

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： CS1801**

**学 号 U201814507**

**姓 名： 李淼**

**指导教师： 甘早斌**

**报告日期： 2019.01.02**

**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](#_Toc534315373)

[1.1实验目的 1](#_Toc534315374)

[1.2实验内容 1](#_Toc534315375)

[1.2.1源程序改错 1](#_Toc534315376)

[1.2.2源程序修改替换 3](#_Toc534315377)

[1.2.3程序设计 4](#_Toc534315378)

[1.3实验小结 10](#_Toc534315379)

[**实验2 流程控制实验** 11](#_Toc534315380)

[2.1实验目的 11](#_Toc534315381)

[2.2实验内容 11](#_Toc534315382)

[2.2.1源程序改错题 11](#_Toc534315383)

[2.2.2源程序修改替换题 12](#_Toc534315384)

[2.2.3编程设计题 15](#_Toc534315385)

[2.2.4选做题 26](#_Toc534315386)

[2.3实验小结 28](#_Toc534315387)

[**实验3 函数与程序结构实验** 29](#_Toc534315388)

[3.1实验目的 29](#_Toc534315389)

[3.2实验内容 29](#_Toc534315390)

[3.2.1源程序改错题 29](#_Toc534315391)

[3.2.2源程序修改替换题 30](#_Toc534315392)

[3.2.3跟踪调试题 32](#_Toc534315393)

[3.2.4编程设计题 35](#_Toc534315394)

[3.3实验小结 45](#_Toc534315395)

[**实验4 编译预处理实验** 46](#_Toc534315396)

[4.1实验目的 46](#_Toc534315397)

[4.2实验题目及要求 46](#_Toc534315398)

[4.2.1源程序改错题 46](#_Toc534315399)

[4.2.2源程序修改替换题 47](#_Toc534315400)

[4.2.3跟踪调试题 51](#_Toc534315401)

[4.2.4编程设计题 53](#_Toc534315402)

[4.3实验小结 58](#_Toc534315403)

[**实验5 数组实验** 59](#_Toc534315404)

[5.1实验目的 59](#_Toc534315405)

[5.2实验内容及要求 59](#_Toc534315406)

[5.2.1源程序改错 59](#_Toc534315407)

[5.2.2源程序完善、修改、替换 60](#_Toc534315408)

[5.2.3跟踪调试源程序 63](#_Toc534315409)

[5.2.4程序设计 65](#_Toc534315410)

[5.2.5选做题程序设计 71](#_Toc534315411)

[5.3 实验小结 76](#_Toc534315412)

[**实验6 指针实验** 77](#_Toc534315413)

[6.1实验目的 77](#_Toc534315414)

[6.2实验内容 77](#_Toc534315415)

[6.2.1源程序改错题 77](#_Toc534315416)

[6.2.2源程序完善、修改、替换题 78](#_Toc534315417)

[6.2.3跟踪调试题 82](#_Toc534315418)

[6.2.4编程设计题 83](#_Toc534315419)

[6.2.5选做题 92](#_Toc534315420)

[6.3实验小结 96](#_Toc534315421)

[**实验7 结构与联合实验** 98](#_Toc534315422)

[7.1实验目的 98](#_Toc534315423)

[7.2实验题目及要求 98](#_Toc534315424)

[7.2.1表达式求值的程序验证题 98](#_Toc534315425)

[7.2.2源程序修改替换题 99](#_Toc534315426)

[7.2.3编程设计题 104](#_Toc534315427)

[7.2.4选做题 114](#_Toc534315428)

[7.3实验小结 124](#_Toc534315429)

[**实验8 文件实验** 125](#_Toc534315430)

[8.1实验目的 125](#_Toc534315431)

[8.2实验题目及要求 125](#_Toc534315432)

[8.2.1文件类型的程序验证题 125](#_Toc534315433)

[8.2.2源程序修改替换题 127](#_Toc534315434)

[8.3编程设计题 128](#_Toc534315435)

[8.4实验小结 130](#_Toc534315436)

# 

# 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；

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

1 #include<stdio.h>

2 #define PI 3.14159;

3 voidmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

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

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

13 /\* for task 2 \*/

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

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

16 s = PI \* r \* r;

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

18 /\* for task 3 \*/

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

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

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

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

}

**解答：**

（1）错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.14159

2) 第3行的函数主题部分少了空格，正确形式为：

void main(void)

3) 第10行的赋值语句错误，正确格式为：

scanf(“%d”,& f ) ;

4) 第11行数字输入格式错误，正确格式为：

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

5) 第15行数字输入形式不对，正确形式为：

scanf("%lf", &r);

6）第17行的输出值不对，正确形式为：

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

7）第21行的移位不对而且输入格式不对，正确形式为：

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

（2）错误修改后运行结果：

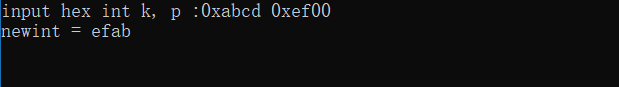
Task1:



Task2:



**Task3:**



### 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：**

可以先将a与b的和赋给a，然后减去b赋值给b，再将a减去

b赋值给a，替换后的程序如下所示：

#include<stdio.h>

void main( )

{

int a, b;

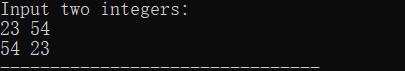
printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

a=a+b；b=b-a；a=a-b；

prinf(“\na=%d,b=%d”,a,b);

}



### 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：**

1） 算法流程如图1.1所示。

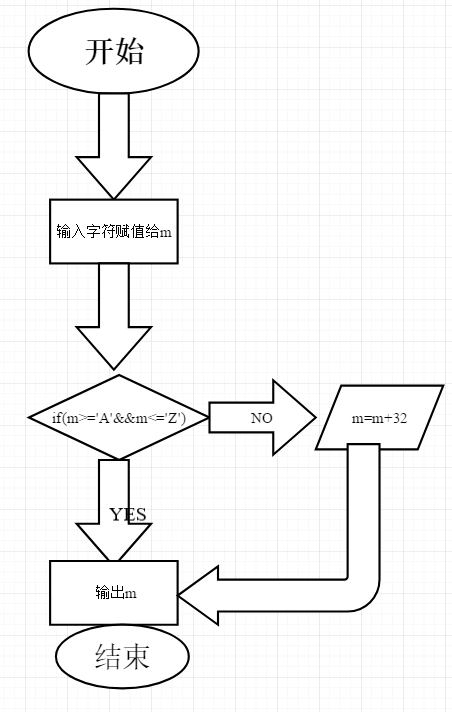


图1-1 编程题1的程序流程图

2）源程序清单

#include "stdio.h"

int main()

{

int m;

m=getchar();

if(m>='A'&&m<='Z')

{

m=m+32;

putchar(m);

}

else

putchar(m);

return 0;

}

3）测试

（a） 测试数据：

输入‘d’，输出‘d’；

输入‘N’，输出‘d’；

（b） 对应测试数据的运行结果截图





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

**解答：**

1）程序清单

#include "stdio.h"

int main()

{

unsigned short x,m,n;

printf("输入x(十六进制)，m(0~15)和n(1~16-m):\n");

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

if(m>=0 && m<=15 && n>=1 && n<=(16-m)) //判断输入是否符合题意

{

x=(x>>m)<<(16-n);

printf("%0X",x);

}

else

{

printf("输入错误！\n");

}

return 0;

}

2）解题思路

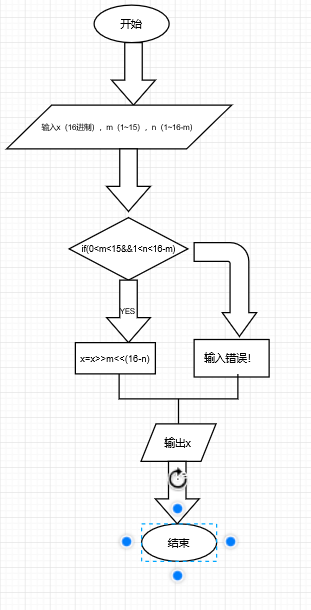


图1-2 编程题2的程序流程图

3）测试

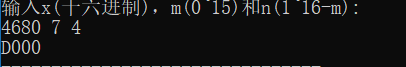
（a） 测试数据：

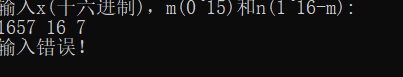
输入4680；7；4；输出：D000

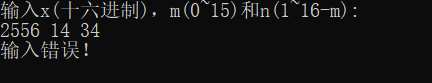
输入1657；16；4；输出：输入错误！

输入2556；14；34；输出：输入错误！

（b）运行效果截图







（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

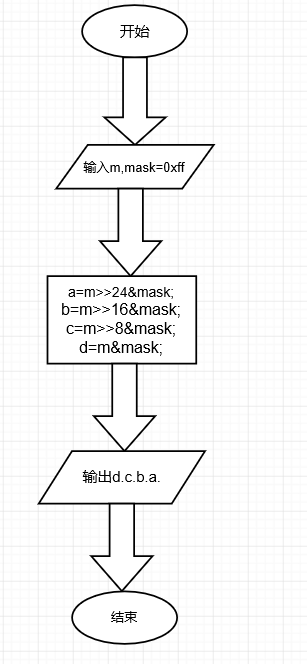


图1-3 编程题3的程序流程图

1. 编译代码

#include "stdio.h"

int main()

{

unsigned long m;

printf("Please printf your number:");

scanf("%d",&m);

unsigned mask=0xff;

int a,b,c,d;

a=m>>24&mask;

b=m>>16&mask;

c=m>>8&mask;

d=m&mask;

printf("%d.%d.%d.%d",d,c,b,a);

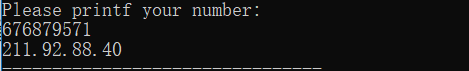
return 0;

}

1. 测试
   1. 测试数据

输入：676879571；输出：211.92.88.40；

（b）运行效果截图



## 1.3实验小结

1）本次实验为第一次实验，所以操作起来遇到了很多困难，实验课上由于对C语言的不太了解，导致没能在课堂上及时让助教检查，没能在规定时间完成任务。但是在课下我抽出了很多时间来完成作业，并且问同学，最后完成了作业。这警示我要努力学习C语言，一定要多下功夫来完成任务。

2）由于自己的打字速度很慢，所以效率就不高，在以后一定要多练习，在实践中提高自己。

3）C语言的学习告诉了我做事一定要严谨，因为你的一个标点符号的错误就会导致程序的失败，然而你却不知道自己错在哪里。

4）实验报告的书写更加让我了解了自己的程序每一步表示什莫意思，更加深入的了解了C语言，并且激发了我学习C语言 的兴趣，看着自己的实验报告，感觉好有成就感呀！

# 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);

1. }

**解答：**

（1）错误修改

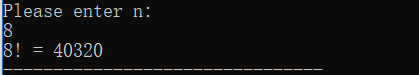
1. 第6行中的赋值语句不正确，正确形式为：

scanf("%d",&n);

1. 第7行中的for语句形式不对，正确形式为：

for(i=1,i<=n,i++)；

(2) 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。

（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。

Task1：

#include <stdio.h>

int main()

{

int i=1;

int s=1;

int n;

printf("Please enter n:");

scanf("%d",&n);

while(i<=n)

{

s=s\*i;

i++;

}

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

return 0;

}

Task2：

#include <stdio.h>

int main()

{

int i=1;

int s=1;

int n;

printf("Please enter n:");

scanf("%d",&n);

do

{

s=s\*i;

i++;

}

while(i<=n);

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

return 0;

}

Task3：

#include <stdio.h>

int main()

{

int n=1;

int s;

int m=1;

printf("Please enter s:");

scanf("%d",&s);

do

{

m=m\*n;

n++;

}

while(m<s);

printf("n=%d",n-1);

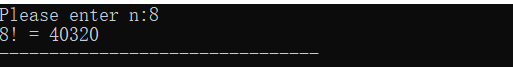
return 0;

}

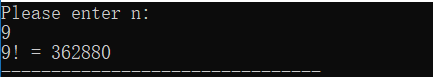
**解答：**

运行结果：

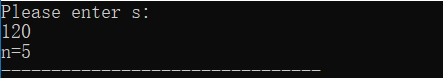
Task1：



Task2：



Task3:



