

第四次实验报告

2018.10.5 PB16000702 韦璐

实验目的

While 语句与使用

Do~while 语句与使用

For 语句与使用

While, do~while 与 for 的区别

循环语句嵌套（多重循环）

Break 和 continue 的区别与使用

要点综述

1. While 语句（当循环）

While 语句的一般形式为：

While (表达式) {循环体语句}

- (1) while 语句的功能是：计算表达式的值，若为零，退出循环语句的执行；若为非零，执行循环体语句，执行完毕后，再次执行循环语句，计算表达式的值
- (2) 特点是：先判断后执行，循环体有可能一次也不执行。通常用于事先不能确定循环次数的情况。
- (3) 循环体可以是一个语句，也可以是多个语句。如果循环体包括一个以上的语句，应该用花括号括起来，以复合语句的形式出现；如果不加花括号，则 while 语句的范围只到 while 后面的第一个语句处。

2. 循环控制变量

在表达式中出现的变量可以称为 循环控制变量。循环控制变量的使用必须注意几个：

- (1) 循环控制变量必须有正确的初值
- (2) 在循环体内应有改变其值的语句
- (3) 其值的改变应使得表达式的值最终趋向于零（循环结束条件），否则将形成死循环。

3. Do-while 语句

语句形式

Do{

循环体语句

}while (表达式);

- (1) 特点是：先执行后判断，执行流程是：执行循环体；若表达式的值为非零，则再次执行语句，如此循环直到表达式的值为零。语句至少执行一次。
- (2) 总次数不确定，至少循环一次

4. For

For(表达式 1; 表达式 2; 表达式 3) {循环体语句}

- (1) for 循环的执行流程是，第一步执行表达式 1，第二步执行表达式 2（循环判断条件），第三步当循环条件成立（表达式二的值非零）时执行循环体中的语句及表达式 3；然后重复第二步和第三步，直到循环条件不成立（表达式 2 的值为零）

结束循环。相当于以下形式的 while 语句：

表达式 1;

While (表达式 2) {

循环体语句;

表达式 3;

}

- (2) 表达式 1 一般用于给循环变量或循环体里的某些变量赋初值（常采用逗号表达式的形式），如果循环变量或循环体里的某些变量已经有确定的初值，则表达式 1 可以忽略，但是其后的分号不可以忽略。
 - (3) 表达式 2 可以是关系表达式，逻辑表达式，也可以是数值和字符表达式，只要其值为非零，就执行循环体。
 - (4) 表达式 3 主要用于修改循环变量的值，确保能够结束，如果循环体内已经包括这样的语句，表达式 3 可以忽略。
 - (5) 表达式 1，表达式 2 和表达式 3 也可以是与循环无关的量，三个表达式甚至都可以省略，（分号不是表达式的一部分，不可忽略）。但若省略表达式 2，则循环体中需要有跳出循环的语句。
5. While 语句，do-while 语句和 for 语句的异同
- 一般情况下三种循环语句可以互相替换，可方便地将某种语句改写为另外两种语句，唯一例外的是 do-while 语句，如一开始就不成立条件，while 语句和 for 语句不执行循环体，而 do-while 语句至少执行一次循环体，此时改写较为繁杂。三种语句中 for 语句最灵活，不仅循环控制变量的初始化可以放在表达式 1 中，而且循环控制变量的控制甚至整个循环体都可以放在表达式 3 中。
6. 循环嵌套
- 循环体内包含另一个完整的循环结构，称为循环嵌套，内层循环中还可以继续嵌套循环，这就是多重循环。
- (1) 三种循环（while，do-while，for）可以互相多重嵌套，还可以与分支选择语句（if，switch）构成互相多重嵌套。
 - (2) 循环嵌套时在语法结构上必须注意：
 - 内外循环应全包含不交叉
 - 内外循环的控制变量不能重名。
7. Return 语句
- 语句形式：return(表达式); 或 return;
- Return 语句功能：
- (1) 将程序控制（执行流程控制）返回到主调函数处；
 - (2) 在有表达式时，将表达式的值带回到主调函数的调用处；无表达式时，调用出的值是不确定的。
8. Break 中断语句
- 语句形式：
- Break;
- (1) break 语句功能：
 - 强迫终止循环的执行，使提前退出循环。
 - 中断 switch 语句的执行
 - Break 语句不能用于循环语句和 switch 语句之外的任何地方
9. Continue 语句

功能：结束本次循环，即跳过循环体中 continue 语句下面尚未执行的语句，直接执行下一次是否继续循环的判断。Continue 语句只能用于循环体中。

10. Continue 语句与 break 的区别

- (1) break 语句中断包含 break 语句 的本层的整个循环语句的执行，即终止本层循环；或中断退出 switch 语句。
- (2) continue 只是中断当前循环体的本次执行，而不是终止整个的循环；不能用于 switch 语句。

11. 标号语句与 goto 语句

标号语句的形式：

标号：语句

- (1) C 语句前可以加标号构成标号语句，以供转移引用。
- (2) 标号是一个标识符，遵循标识符命名规则。
- (3) 标号的作用域为标号所在函数，即只允许在定义该标号的函数内语句引用。

Goto 的形式：

Goto 语句标号；

Goto 语句的作用：

- (1) 实现无条件转移。执行到 goto 语句后，程序将转移到标号指定的语句继续执行（不能从循环体外转到体内）。需要注意的是：这种用法不符合结构化程序的要求，应该尽量避免。
- (2) 可用于构成 goto 型循环。

