行将直接输出253f 7b7d

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

答：第一行打印的字符数量变成了2个，因为两次写入char长度的内容对应2个字符。

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

答：写入的内容变成十进制，输出的内容也变成了十进制。

8.2.2 源程序修改、替换

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

type filename

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

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

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

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

exit(-1);

}

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

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

解答：阴影部分代码（while(ch=fgetc(fp)!=EOF)）错误，应改为while ((ch = fgetc(fp)) != EOF)，修改后程序如下。

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

FILE \*fp;

if (argc != 2) {

printf("Arguments error!\n");

exit(-1);

}

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

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

exit(-1);

}

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

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

参数为D：\\abc123.txt，输出文件中的的内容，程序正确。运行结果如图8.1所示。

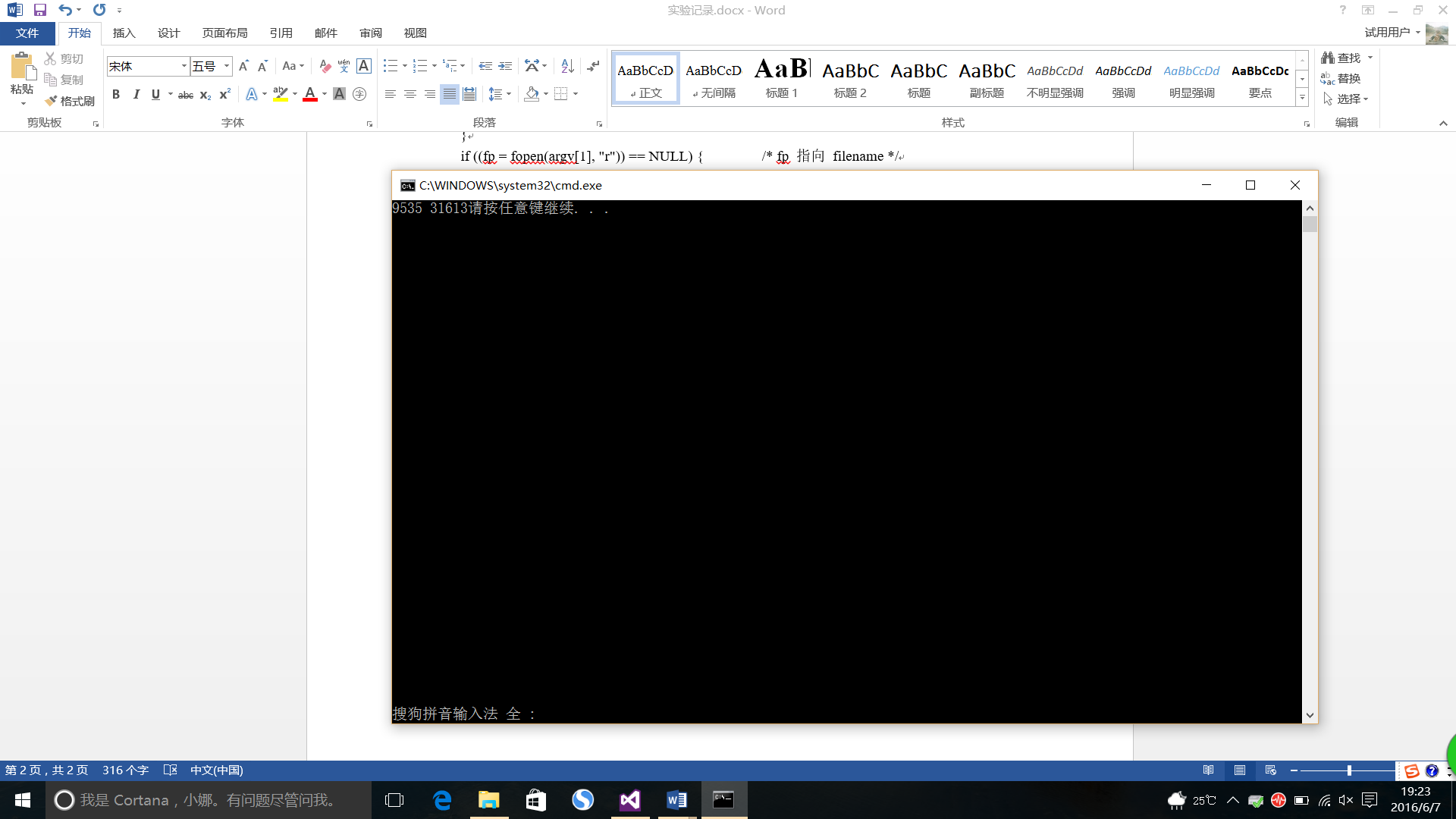


图8.1运行结果

（2）用输入输出重定向freopen改写main函数。

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

FILE \*fp;

if (argc != 2) {

printf("Arguments error!\n");

exit(-1);