### 2.2.3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. 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：**

1. 流程图如下所示：

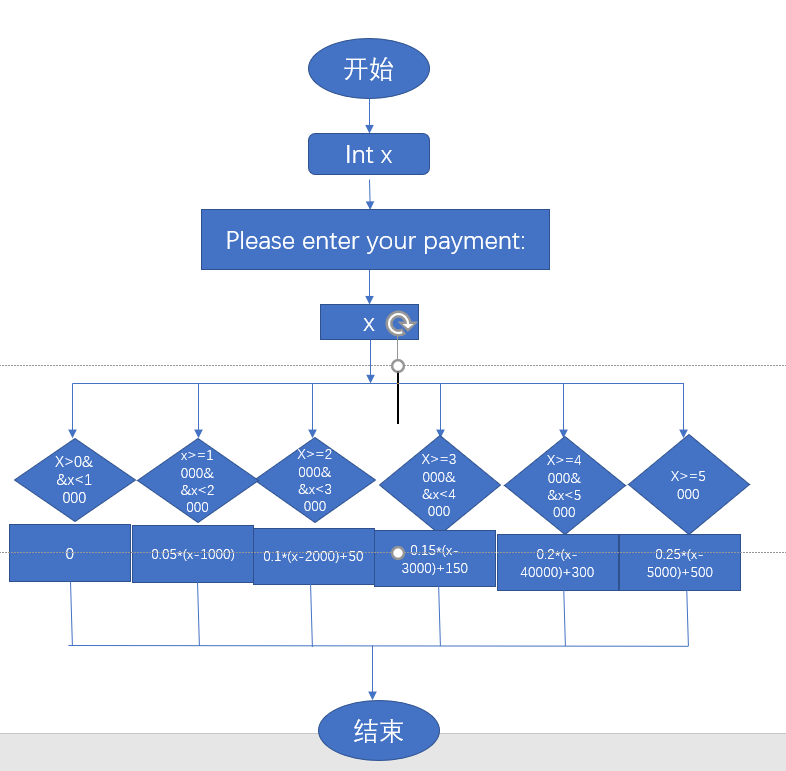


图2-1 编程题1的流程图

1. 源程序代码如下：

#include "stdio.h"

int main()

{

int x;

printf("Please enter your payment:");

scanf("%d",&x);

if(x>0&&x<1000) printf("0");

if(x>=1000&&x<2000) printf("%f",0.05\*(x-1000));

if(x>=2000&&x<3000) printf("%f",0.1\*(x-2000)+50);

if(x>=3000&&x<4000) printf("%f",0.15\*(x-3000)+150);

if(x>=4000&&x<5000) printf("%f",0.2\*(x-4000)+300);

if(x>=5000) printf("%f",0.25\*(x-5000)+500);

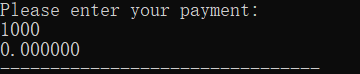
return 0;

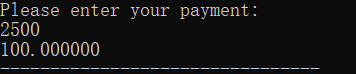
}

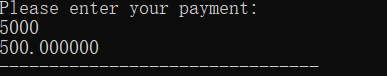
3）调试：

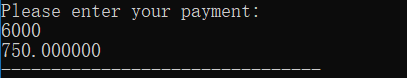
|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1000 | 0 | 0．000000 |
| 2500 | 100 | 100.000000 |
| 5000 | 500 | 500.000000 |
| 6000 | 750 | 750.000000 |

4）运行结果：









5）流程图如下：

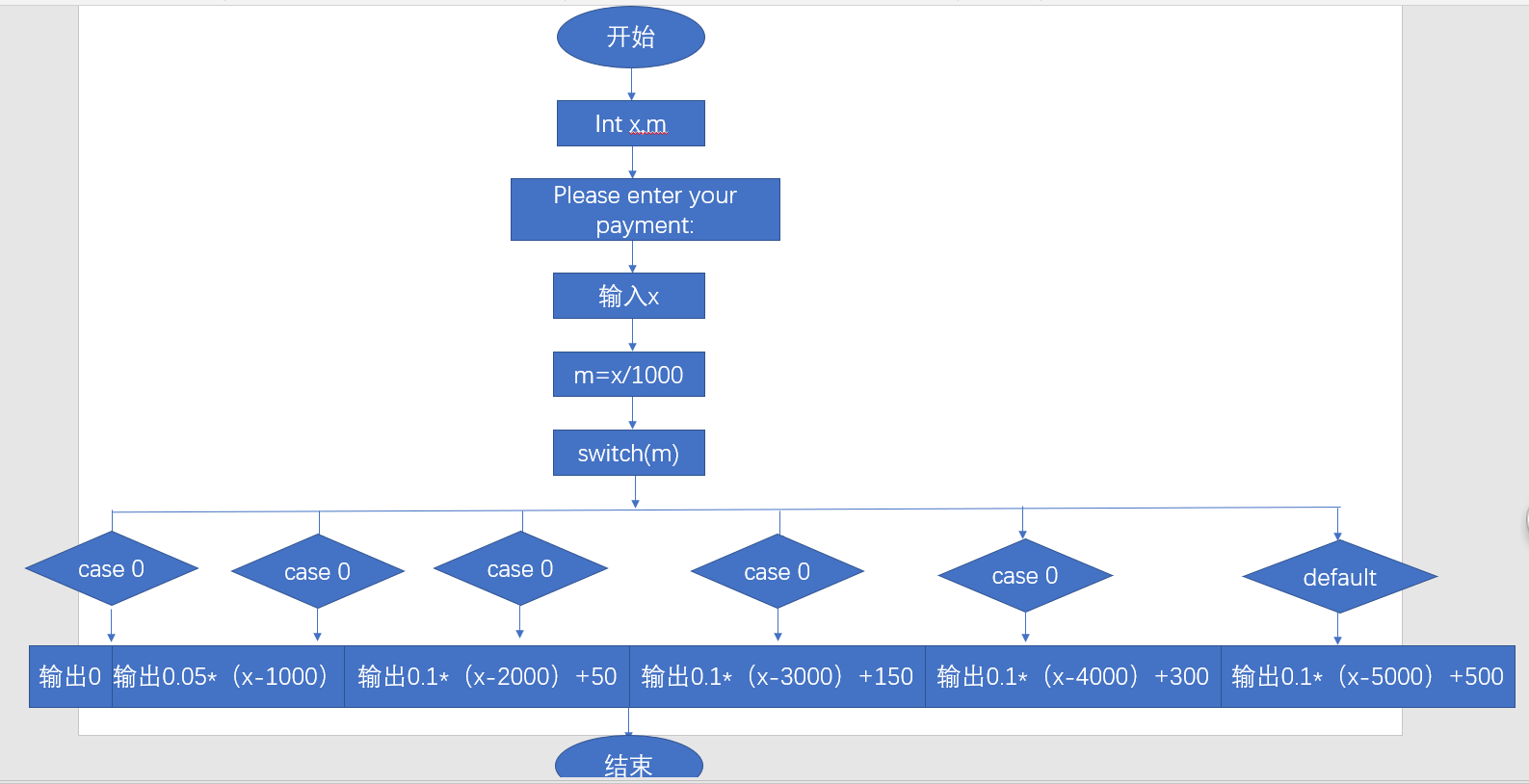


图2-2 编程题1的流程图

6）源代码如下：

#include "stdio.h"

int main()

{

int x,m;

printf("Please enter your payment:");

scanf("%d",&x);

m=x/1000;

switch(m)

{

case 0:printf("0");break;

case 1:printf("%f",0.05\*(x-1000));break;

case 2:printf("%f",0.1\*(x-2000)+50);break;

case 3:printf("%f",0.15\*(x-3000)+150);break;

case 4:printf("%f",0.2\*(x-4000)+300);break;

default:printf("%f",0.25\*(x-5000)+500);

}

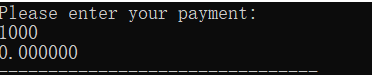
return 0;

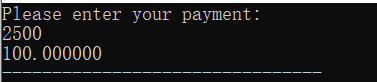
}

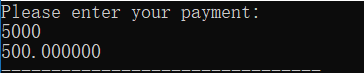
7）测试数据如下：

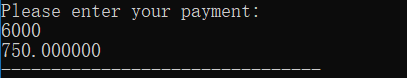
|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1000 | 0 | 0．000000 |
| 2500 | 100 | 100.000000 |
| 5000 | 500 | 500.000000 |
| 6000 | 750 | 750.000000 |

8）运行结果：









1. 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：**

1. 源程序代码如下所示：

#include "stdio.h"

int main()

{

const int a=1;

const int b=0;

int state=1;

char c;

printf("Please input words:\n");

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

{

if(state==a)

{

putchar(c);

if(c==' ')

state=b;

}

else if(state==b)

{

if(c!=' ')

{

putchar(c);

state=a;

}

}

}

return 0;

}

1. 流程图如下：

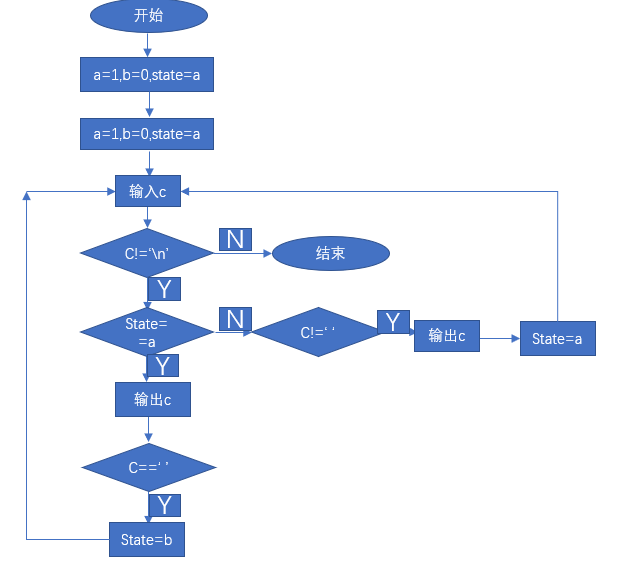
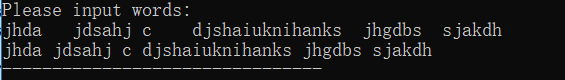


图2-3 编程题2的流程图

1. 测试如下：



（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)

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. 源程序代码如下所示：

#include "stdio.h"

int main()

{

int i,k,m,n,p,s[10][10];

for(i=0;i<=9;i++)

{

for(k=1;k<=18-2\*i;k++)

{

putchar(' ');

}

s[0][0]=1;

for(m=1;m<=9;m++)

{

s[m][0]=1;

s[m][m]=1;

}

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

{

s[i][n-1]=s[i-1][n-2]+s[i-1][n-1];

}

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

{

printf("%4d",s[i][p]);

}

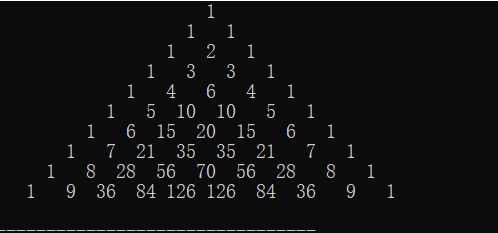
printf("\n");

}

return 0;

}

1. 输入测试：



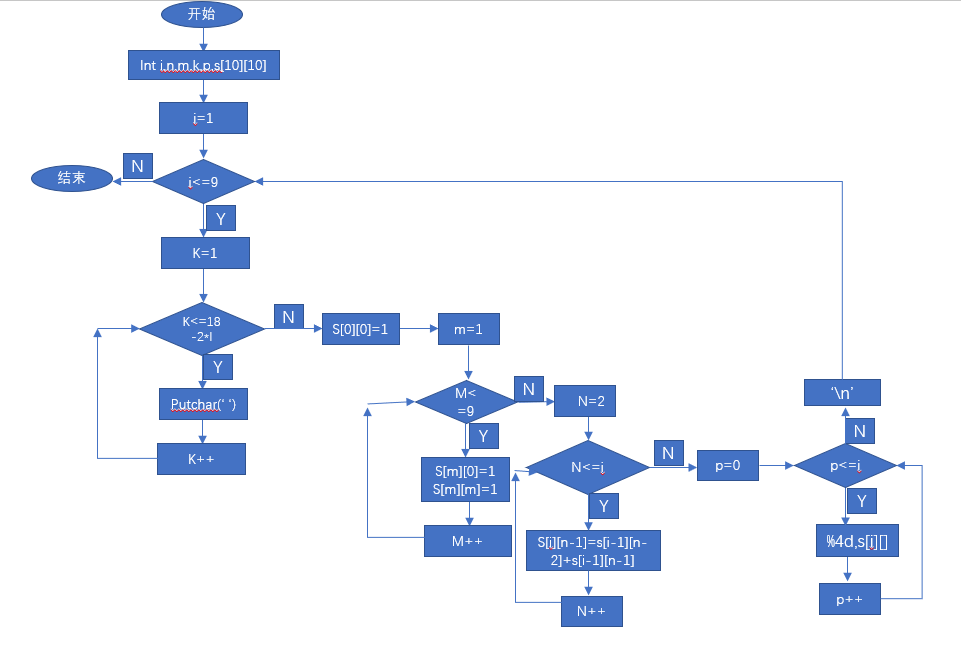
1. 流程图如下所示：  
   

图2-4 编程题3的流程图

（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：**

1. 源程序代码如下所示：

#include "stdio.h"

int main()

{

int o, i,m,n,s[10];

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

scanf("%d",&m);

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

{

s[i]=m%10;

n=m/10;

m=n;

}

o=i;

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

{

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

}

return 0;

}

1. 流程图如下所示：

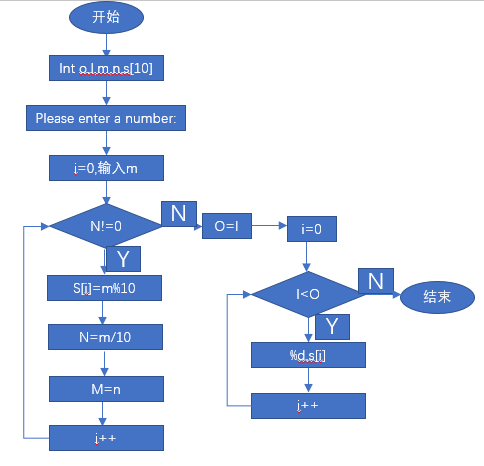
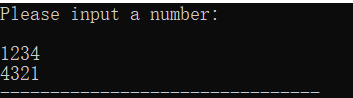
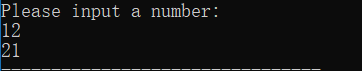


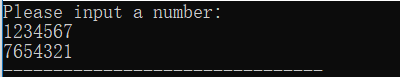
图2-5 编程题4的流程图

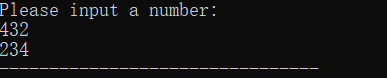
1. 数据测试如下：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1234 | 4321 | 4321 |
| 12 | 21 | 21 |
| 1234567 | 7654321 | 7654321 |
| 432 | 234 | 234 |









### 2.2.4选做题

编写并上机调试运行能实现以下功能的程序。

编写一个程序，用牛顿迭代法求方程*f(x)=*3*x3-*4*x3-5x+*13*=0*满足精度e=10-6的一个近似根，并在屏幕上输出所求近似根。

牛顿迭代法求方程近似根的迭代公式为：

，

其中, *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来作为近似根的迭代初值*x*0，然后用迭代公式计算下一个近似根*x*1。如此继续迭代计算*x*2, *x*3, …, *x*n, 直到 |*x*n- *x*n-1|≤精度e，此时值*x*n即为所求的近似根。

解答

1. 源程序代码如下所示：

#include "stdio.h"

#include "math.h"

int main()

{

const double STEP=1e-6;

double x,d;

printf("Please enter initial root:\n");

scanf("%lf",&x);

do

{

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

x=x+d;

}

while(fabs(d)>STEP);

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

return 0;

}

1. 流程图如下所示：

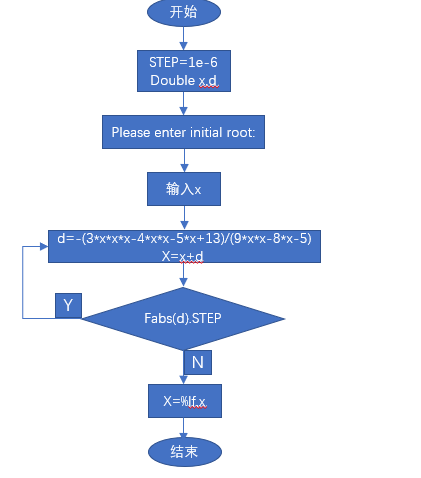
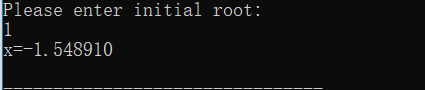


图2-6 选做题的流程设计图

1. 数据测试：



## 2.3实验小结

1）本次实验中感觉较难的是编程题的2，3，花费了我较多的时间，由于在机房较吵，所以没有静下心来做题，导致我没有在机房完成任务，花费了课余的时间，这是教训。

2）C语言必须要有良好的数学功底，在做税收实验中，假如数学功底不够，常见的数学问题不知道，就会直接导致不会做那个题。因此，在以后，一定要给数学留足够的时间来学习，数学是一切的基础。

3）由于有了第一次的经验，第二次的实验与实验报告就顺利多了。很重要的是我学会了用powerpoint画流程图，而且找到了技巧，事半功倍。

# 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：

1）第2行中缺少自定义函数的声明，应该加上：

Long sum fac(int n);

2）第13行中缺少循环语句的花括号

（2）运行截图如下所示：

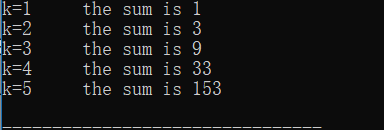


图3-1 程序修改题运行截图

### 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。

（2）修改第1题中sum\_fac函数，计算。

**解答：**

1. 第一题源程序代码如下所示：

#include "stdio.h"

long sum\_fac(int n);

int main()

{

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)

{

static long fac=1;

static long s=0;

fac\*=n;

s+=fac;

return s;

}

1. 第二题源程序代码如下所示：

#include "stdio.h"

double sum\_fac(int n);

int main()

{

int k;

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

{

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

}

return 0;

}

double sum\_fac(int n)

{

static double fac=1;

static double s=0;

fac\*=1.0/n;

s+=fac;

return s;

}

1. 第一题运行截图如下所示：

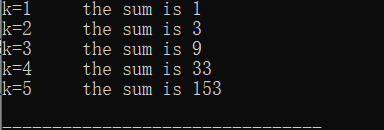


图3-2 程序修改题运行截图

1. 第二题运行截图如下所示：

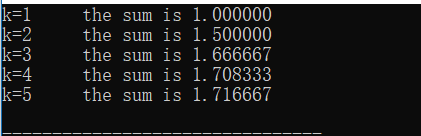


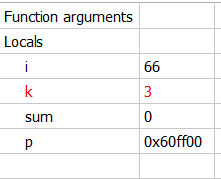
图3-3 程序修改题运行截图

### 3.2.3跟踪调试题

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

1. 刚执行完scanf("%d",&k);语句，p,i值是多少？

**解答：**此时p和i都是随机值；



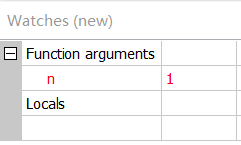
1. 从fabonacci函数返回后光条停留在哪个语句上？

**解答**：停留在调用函数的下一行，



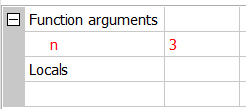
1. 进入fabonacci函数时，watch窗口显示的是什么？

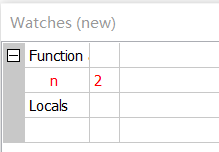
**解答：**

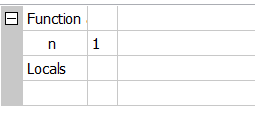


（4）当i=3时，从调用fabonacci函数到返回，n值如何变化？

**解答：**n的值从3变为2，再变为1







**源程序**

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);

}

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）。

### 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. 流程图如下所示：

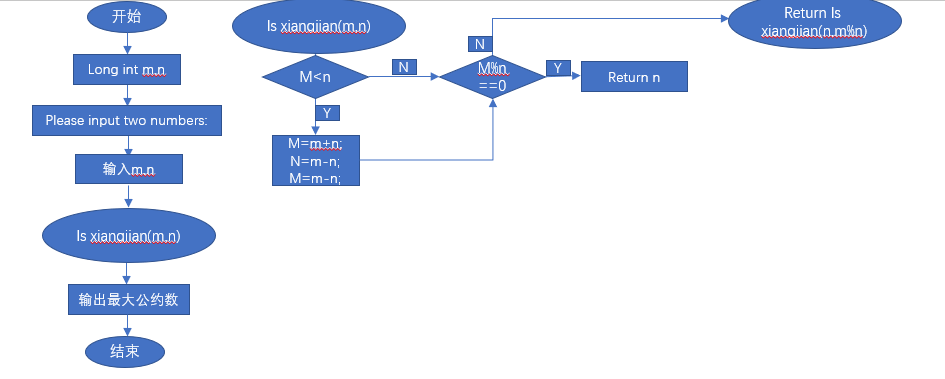


图3-4 编程题1流程图

2）源程序代码如下所示：

#include "stdio.h"

int Is\_xiangjian(int m,int n);

int main()

{

long int m,n;

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

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

printf("%d",Is\_xiangjian(m,n));

return 0;

}

int Is\_xiangjian(int m,int n)

{

if(m<n){

m=m+n;

n=m-n;

m=m-n;

}

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

Is\_xiangjian(n,m%n);

}

3）测试数据如下所示：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 18，8 | 2 | 2 |
| 10000，9888 | 2 | 2 |
| 4，10000 | 4 | 4 |

4）运行截图如下所示：

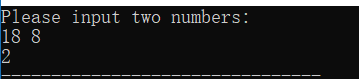


图3-5 编程题1运行截图

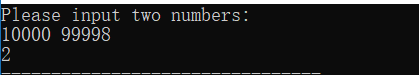


图3-6 编程题1运行截图

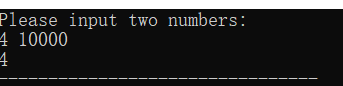


图3-7 编程题1运行截图

（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。

**解答：**

1. 流程图如下所示：

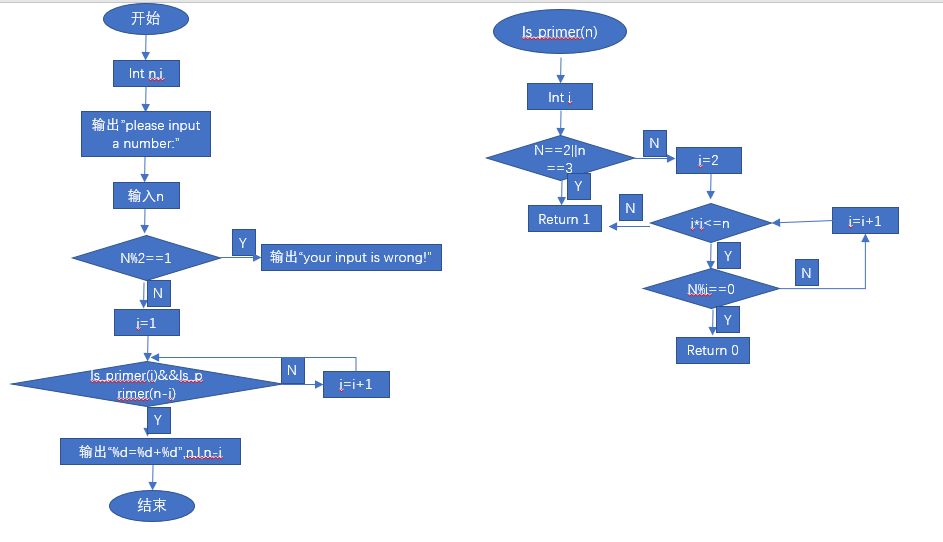


图3-8 编程题2流程图

1. 源程序代码如下所示：

#include "stdio.h"

int Is\_primer(int n);

int main()

{

int n,i;

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

scanf("%d",&n);

if(n%2==1)

printf("Your input is a fault!\n");

else

{

for(i=2;;i++)

{

if(Is\_primer(i)&&Is\_primer(n-i))

{

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

break;

}

}

}

return 0;

}

int Is\_primer(int n)

{

int i;

if(n==2||n==3) return 1;

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

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

return 1;

}

1. 数据测试如下所示：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 12 | 12=5+7 | 12=5+7 |
| 4 | 4=2+2 | 4=2+2 |
| 15 | Your input is a fault! | Your input is a fault! |