}

fclose(stdin);

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

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

exit(-1);

}

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

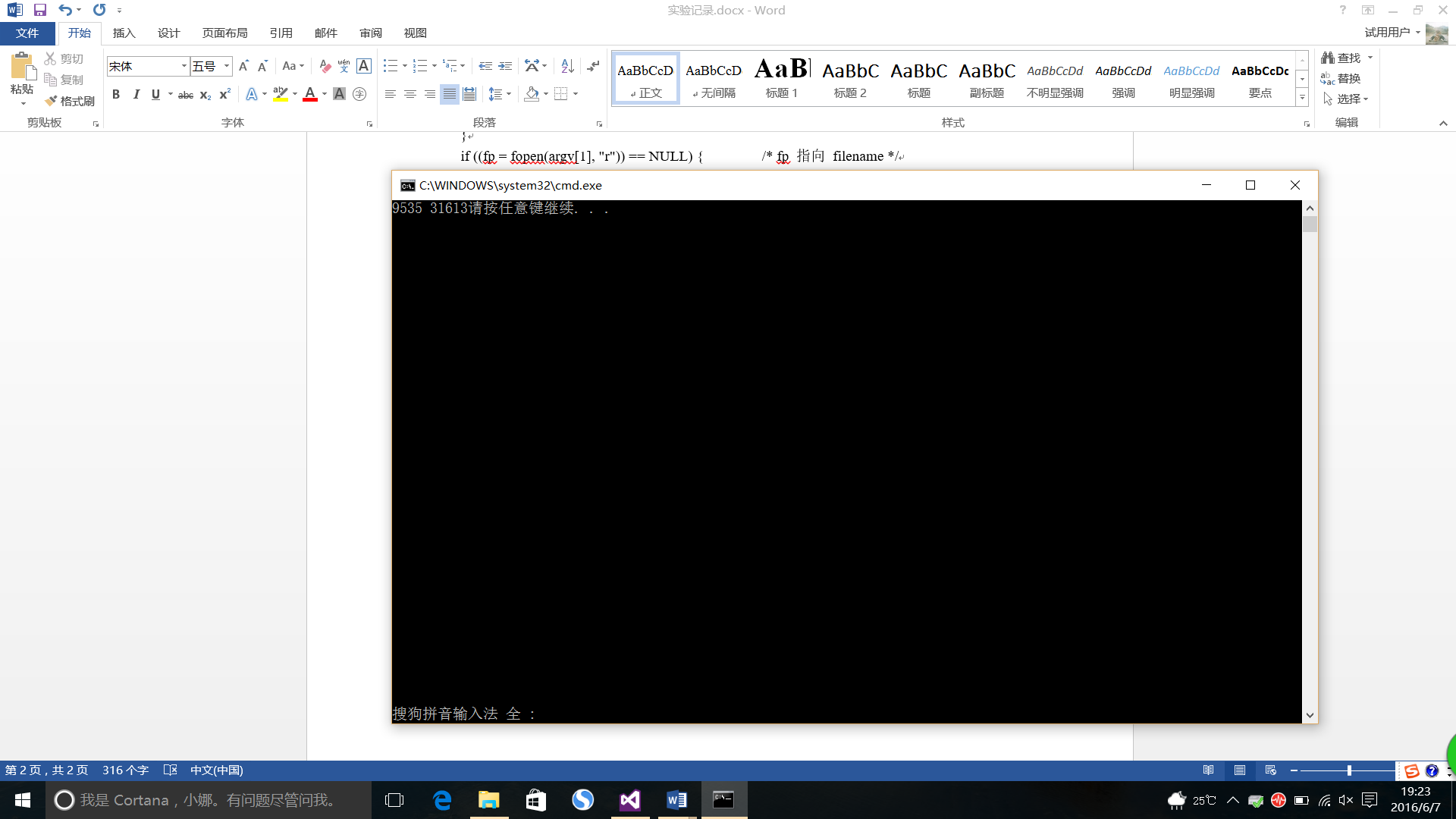
putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

参数为D：\\abc123.txt，输出文件中的的内容，程序正确。运行结果如图8.2所示。



8.2.3 程序设计

编写并上机调试运行能实现以下功能的程序或函数：

（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**：如程序框图8.3所示



图8.3程序框图

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

char c = 0;

int flag = 0;

int count = 0;