1. 运行截图如下所示：

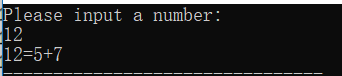


图3-9 编程题2运行截图

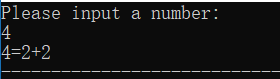


图3-10 编程题2运行截图

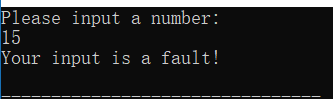


图3-11 编程题2运行截图

（3）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. 流程图如下所示：

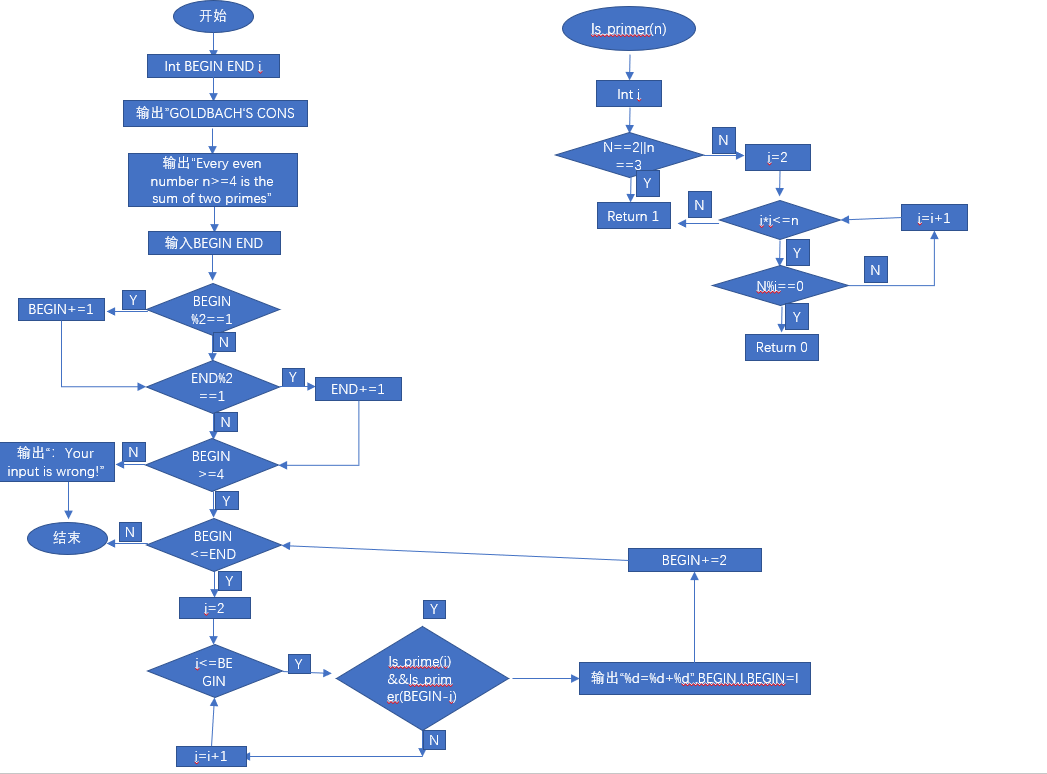


图3-12 编程题3流程图

1. 源程序代码如下所示：

#include "stdio.h"

int Is\_prime(int n);

int main()

{

int BEGIN,END,i;

printf("GOLDBACH'S CONJECTURE:\nEvery even number n>4 is the sum of two primes.\n");

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

if(BEGIN%2==1) BEGIN=BEGIN+1;

if(END%2==1) END=END-1;

if(BEGIN<4) printf("Your input is wrong!\n");

else

{

for(;BEGIN<=END;BEGIN+=2)

{

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

{

if(Is\_prime(i)&&Is\_prime(BEGIN-i))

{

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

break;

}

}

}

return 0;

}

}

int Is\_prime(int n)

{

int i;

if(n==2||n==3) return 1;

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

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

return 1;

}

1. 数据测试如下所示：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 10.，20 | 10=3+7  12=5+7  14=3+11  16=3+13  18=5+13  20=3+17 | 10=3+7  12=5+7  14=3+11  16=3+13  18=5+13  20=3+17 |
| 9，21 | 10=3+7  12=5+7  14=3+11  16=3+13  18=5+13  20=3+17 | 10=3+7  12=5+7  14=3+11  16=3+13  18=5+13  20=3+17 |

4）：运行截图如下所示：

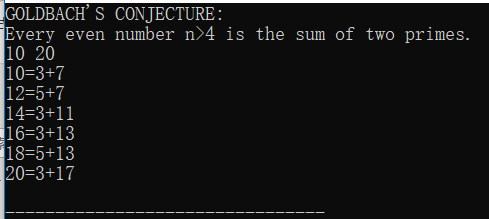


图3-13 编程题3运行截图

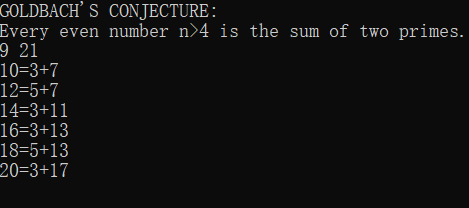


图3-14 编程题3运行截图

### 3.2.5选做题

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

源文件file1.c的内容为：

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

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

func1();

return 0;

}

源文件file2.c的内容为：

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

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

}

头文件file.h的内容为：

#include <stdio.h>

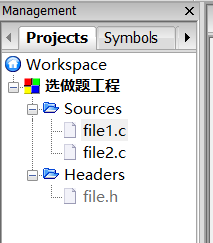
extern int x,y; /\* 外部变量的引用性说明 \*/

extern char ch; /\* 外部变量的引用性说明 \*/

void func1(void); /\* func1函数原型 \*/

**解答：**

1）建立的工程截图如下所示：



2）运行截图如下所示：

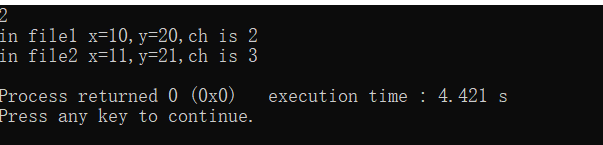


图3-14 选做题运行截图

## 3.3实验小结

1）本次实验的内容并不难，但是我花费的时间却并不少于之前的实验，总结之后，发现是因为自己对于编译器的依赖性决定的。之前习惯于使用dev编程，但是它对于初学者比较适合，对于跟踪与调试却不太方便，而我却不会用codeblocks,这就花费了我好多的时间。

2）不过要感谢三班的两位好兄弟，是他们手把手教我用codeblocks来调试，我受益匪浅，不仅学会了如何调试，而且还知道了怎末来检查自己的编程错误，这些经验是无价的。所以在此感谢两位好兄弟，我们一直都会是好兄弟的。

3）本次实验主要是函数的调用问题，我觉得易错在于调用的格式，比如说返回值，函数类型啊。而且我深刻的体会到了数学的重要性，深厚的数学功底会有很大啊的优势。

# 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

5）int 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 11) is:%d",SUM, SUM\*DIF);

12) SWAP(a,b);

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

14) }

**解答：**

1)错误修改如下所示：

1. 第2）行中的宏定义错误，正确形式应该为：#define SUM(a,b) (a+b)
2. 第3）行中的宏定义错误，正确形式应该为：#define DIF(a,b) (a-b)
3. 第4）行中的两数交换错误，正确行使应该为：#define SWAP(a,b) a=a+b,b=a-b,a=a-b
4. 第9）行中的赋值语句错误，正确形式应该为：scanf(“%d%d”,&a,&b);
5. 第10）行中的宏引用错误，正确引用形式应该为：SUM(a,b),SUM(a,b)\*DIF(a,b)

2）运行截图如下所示：

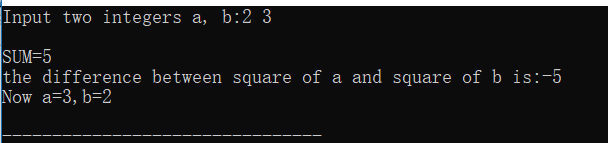


图4-1 程序修改题运行截图

### 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：**

1. 程序修改如下所示：

#include "stdio.h"

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

float sum(float x,float y);

int main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

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

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

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

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

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

return 0;

}

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;

}

运行截图如下所示：

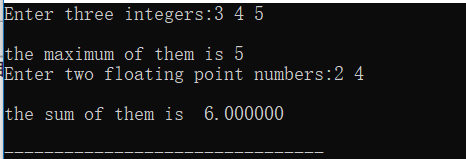


图4-2 程序替换题运行截图

1. 程序替换如下所示：

#include "stdio.h"

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

int main()

{

int a, b, c;

printf("Enter three integers:");

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

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

return 0;

}

运行截图如下所示：

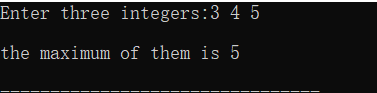


图4-3 程序替换题运行截图

### 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。现要求：

（1）修改程序，使程序编译通过且能运行；

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

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

#define R

void main(void)

{

float r, s;

int s\_integer=0;

printf ("input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

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

s\_integer= integer\_fraction(s);

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

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

#endif

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

**解答：**

1. 源程序修改如下所示：

#include "stdio.h"

#include "assert.h"

int integer\_fraction(float x);

#define R

int main(void)

{

float r, s;

int s\_integer=0;

printf ("input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

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

s\_integer= integer\_fraction(s);

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;

}

1. 运行截图如下所示：

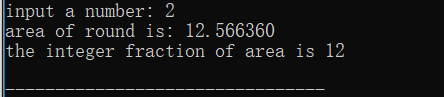
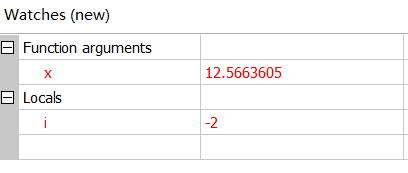
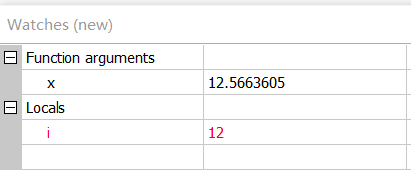


图4-4 跟踪调试题运行截图

1. 条件追踪：





进入函数时，watches窗口中的x为计算的到的s值；

返回主函数时，i的值为x的整数部分；

### 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. 源程序代码如下所示：

#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()

{

int a,b,c;

double s,area;

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

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

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

{

s=s(a,b,c);

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

}

else

printf("Your input is wrong!");

return 0;

}

1. 流程图如下所示：

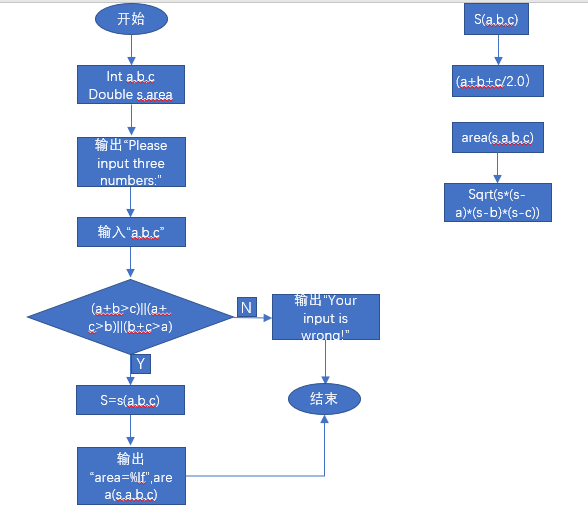


图4-5 编程题1流程图

1. 数据测试：

|  |  |  |
| --- | --- | --- |
| 输入 | 应该输出 | 实际输出 |
| 3，4，5 | 6.000000 | 6.000000 |
| 2，3，6 | Your input is wrong! | Your input is wrong! |
| 4，4，4， | 6.928203 | 6.928203 |

1. 测试截图如下所示：

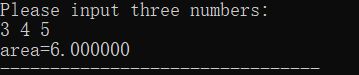


图4-6 编程题1运行截图

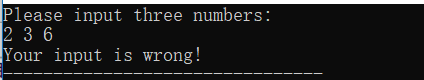


图4-7 编程题1运行截图

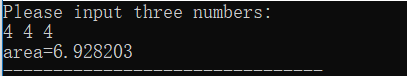


图4-8 编程题1运行截图

（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s文字，若#define CHANGE 0则原文输出。

**解答：**

1. 源程序代码如下所示：

#include "stdio.h"

#define CHANGE 1

int main()

{

char c;

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

{

if(CHANGE==1)

{

if(c>='a'&&c<='z') putchar(c-32);

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

else putchar(c);

}

else

putchar(c);

}

return 0;

}

1. 流程图如下所示：

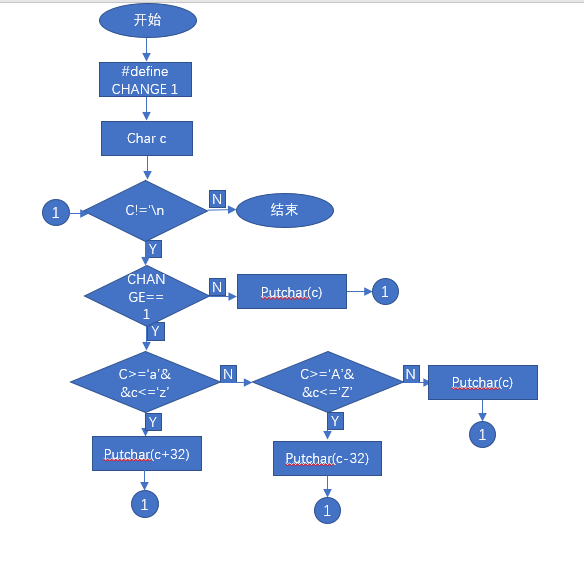


图4-9 编程题2流程图

3数据测试如下所示：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| ABCDEabcde ab | abcdeABCDE AB | abcdeABCDE AB |
| ABCDEabcde FG | ABCDEabcde FG | ABCDEabcde FG |

1. 流程图如下所示：



图4-10 编程题2运行截图

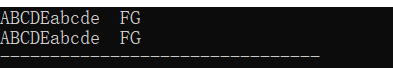


图4-11 编程题2运行截图

## 4.3实验小结

1）本次实验题目较为简单，主要考察了宏定义的操作，关键点在于定义时的一些规范操作，只要注意就可以了。

2）有了前几次的操作，发现我写实验报告的效率快乐很多，而且画出的流程图也好看了不少，好开心。

# 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：

（1）错误改正：

1) 第7行中的调用函数不正确，应该为：

sort(a,10);

2) 第18行中的判断条件不正确，应该为：

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

（2）运行截图如下所示：



图5-1 程序修改题运行截图

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

if(j)

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

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

}

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

printf(“%6d”, b[i]);

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

return 0;

}

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

#include "stdio.h"

int main()

{

int a[10],i,j,k=1;

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

a[i]=i+1;

while(k<31)

{

if(a[i%10]) k++,i++;

else

{i++;

continue;

}

if((k-1)%3==0)

{

printf("%d ",a[(i-1)%10]);

a[(i-1)%10]=0;

}

}

return 0;

}

3)实验思路：

使用两个计数器，一个负责调节程序的运行，另外一个负责计数，如果被取出，那么就记为0，计数器不自增。

1. 运行截图如下所示：



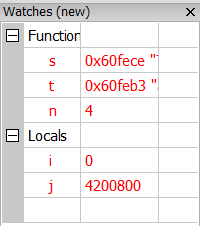
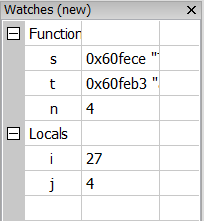
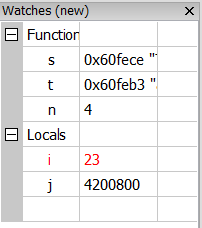
图5-2 程序替换题运行截图

### 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？

解答：

1. 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。

解答：

1. 错误原因:由于调用函数时对于变量的理解不够，所以循环语句开始时，是s字符串已经结束了，导致不能在他后边加上理想中的字符

2）改正：在while语句之后，for语句之前加上i=i-1；使加上的字符串

3）运行截图如下所示：

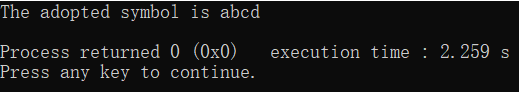


图5-3 跟踪调试题运行截图

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';

}

### 5.2.4程序设计

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. 源程序代码如下所示：

#include "stdio.h"

int main()

{

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

int i,j;

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

{

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

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

}

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

{

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

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

}

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

{

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

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

putchar('\n');

}

putchar('\n');

putchar('\n');

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

{

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

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

putchar('\n');

}

return 0;

}

1. 设计思路：

A：先定义两个数组，并且对原来的数组进行赋值。

B：然后进行循环，把已有的数组的行赋值给另一数组的列，列赋值给行，从而实现矩阵的转置。

C：接着用两个循环，分别打出转置前后的矩阵即可。

1. 运行截图如下所示：

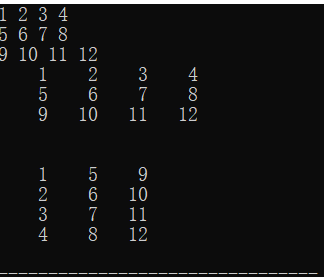


图5-4 程序设计题1运行截图

1. 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. 源程序代码如下所示：

#include "stdio.h"

int main()

{

int n,i;

char c[32];

int mask=1<<31;

scanf("%d",&n);

for(i=0;i<32;i++)

{

c[i]=n&mask?'1':'0';

n=n<<1;

}

for(i=0;i<32;i++)

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

return 0;

}

1. 实验思路：

A：本题主要采用逻辑尺的相关知识，原理是计算机中的数字都是以二进制形式存储的。

B：首先定义一个32位的字符数组，同时定义一个逻辑尺，该逻辑尺是1左移31位后得到的

C；然后分别用该逻辑尺和输入的数字进行按位与，如果结果为0，则将0这个字符储存在数组中，否则储存1在这个数组中。

D：最后用一个循环就可以将这个数组中的字符输出。

1. 运行截图如下所示：

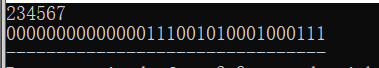


图5-5 程序设计题2运行截图

1. 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. 源程序代码如下所示：

#include "stdio.h"

#include "string.h"

int binarysearch(int s[],int k,int o);

int main()

{

int i,k,m,t,o;

scanf("%d",&k);

char n[k][20],c[20];

int s[k];

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

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

int p,q;

for(p=0;p<k-1;p++)

{

for(q=0;q<k-p-q;q++)

{

if(s[q]<s[q+1])

{

s[q]=s[q]+s[q+1];

s[q+1]=s[q]-s[q+1];

s[q]=s[q]-s[q+1];

strcpy(c,n[q]);

strcpy(n[q],n[q+1]);

strcpy(n[q+1],c);

}

}

}

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

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

putchar('\n');

scanf("%d",&m);

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

{

scanf("%d",&o);

t=binarysearch(s,k,o);

if(t==-1)

printf("Not found!\n");

else

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

}

return 0;

}

int binarysearch(int s[],int k,int o)

{

int f=0,b=k-1,m;

for(;f<=b;)

{

m=(f+b)/2;

if(o<s[m]) f=m+1;

else if(o>s[m]) b=m-1;

else return m;

}

return -1;

}

1. 实验思路：

A：本体主要应用到了函数调用，冒泡排序，多维数组（二维数组），以及二分法查找。

B：首先定义一个二维数组，用来存储姓名，定义一个一维数组，用来存储成绩，定义一个函数，用来二分法查找成绩。

C：然后用一个循环来输入姓名，注意，姓名是以字符串形式输入的，因此在存储时不用加那一个&。

D：冒泡排序时要注意循环的次数，并且要用到字符串的复制，因此要在头文件中include，并且要注意成绩交换时姓名也要同步交换，保证成绩和姓名要一一对应。

F：调用函数时要注意调用的形式要正确。

1. 运行截图如下所示;

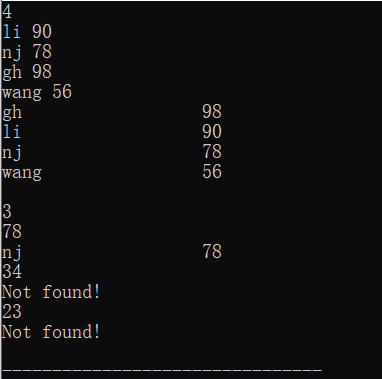


图5-6 程序设计题3运行截图

### 5.2.5选做题程序设计

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：

1. 源程序代码如下所示：

#include "stdio.h"

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

int main()

{

int s[100],t[100],n;

int i=0;

int j=0;

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

s[i-1]='\0';

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

t[j-1]='\0';

scanf("%d",&n);

strnins(s,t,n);

return 0;

}

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

{

int k;

char m[100];

int i=0;int j=0;

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

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

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

m[k]=s[n+k];

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

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

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

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

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

printf("%c",m[k]);

}

1. 解题思路：

A：首先，本题考查的是对于字符数组的使用，而且涉及到了函数调用的问题。

B：主要在于用另外一个数组暂时存储一部分字符，然后再将字符交换，从而实现实验的目的。

3）运行截图如下所示：

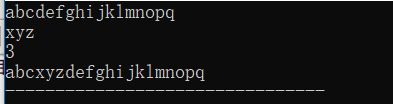


图5-7 选做题1运行截图

1. 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：

1. 源程序代码如下所示：

#include "stdio.h"

int main()

{

int k1,k2,k3,k4,k5,k6,k7,k8;

int m=0;

for(k1=0;k1<8;k1++)

{

for(k2=0;k2<8;k2++)

{

if(k2==k1||k2==k1-1||k2==k1+1) continue;

for(k3=0;k3<8;k3++)

{

if(k3==k1||k3==k2||k3==k2-1||k3==k2+1||k3==k1-2||k3==k1+2) continue;

for(k4=0;k4<8;k4++)

{

if(k4==k1||k4==k2||k4==k3||k4==k3-1||k4==k3+1||k4==k1-3||k4==k1+3||k4==k2-2||k4==k2+2) continue;

for(k5=0;k5<8;k5++)

{

if(k5==k1||k5==k2||k5==k3||k5==k4||k5==k4-1||k5==k4+1||k5==k1+4||k5==k1-4||k5==k2-3||k5==k2+3||k5==k3-2||k5==k3+2) continue;

for(k6=0;k6<8;k6++)

{

if(k6==k1||k6==k2||k6==k3||k6==k4||k6==k5||k6==k5-1||k6==k5+1||k6==k1-5||k6==k1+5||k6==k2-4||k6==k2+4||k6==k3-3||k6==k3+3||k6==k4-2||k6==k4+2) continue;

for(k7=0;k7<8;k7++)

{

if(k7==k1||k7==k2||k7==k3||k7==k4||k7==k5||k7==k6||k7==k6-1||k7==k6+1||k7==k1-6||k7==k1+6||k7==k2-5||k7==k2+5||k7==k3-4||k7==k3+4||k7==k4-3||k7==k4+3||k7==k5-2||k7==k5+2) continue;

for(k8=0;k8<8;k8++)

{

if(k8==k1||k8==k2||k8==k3||k8==k4||k8==k5||k8==k6||k8==k7||k8==k7-1||k8==k7+1||k8==k1-7||k8==k1+7||k8==k2-6||k8==k2+6||k8==k3-5||k8==k3+5||k8==k4-4||k8==k4+4||k8==k5-3||k8==k5+3||k8==k6-2||k8==k6+2) continue;

m++;

printf("%d %d %d %d %d %d %d %d\n",k1,k2,k3,k4,k5,k6,k7,k8);

}

}

}

}

}

}

}

}

printf("%d",m);

return 0;

}

1. 程序设计思路如下所示：

A：首先用到了嵌套循环，并且用到了枚举法来求解。

B：主要是利用了一层层来判断是否满足条件，如果满足就向下一层进行，否则的话就返回后加一。

C：判断时主要利用行数于他们之间距离的关系来进行，如他们在同一条斜线，那么他们的差值绝对值是行数之差。因此来判断是否满足题意。

1. 运行截图如下所示：

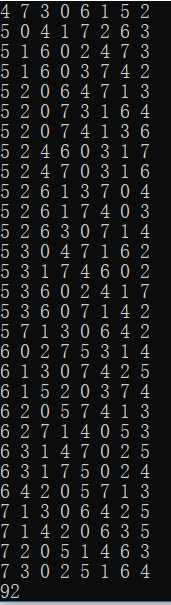
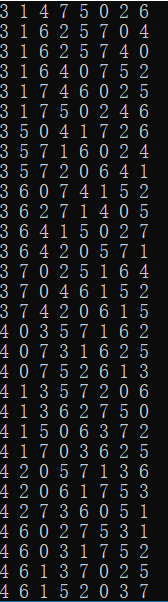
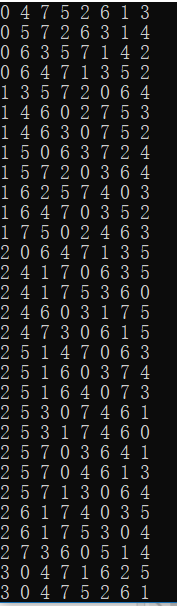


图5-8 选做题2运行截图

## 5.3 实验小结

1）：本次实验难度较大，而且我存在的问题也挺多的，真正去编程的时候发现自己好多地方都不会，有些知识点都记不起来了，导致错误一大堆，浪费了大量的时间。

2）不过本次实验也启示我要多注意细节，因为细节决定成败，而且好多小问题其实是不容易被发现的，就会浪费大量的时间。

3）通过本次实验我发现自己对于字符的处理不够熟悉，而且知识点也较多，自己记得也不是很熟练，运用起来就显得生疏。

4）这次试验让我看到了自己的不足，以后要在C语言中投入更多的时间才可以达到理想中的高度。加油！

# 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. 存在错误，因为定义的指针无所指，属于悬挂指针，非常危险。
2. 源程序改错如下所示：

#include "stdio.h"

void main(void)