FILE\* file\_name = fopen("D:\\abc2.txt", "w");

//fputc(c, stdout);

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

{

count++;

if (c == '\n') count = 0;

if (count == 1)

{

if ('a' <= c&&c <= 'z')

{

fputc(c + 'A' - 'a', file\_name);

flag = 0;

continue;

}

else

{

fputc(c, file\_name);

continue;

}

}

if (c == ' ') flag = 1;

if ('a' <= c&&c <= 'z')

{

if (flag == 1)

{

fputc(c + 'A' - 'a', file\_name);

flag = 0;

}

else fputc(c, file\_name);

}

else fputc(c, file\_name);

}

return 0;

}

运行结果如图8.4，8.5所示。

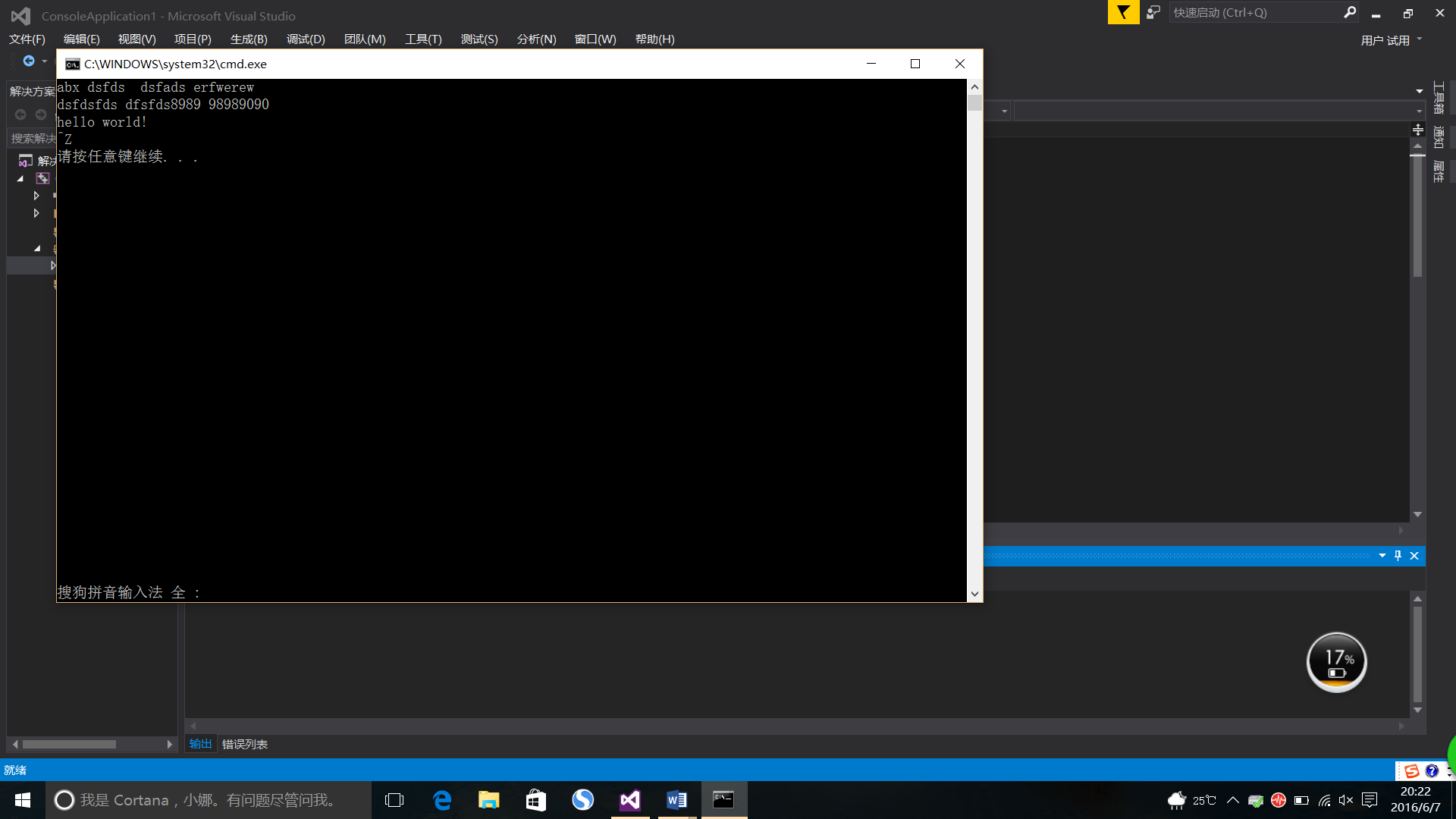


图8.4运行结果1

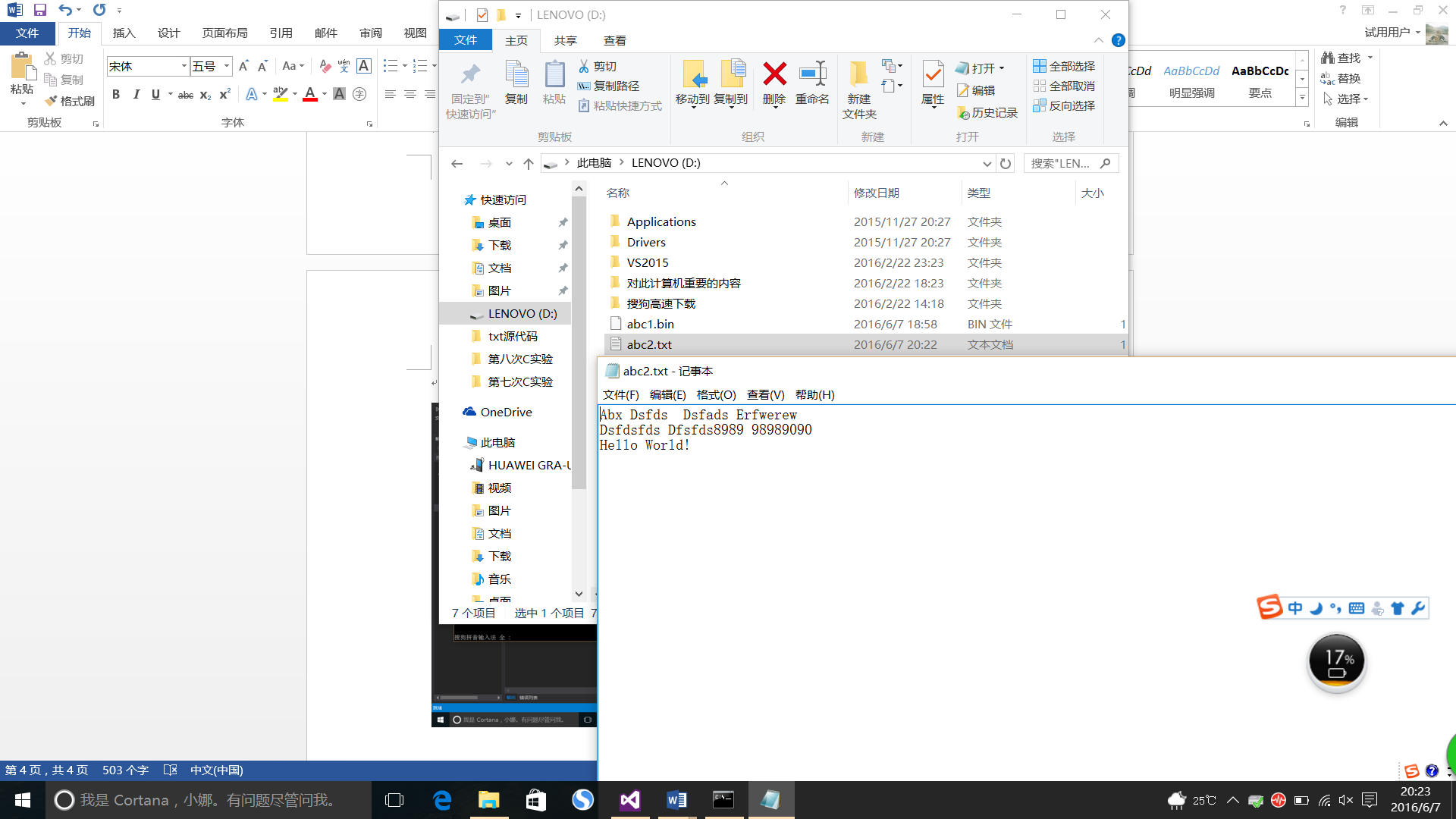


图8.5运行结果2

8.3实验心得

此次实验是在机房上完成的。文件实验相较于前面实验较为繁琐，主要是我自己没能记住那些函数名，在做实验的时候绕了不少弯子，其实想想文件实验也没有什么特别新的东西，就是把那些指令记牢了就好。

9 参考文献

无