{

float n;

float \*p=&n;

scanf("%f",p);

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

}

1. 运行截图如下所示：



图6-1 改错题的运行截图

### 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:

;

}

（2）请上机运行第（1）题程序，使之能按下面要求输出结果：（（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：

1. 分别应该填写的是：
2. char \* (\*p)(char \*s,const char \*t);
3. while((a[i++]=getchar())!='\n');

a[i-1]='\0';

（3） while((b[i++]=getchar())!='\n');

b[i-1]='\0';

1. p
2. 运行截图如下所示：

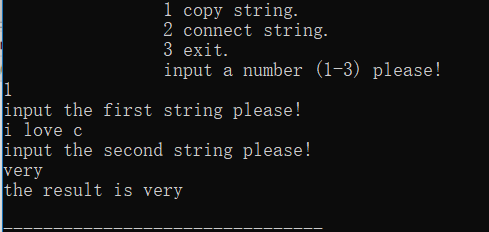


图6-2 程序完善题运行截图

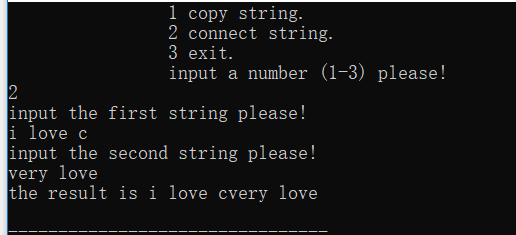


图6-3 程序完善题运行截图

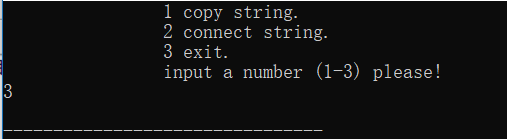


图6-4 程序完善题运行截图

### 6.2.3跟踪调试题

请按下面的要求对源程序进行操作，并回答问题和排除错误。

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

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

#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++)

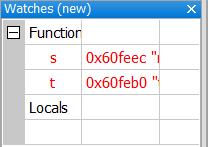
;

return (s);

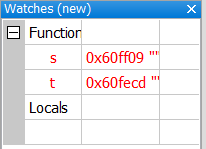
}

解答：

1)进入函数时，s的值为



返回主函数时，s的值为



2）源程序修改：由于函数的运行，导致s的地址发生了变化，因此不能正确输出所得结果。

3）程序修改后代码如下所示：

#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)

{

char \*p=s;

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

return p;

}

1. 运行截图如下所示：



图6-5 跟踪调试题运行截图

### 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：

1. 实验思路：

A：首先要知道整数在计算机内部是以二进制形式存储的，然后要用到按位与的运算。

B；将输入的整数和0xf进行按位与操作可以分别得到每四位的整数，然后判断这个数是不是大于十的，如果是，则转化为a到f中的字符，否则就转化为0到9的字符。

C；最后将这个字符数组输出即可。

1. 源程序代码如下所示：

#include "stdio.h"

int main()

{

char n[8];

char \*p=n;

long m;

scanf("%ld",&m);

for(;p<&n[0]+8;)

{

\*p=m&0xf;

if(\*p<10) \*p=\*p+'0';

else \*p=\*p-10+'a';

p++;

m=m>>4;

\*p=m&0xf;

if(\*p<10) \*p=\*p+'0';

else \*p=\*p-10+'a';

p++;

m=m>>4;

}

p=p-1;

for(;p>=&n[0];p--)

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

return 0;

}

1. 数据测试

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1023 | 000003ff | 000003ff |
| 1024 | 00000400 | 00000400 |
| 1 | 00000001 | 00000001 |

4)运行截图如下所示：



图6-6 编程设计题1运行截图

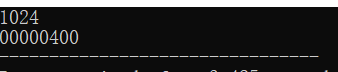


图6-7 编程设计题1运行截图



图6-8 编程设计题运行截图

（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：

1. 实验思路：

A:本题主要考察空格压缩的程序编写和字符指针的应用。

B:空格压缩：首先要有一个字母控制是否是空格，然后通过对该字母的变化，来实现对字符串复制到一个新的字符数组中，然后再将这个字符数组输出即可。

C:对于特殊情况的处理问题主要是对于空行的处理，如果是空行，也即第一个字符是换行符就不于输出即可。

1. 源程序代码如下所示;

#include "stdio.h"

void konggeyasuo(char \*p1,char \*p2);

int main()

{

int i,n;

scanf("%d",&n);

getchar();

char c1[n][81],c2[n][81],\*pc1[n];

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

{

pc1[i]=c1[i];

gets(pc1[i]);

konggeyasuo(pc1[i],c2[i]);

if(c2[i][0]=='\0')

continue;

puts(c2[i]);

}

return 0;

}

void konggeyasuo(char \*p1,char \*p2)

{

int flag=0;

while(\*p1!='\0')

{

if(\*p1!=' ')

\*p2++=\*p1,flag=0;

else if(flag==0)

\*p2++=\*p1,flag=1;

p1+=1;

}

\*p2='\0';

}

3)运行截图如下所示：

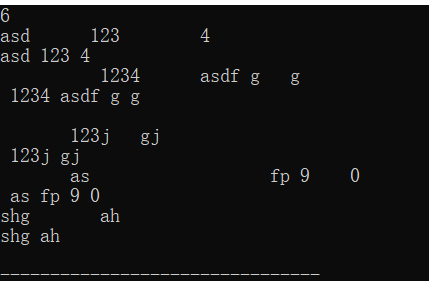


图6-9 编程设计题2运行截图

（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。）

解答：

1）源程序代码如下所示：

#include "stdio.h"

#define N 5

#define M 5

void studentaverage(int \*p1,char \*p2,float \*p3,int x,int y);

void objectaverage(int \*p1,char \*p2,float \*p3,int x,int y);

void lowaverage(int \*p1,char \*p2,float \*p3,int x,int y);

void numberfail(int \*p1,char \*p2,int x,int y);

void numberperfect(int \*p1,char \*p2,int x,int y);

int main()

{

char obname[N][20],stname[N][20];

int i,j,score[N][M],fail[M],perfect[M];

float staverage[N],claverage[M];

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

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

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

{

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

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

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

}

studentaverage(&score[0][0],&stname[0][0],staverage,N,M);

objectaverage(&score[0][0],&obname[0][0],claverage,N,M);

lowaverage(&score[0][0],&obname[0][0],claverage,N,M);

numberfail(&score[0][0],&obname[0][0],N,M);

numberperfect(&score[0][0],&obname[0][0],N,M);

return 0;

}

void studentaverage(int \*p1,char \*p2,float \*p3,int x,int y)

{

int i,j;

float sum;

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

{

sum=0;

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

sum+=\*(p1+i\*y+j);

\*p3=sum/y;

printf("Average score of %s is %.2f\n",p2+i\*20,\*p3);

p3++;

}

}

void objectaverage(int \*p1,char \*p2,float \*p3,int x,int y)

{

int i,j;

float sum;

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

{

sum=0;

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

sum+=\*(p1+j\*y+i);

\*p3=sum/x;

printf("Average score of %s is %.2f\n",p2+i\*20,\*p3);

p3++;

}

}

void lowaverage(int \*p1,char \*p2,float \*p3,int x,int y)

{

int i,j,num;

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

{

num=0;

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

if(\*(p1+j\*y+i)<\*(p3+i))

num+=1;

printf("Number of students lower than avg of %s is %d\n",p2+i\*20,num);

}

}

void numberfail(int \*p1,char \*p2,int x,int y)

{

int i,j,num;

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

{

num=0;

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

if(\*(p1+j\*y+i)<60)

num+=1;

printf("Number of students %s fail is %d\n",p2+i\*20,num);

}

}

void numberperfect(int \*p1,char \*p2,int x,int y)

{

int i,j,num;

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

{

num=0;

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

if(\*(p1+j\*y+i)>=90)

num+=1;

printf("Number of students %s perfect is %d\n",p2+i\*20,num);

}

}

2）运行截图如下所示：

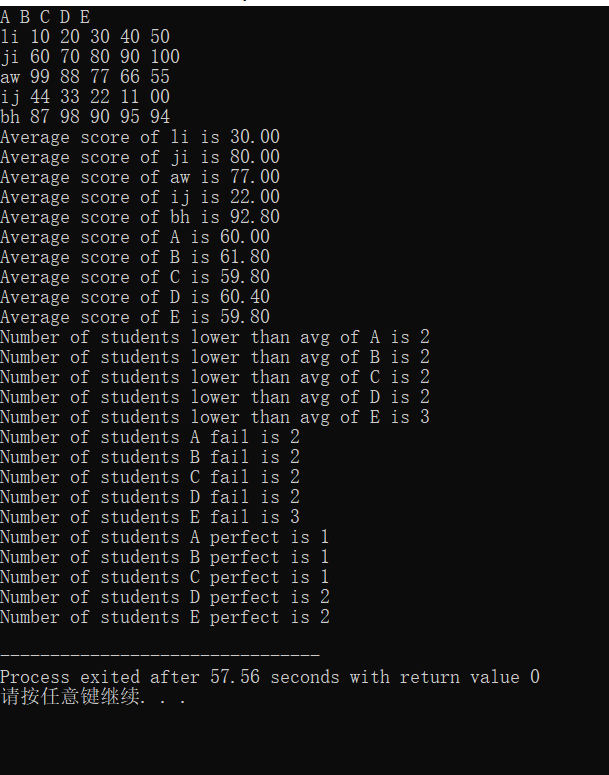


图6-10 编程题4运行截图

3）实验思路：

A:本题主要考察对于指针的应用，对于函数指针的应用。

B:首先要定义字符数组和二维的整形数组和浮点型数组，分别用来存储科目名称，姓名，学生成绩，学生平均成绩，学科平均成绩，同时定义相应类型的指针，便于后续的指针操作。

C:根据假设定义s[N][M],那么s[a][b]就可以用\*(p+a\*M+b)来表示该数值。这是本题不用下标操作的关键之所在。

D:根据数组传递的是数组的地址，因此直接对数组的数据进行操作，函数调用之后就对数组中的元素进行了改变。

### 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. 实验思路：

A:本题是高精度数据的计算。根据两个较大的数相加，所得结果最多比较大的数多一位进行计算。

B:首先将一个数据输入存储到数组中，然后将数组反转，就可以得到低位在前，高位在后，进行相加运算时，要注意进位的情况。

C:然后将相加后的数组输出即可，要注意最高为是否为零，要进行判断是否输出。

* 1. 源程序代码如下所示：

#include "stdio.h"

#define N 20

#define M 10

void reverse(int \*a,int SIZE);

int main()

{

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

int \*pa=a,\*pb=b,\*pc=c;

for(;pc<&c[0]+N+M+1;)

\*pc++=0;

for(;pa<&a[0]+N+M;pa++)

{

if(pa==&a[N]) getchar();

\*pa=getchar()-'0';

}

getchar();

for(;pb<&b[0]+N+M;pb++)

{

if(pb==&b[N]) getchar();

\*pb=getchar()-'0';

}

reverse(a,N+M);

reverse(b,N+M);

for(pc=c,pa=a,pb=b;pc<&c[0]+N+M;pa++,pb++,pc++)

{

\*pc=\*pa+\*pb+\*pc;

if(\*pc>9) \*(pc+1)=1,\*pc=\*pc-10;

}

reverse(c,N+M+1);

if(c[0]==0)

{

for(pc=c+1;pc<&c[0]+N+M+1;pc++)

{

printf("%d",\*pc);

if(pc==&c[N]) printf(".");

}

}

else

{

for(pc=c;pc<&c[0]+N+M+1;pc++)

{

printf("%d",\*pc);

if(pc==&c[N]) printf(".");

}

}

return 0;

}

void reverse(int \*a,int SIZE)

{

int i,j,t;

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

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

* 1. 数据测试如下所示：

|  |  |
| --- | --- |
| 数据输入 | 12345678912345678912.1234567891  98765432198765443198.9876543219 |
| 应该输出 | 111111111111111111111.1111111110 |
| 实际输出 | 111111111111111111111.1111111110 |
| 数据输入 | 11111111111111111111.1111111111  88888888888888888888.8888888889 |
| 应该输出 | 100000000000000000000.0000000000 |
| 实际输出 | 100000000000000000000.0000000000 |

* 1. 运行截图如下所示：

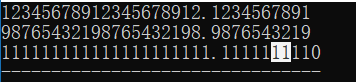


图6-11 选做题1运行截图

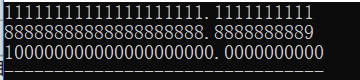


图6-12 选做题1运行截图

（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：

1. 实验思路：

A:本题主要考察复杂的函数声明以及函数指针的应用。

1. 源程序代码如下所示：

#include "stdio.h"

#include "string.h"

int main()

{

int i;

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

char c[2][81],\*pc[2];

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

pc[i]=&c[i][0];

gets(pc[0]);

gets(pc[1]);

p[0]=strcat;

p[1]=strcpy;

printf("%s\n",p[0](pc[0],pc[1]));

printf("%s\n",p[1](pc[0],pc[1]));

return 0;

}

1. 运行截图如下所示：

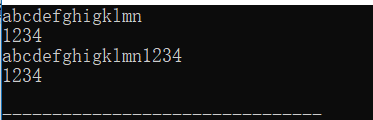


图6-13 选做题2运行截图

**(三) 指定main函数的参数**

选择“project/ set programs' arguments…”菜单命令，即可打开图2.12所示的对话框，在“Program arguments”文本框中输入main函数的参数。注意只输入命令行中文件名后的参数，文件名不输人。

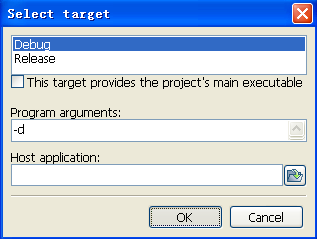
****

图6.？ 输入main函数的参数

## 6.3实验小结

1）本次实验我感觉难度较大，主要考察了对于指针的熟练应用，指针有多种表达方式，因此要掌握不同的表达方式，并且能够熟练应用。

2）本次实验中我充分理解到了定义数组名称和变量名称的时候形象的重要性，因为当有好多数组和指针，变量时，要能够马上找到所要的数组，指针，变量。

因此，定义时一定不能随便的定义，要有所依据。

3）非常感谢向翔宇同学对我的帮助，当我找不到出错的原因时，总是他来帮的我，而且给我鼓励与支持。非常感谢！

# 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 |

解答：

1. 程序验证截图如下所示：

1：

图7-1 表达求值题（1）运行截图

2：

图7-2 表达求值题（2）运行截图

3：

图7-3 表达求值题（3）运行截图

4：

图7-4 表达求值题（4）运行截图

5：

图7-5 表达求值题（5）运行截图

6：

图7-6 表达求值题（6）运行截图

### 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. 错误原因：由于程序使用的是单重指针，所以相当于是对head的引用，然而并未在主函数中改变head的值，因此并不能输出来什么。
2. 错误改正：

A:第（7）（20）行中的函数形参变为双重指针，修改之后如下所示void create\_list (struct s\_list \*\*headp,int \*p)

B:第（12）行中的调用head的地址时形式不正确，正确形式如下所示create\_list(&head,s)

C:第（36）行中的赋值形式不正确，正确形式应该为\*headp=loc\_head

1. 源程序代码如下所示：

#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);

void main(void)

{

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

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

create\_list(&head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

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

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

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

{

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

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

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

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

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

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

tail=tail->next; /\* tail指向新创建的结点 \*/

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

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

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

1. 运行截图如下所示：



图7-7 源程序修改题运行截图

1. 修改替换create\_list函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

解答：

1. 程序设计思路：

主要的程序与上文的相似，不同点在于采用先进后出的链表，只需要只需要使新创建的节点指向已存在的节点即可。

1. 源程序代码如下所示：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data;

struct s\_list \*next;

} ;

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};

create\_list(&head,s);

p=head;

while(p){

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

p=p->next;

}

printf("\n");

}

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

{

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

if(p[0]==0)

;

else {

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

tail->data=\*p++;

tail->next=NULL;

loc\_head=tail;

while(\*p){

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

temp->data=\*p++;

temp->next=loc\_head;

loc\_head=temp;

}

}

\*headp=loc\_head;

}

1. 运行截图如下所示：

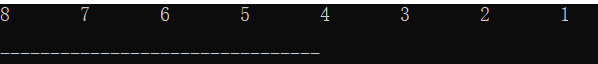


图7-8 源程序替换题运行截图

### 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）：程序设计思路如下所示：

A:首先，本题考查的是结构与联合的基本概念，联合中的成员共享存储，根据联合的这一个特性，便可以做到依次读取一个输的各个二进制数。

2）：源程序代码如下所示：

#include "stdio.h"

#include "stdlib.h"

struct bits{

unsigned char bit0:1;

unsigned char bit1:1;

unsigned char bit2:1;

unsigned char bit3:1;

unsigned char bit4:1;

unsigned char bit5:1;

unsigned char bit6:1;

unsigned char bit7:1;

};

union w8\_bits{

unsigned char c1;

struct bits s1;

};

void f0(struct bits t);

void f0(struct bits t)

{

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

}

void f1(struct bits t);

void f1(struct bits t)

{

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

}

void f2(struct bits t);

void f2(struct bits t)

{

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

}

void f3(struct bits t);

void f3(struct bits t)

{

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

}

void f4(struct bits t);

void f4(struct bits t)

{

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

}

void f5(struct bits t);

void f5(struct bits t)

{

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

}

void f6(struct bits t);

void f6(struct bits t)

{

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

}

void f7(struct bits t);

void f7(struct bits t)

{

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

}

int main()

{

int i;

union w8\_bits w1;

unsigned n;

scanf("%d",&n);

w1.c1=n;

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

p\_fun[0]=f0;

p\_fun[1]=f1;

p\_fun[2]=f2;

p\_fun[3]=f3;

p\_fun[4]=f4;

p\_fun[5]=f5;

p\_fun[6]=f6;

p\_fun[7]=f7;

if(w1.s1.bit0) p\_fun[0](w1.s1);

if(w1.s1.bit1) p\_fun[1](w1.s1);

if(w1.s1.bit2) p\_fun[2](w1.s1);

if(w1.s1.bit3) p\_fun[3](w1.s1);

if(w1.s1.bit4) p\_fun[4](w1.s1);

if(w1.s1.bit5) p\_fun[5](w1.s1);

if(w1.s1.bit6) p\_fun[6](w1.s1);

if(w1.s1.bit7) p\_fun[7](w1.s1);

return 0;

}

1. 数据测试如下所示：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1 | the 1 function 1 is called! | the 1 function 1 is called! |
| 255 | the 1 function 1 is called!  the 2 function 1 is called!  the 3 function 1 is called!  the 4 function 1 is called!  the 5 function 1 is called!  the 6 function 1 is called!  the 7 function 1 is called!  the 8 function 1 is called! | the 1 function 1 is called!  the 2 function 1 is called!  the 3 function 1 is called!  the 4 function 1 is called!  the 5 function 1 is called!  the 6 function 1 is called!  the 7 function 1 is called!  the 8 function 1 is called! |
| 64 | the 7 function 1 is called! | the 7 function 1 is called! |

1. 运行截图如下所示：

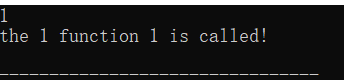


图7-9 程序设计题（1）运行截图

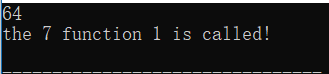


图7-10 程序设计题（1）运行截图

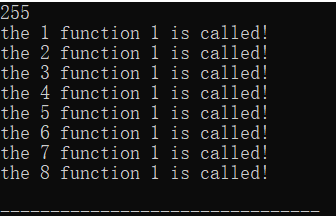


图7-11 程序设计题（1）运行截图

（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. 程序设计思路如下所示：

A：首先本题主要考察单向链表的建立，单向链表的遍历，单向链表的修改。

B:首先建立一个单向链表，主要注意点在于调用函数的时候，注意传递的是地址。

1. 源程序代码如下所示：

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

struct students\_list{

char id[15];

char name[20];

float e;

float m;

float p;

float c;

struct students\_list \*next;

};

struct students\_list \*creat(int n);

void xiugai(char \*p1,char \*p2,float f,struct students\_list \*p);

void average(struct students\_list \*p,float \*f1,float \*f2);

int main()

{

char c1[15],c2[20];

int n,m,i;

scanf("%d",&n);

float score,sum[n],aver[n],\*fs=sum,\*fa=aver;

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

head=creat(n);

p=head;

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

while(p)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->id,p->name,p->e,p->m,p->p,p->c);

p=p->next;

}

putchar('\n');

p=head;

scanf("%d",&m);

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

{

scanf("%s",c1);

scanf("%s",c2);

scanf("%f",&score);

xiugai(c1,c2,score,p);

}

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

p=head;

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

while(p)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->id,p->name,p->e,p->m,p->p,p->c);

p=p->next;

}

putchar('\n');

p=head;

average(head,sum,aver);

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

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

while(p)

{

printf("%-15s%-20s%-10.2f%-10.2f\n",p->id,p->name,\*fs++,\*fa++);

p=p->next;

}

p=head;

fa=aver;

fs=sum;

putchar('\n');

return 0;

}

struct students\_list \*creat(int n)

{

struct students\_list \*head=NULL;

static int i=0;

if(i==n) return NULL;

else

{

head=(struct students\_list \*)malloc(sizeof(struct students\_list));

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

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

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

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

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

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

i++;

head->next=creat(n);

}

return head;

}

void xiugai(char \*p1,char \*p2,float f,struct students\_list \*p)

{

int i;

struct students\_list \*head=p;

while(head)

{

i=strcmp(p1,head->id);

if(!i)

{

if(\*p2=='E') head->e=f;

if(\*p2=='M') head->m=f;

if(\*p2=='P') head->p=f;

if(\*p2=='C') head->c=f;

break;

}

else

head=head->next;

}

}

void average(struct students\_list \*p,float \*f1,float \*f2)

{

struct students\_list \*head=p;

while(head)

{

\*f1++=head->e+head->m+head->p+head->c;

\*f2++=(head->e+head->m+head->p+head->c)/4;

head=head->next;

}

}

1. 运行截图如下所示：

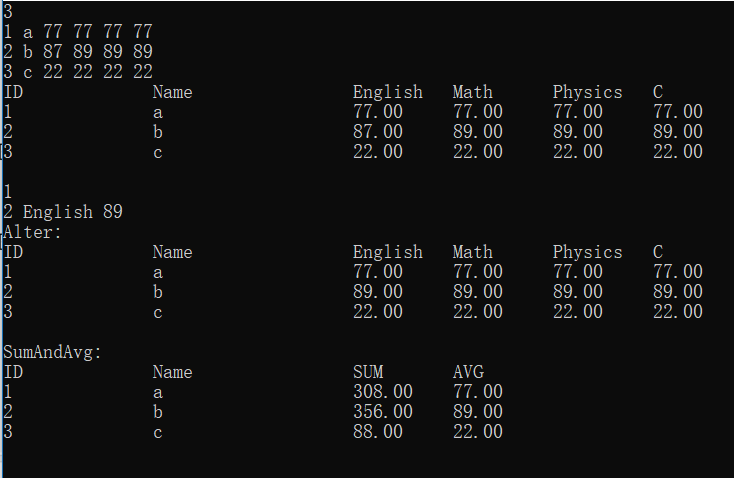


图7-12 程序设计题（2）运行截图

### 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. 主要程序和上题的程序类似，关键在于交换结点数据域，由于数据是字符串，因此要采用字符串的处理函数，即strcpy.
2. 远程源程序代码如下所示：

（由于主要的程序和上一题的相似，所以本次的源代码只显示了交换结点数据域的部分程序）

void sort(struct students\_list \*p,float \*fa)

{

int i,j,len=0;

char c[20];

float \*pf,t;

struct students\_list \*head=p;

while(head)

{

len++;

head=head->next;

}

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

{

head=p;

pf=fa;

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

{

if(\*pf>\*(pf+1))

{

t=\*pf;

\*pf=\*(pf+1);

\*(pf+1)=t;

strcpy(c,head->id);

strcpy(head->id,head->next->id);

strcpy(head->next->id,c);

strcpy(c,head->name);

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

strcpy(head->next->name,c);

}

}

}

}

1. 运行截图如下所示：

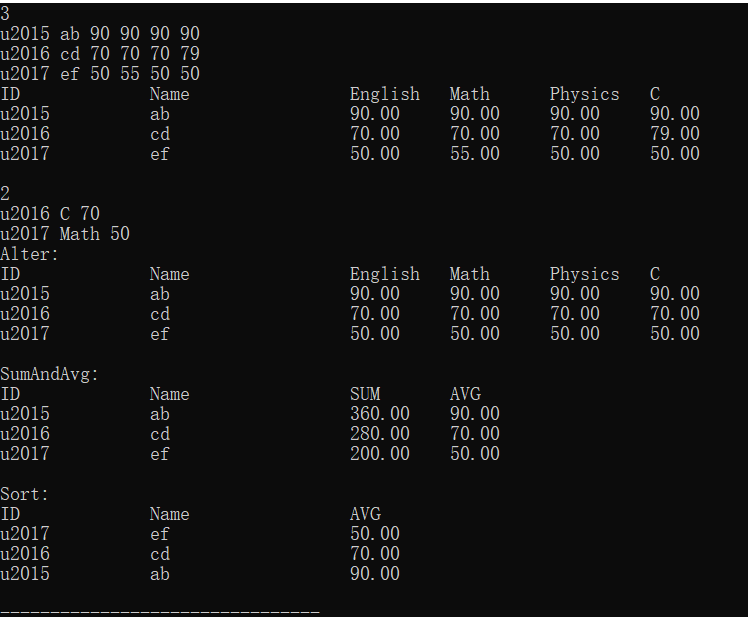


图7-13 选做题（1）运行截图

（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. 程序设计思路如下所示：

本题主要考察单向链表中结点的交换。必须要清楚结点的next的指向，明白交换后结点中的next的指向。

1. 源程序代码如下所示：

（由于本题中的代码与上一题的大部分都是相似的，所以只展示了交换结点的部分）

void sort(struct students\_list \*\*headp,float \*fa)

{

float \*pf=fa,f;

struct students\_list \*prior1, \*prior2, \*p1, \*p2, \*t;

int i=0;

p1=(struct students\_list \*)malloc(sizeof(struct students\_list));

p1->next=\*headp;

(\*headp)=prior1=p1;

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

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

if(\*pf>\*(pf+1))

{

f=\*pf;

\*pf=\*(pf+1);

\*(pf+1)=f;

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);

}

1. 运行截图如下所示：

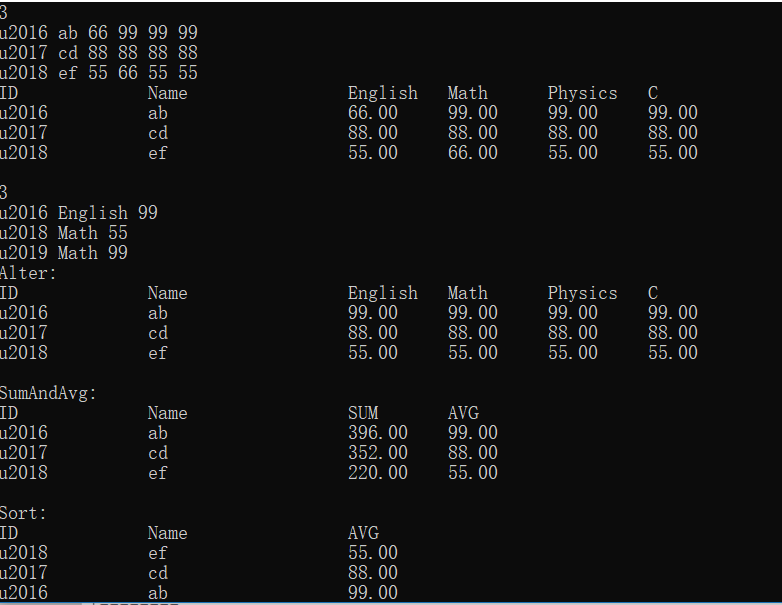


图7-14 选做题（2）运行截图

1. 采用双向链表重做编程设计题第（2）题。

解答：

1. 程序设计思路如下所示:

A:首先建立双向链表时要注意有两个结构指针，一个指针指向下一个结点，另一个指针指向上一个结点，因此使得链表得以循环下去。

B:由于双向链表没有NULL的结构指针，因此在对双向链表的遍历时不能通过判断有没有空指针来结束循环遍历。而是要通过两个指针，一个指向下一个，另一个指向最后的一个指针，当两者相等时循环结束，但是此时还有最后一个没有遍历，因此还要增加一次数据的读取。

1. 源程序代码如下所示：

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

struct students\_list{

char id[15];

char name[20];

float e;

float m;

float p;

float c;

float s;

float a;

struct students\_list \*next;

struct students\_list \*prior;

};

struct students\_list \* creat(int n);

void xiugai(char \*p1,char \*p2,float f,struct students\_list \*p);

void average(struct students\_list \*p);

int main()

{

struct students\_list \*head,\*p1,\*p;

int n,i,m;

char c1[15],c2[20];

float f;

scanf("%d",&n);

head=creat(n);

p=head;

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

while(p!=head->prior)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->id,p->name,p->e,p->m,p->p,p->c);

p=p->next;

}

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->id,p->name,p->e,p->m,p->p,p->c);

putchar('\n');

p=head;

scanf("%d",&m);

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

{

scanf("%s",c1);

scanf("%s",c2);

scanf("%f",&f);

xiugai(c1,c2,f,head);

}

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

p=head;

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

while(p!=head->prior)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->id,p->name,p->e,p->m,p->p,p->c);

p=p->next;

}

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->id,p->name,p->e,p->m,p->p,p->c);

putchar('\n');

p=head;

average(head);

p=head;

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

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

while(p!=head->prior)

{

printf("%-15s%-20s%-10.2f%-10.2f\n",p->id,p->name,p->s,p->a);

p=p->next;

}

printf("%-15s%-20s%-10.2f%-10.2f\n",p->id,p->name,p->s,p->a);

return 0;

}

struct students\_list \* creat(int n)

{

int i=0;

struct students\_list \*headp,\*head,\*tail,\*temp;

headp=(struct students\_list \*)malloc(sizeof(struct students\_list));

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

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

scanf("%f",&headp->e);

scanf("%f",&headp->m);

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

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

head=headp;

i++;

while(i<n)

{

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

tail=temp;

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

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

scanf("%f",&tail->e);

scanf("%f",&tail->m);

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

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

head->next=tail;

tail->prior=head;

head=tail;

tail=temp;

i++;

}

headp->prior=tail;

tail->next=headp;

return headp;

}

void xiugai(char \*p1,char \*p2,float f,struct students\_list \*p)

{

int i;

struct students\_list \*head,\*ps;

ps=p;

while(ps!=p->prior)

{

i=strcmp(p1,ps->id);

if(!i)

{

if(\*p2=='E') ps->e=f;break;

if(\*p2=='M') ps->m=f;break;

if(\*p2=='P') ps->p=f;break;

if(\*p2=='C') ps->c=f;break;

}

else

ps=ps->next;

}

i=strcmp(p1,ps->id);

if(!i)

{

if(\*p2=='E') ps->e=f;

if(\*p2=='M') ps->m=f;

if(\*p2=='P') ps->p=f;

if(\*p2=='C') ps->c=f;

}

}

void average(struct students\_list \*p)

{

struct students\_list \*ps=p;

while(ps!=p->prior)

{

ps->s=ps->e+ps->m+ps->p+ps->c;

ps->a=(ps->e+ps->m+ps->p+ps->c)/4;

ps=ps->next;

}

ps->s=ps->e+ps->m+ps->p+ps->c;

ps->a=(ps->e+ps->m+ps->p+ps->c)/4;

}

1. 运行截图如下所示：

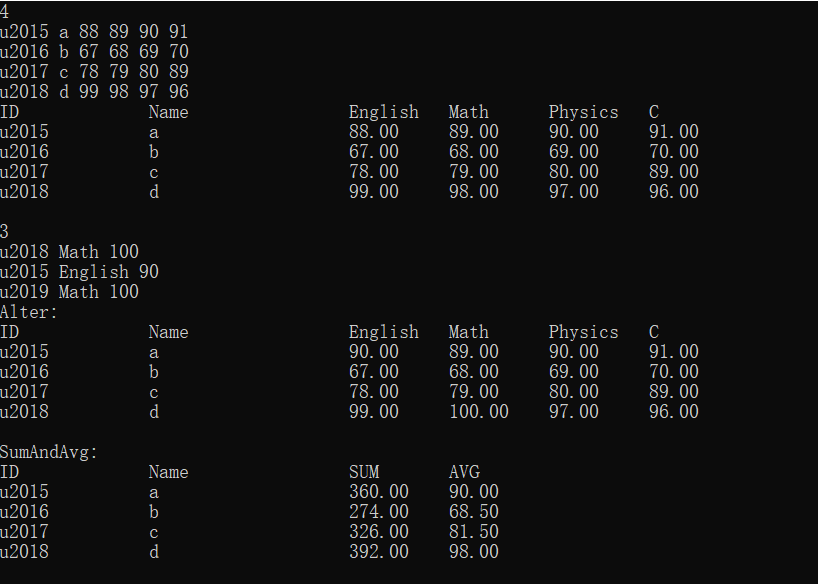


图7-15 选做题（3）运行截图

## 7.3实验小结

1）自我感觉本次实验较难，自己在课下做了好长的时间才写得差不多，在实验课上又写了一会儿才写完。主要难点在于程序越来越长，出错率就越大，而且不容易找到错误点。

2）经过本次的实验，感觉自己收获良多。首先是对于结构与指针的理解更深一步了。我学会了怎样创建单向链表，双向链表，以及对单双向链表的遍历，修改，以及排序。

3）其次，我学会了怎样在一个较长的程序中能看懂自己所写的代码。因为不可能一次就写完，所以要知道自己写的意思，思路，这就要求自己对于变量的设置，自定义函数的命名，要有所含义，而不是随便的a ,b ,c等等。

4）本次实验过后，自己用C语言处理实际问题的能力又增加了一些。

# 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。

解答：程序运行截图如下所示：

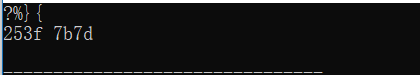


图8-1 程序验证题运行截图

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

解答：由于char类型时一个字节，而short类型是；两个字节，因此只能一个一个字节读取。

运行截图如下所示;

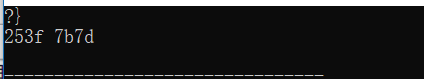


图8-2 程序验证题运行截图

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

解答：hx表示十六进制，而d表示十进制，所以输出的形式不同。

运行截图如下所示：

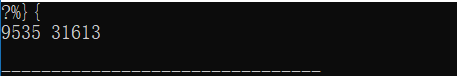


图8-3 程序验证题运行截图

### 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。

#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：

1. 错误修改：将while(ch=fgetc(fp)!=EOF)修改为while(（ch=fgetc(fp))!=EOF)。

（2）用输入输出重定向freopen改写main函数。

解答：

1. 源程序代码如下所示：

#include<stdio.h>

#include<stdlib.h>

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

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=freopen(argv[1],"r",stdin))==NULL){

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

exit(-1);

}

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

putchar(ch);

fclose(fp);

return 0;

}

## 8.3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. 程序设计思路如下所示：

首先，要学会fopen的用法，知道具体的含义是什么，有什么作用。

其次，要学会将一个单词的英文字母首写字母变为大写字母。

1. 源程序代码如下所示：

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

int main()

{

int i=1,k,m;

char c[100],ch;

FILE \*fp;

gets(c);

k=strlen(c);

if(c[0]>='a'&&c[0]<='z')

c[0]-=32;

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

{

if(c[i]==' '&&c[i+1]>='a'&&c[i+1]<='z')

c[i+1]-=32;

i++;

}

fp=fopen("D:\\C语言\\C实验作业第八次\\text.txt","w+");

fprintf(fp,"%s",c);

fclose(fp);

return 0;

}

1. 数据测试如下所示：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| I am a good person! | I Am A Good Person! | I Am A Good Person! |
| i love you | I Love You | I Love You |
| i hate you | I Hate You | I Hate You |

4)运行截图如下所示;

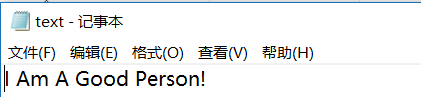


图8-4 程序设计题运行截图

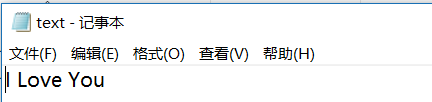


图8-5 程序设计题运行截图

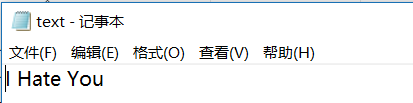


图8-6 程序设计题运行截图

## 8.4实验小结

1）自我感觉，本次实验虽然题目较少，但是并不简单，因为要记得东西太多了，而且自己之前也一直不知道命令行参数实在怎么操作的，所以浪费了大量的时间，多亏了好友蔡子腾同学，在我危难之时帮助我，真的是患难见真情。

2）这次的文件实验真的是让我大开眼界，原来还有这么强大的功能，好喜欢C语言啊。

3）感觉自己的C语言学的不太好，这和自己高中之前从来没有怎末接触电脑有关，好多都不知道，但是这刚好让我又学习到好多知识。我计划在寒假认真的把C语言再学习一下，弥补先天的不足。

# 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