范例

第二题

实验代码：

```
#include<stdio.h>
#include<math.h>
main(){
    double x,x1,f,f1;
    int n=1;
    scanf("%f",&x);
    do{
        x1=x;
        f=pow(x1,3.0)-2.0*pow(x1,2.0)+4.0*x1+1;
        f1=3.0*pow(x1,2.0)-4.0*x1+4.0;
        x=x1-f/f1;
        printf("n=%4d|x|=%-12.7fx=%-12.7f\n",n++,x1,x);
    }while(fabs(x-x1)>=1e-6);
    printf("root=%-12.7f\n",x);
}
```

```

1.0
n=    1x1=0.0000000    x=-0.2500000
n=    2x1=-0.2500000   x=-0.2228916
n=    3x1=-0.2228916   x=-0.2224946
n=    4x1=-0.2224946   x=-0.2224945
root=-0.2224945

```

实验结果：

结果分析：

只用了四步就得到正确的结果，非常快速。

第三题：

实验代码：#include<stdio.h>

#include<math.h>

```

main(){
    double x0,x1,x2,fx0,fx1,fx2;
    do{
        printf("enter x1,x2:");
        scanf("%lf,%lf",&x1,&x2);
        fx1=pow(x1,3.0)-6*x1-1;
        fx2=pow(x2,3.0)-6*x2-1;
    }while(fx1*fx2>0);
    do{
        x0=(x1+x2)/2;
        fx0=pow(x0,3.0)-6*x0-1;
        if(fx0*fx1<0){
            x2=x0;
            fx2=fx0;
        }
        else{
            x1=x0;
            fx1=fx0;
        }
    }while(fabs(fx0)>=1e-5);
    printf("root=%15.12lf\n",x0);
}

```

实验分析：

实验的时候第一次的时候因为输入 scanf 的时候使用的是英文的逗号所以一直在重复打印第一行的东西，没有结果。

实验结果：

```

enter x1, x2:1.0, 4.0
root= 2.528917789459

```

第四题：

题目：编写一个小程序，读入形为 $xxx \cdots x < a > b$ 的字符串，其中 $xxx \cdots x$ 是 a 进制的数字串，

代表一个 a 进制的整数。然后将此整数转换成 b 进制的整数，再输出此整数相应的字符串。A, b 为 2 到 10 之间的数字。如字符串 "1765<8>2", 表示将 8 进制的数字串 1765 转换成 2 进制的数字串后输出。

根据题意，程序需要执行以下工作：

```
{
    读入 a 进制的数字串到数组 digits;
    跳过字符 '<';
    读入数值 a 并转换成相应的整数值;
    跳过字符 '>';
    读入数值 b 并转换成相应的整数值;
    将 digits 中的 a 进制数字串转换成 10 进制整数 c;
    将 c 转换成 b 进制整数并将此整数转换成相应的字符串;
    输出转换后的数字串;
}
```

由上述程序流程可知，为把一个 a 进制的数字串转换 b 进制的数字串，首先将 a 进制数字串转换成十进制整数，然后将十进制整数转换成 b 进制整数，再转换成相应的数字串。具体的转换方法如下：

设 a 进制整数为 $d_n d_{n-1} \cdots d_1 d_0$ ，则此 a 进制整数的十进制整数值为：

$$C = d_0 + d_1 a_1 + \cdots + d_{n-1} a^{n-1} + d_n a^n \\ = ((\cdots (d_n a + d_{n-1}) a + \cdots + d_1) a + d_0$$

将一个十进制整数转换成 b 进制整数的方法是除 b 取余法。下面用实例说明此方法的计算过程。

实验代码：

```
#include <stdio.h>
#define MAXLEN 16
main(){
    int inform,outform,c,i,next;
    char digits[MAXLEN],a,b,ch;
    next=0;
    while((ch=getchar())<='9'&&ch>='0')
        digits[next++]=ch;
    a=getchar();
    if(a<='9'&&a>='2')
        inform=a-'0';
    else
        inform=10;
    getchar();
    b=getchar();
    outform=(b<='9'&&b>='2')?(b-'0'):10;
    for(c=i=0;i<=next-2;i++)
        c=(c+digits[i]-'0')*inform;
    c=c+digits[i]-'0';
    next=0;
    do
```

```

    digits[next++] = c % outform + '0';
    while((c /= outform) > 0);
    for(i = next - 1; i >= 0; i--)
        printf("%c", digits[i]);
}

```

实验结果：

```

1765<8>2
1111110101

```

```

3891<a>7
14226

```

第五题

给定某个游泳池的长度和宽度，已知他的游泳速度和步行速度，现在从一个对角线游到另一个对角线，问从长边的哪一点下水可以达到所求时间最短。

算法描述：本题采用穷举法

1. 假设在边上行走 x 米后下水，历时可以计算。
2. 从这个点下水游到 c 点用时也可以计算
3. 总的用时就可以计算了

从上面的分析可以推测求解的值的范围，精度要控制，所以可以使用枚举法计算最小的时间。

设用 $time$ 存储用时最小值，用 dx 存储 D 点到 A 点的距离，枚举算法：

```

X=0.0,time=32768,dx=0

```

```

While(x<=50.0){

```

```

    Temp=x/1.2+sqrt(WIDTH*WIDTH+(LENGTH-X)*(LENGTH-X)/0.8;

```

```

    If(若所求时间小于等于前次所求最短时间){

```

```

        更新保存最短时间和相应距离两变量的值;

```

```

    }

```

```

    X=x+0.1;

```

```

}

```

实验代码：#include<stdio.h>

```

#include<math.h>

```

```

#define LENGTH 50.0

```

```

#define WIDTH 25.0

```

```

#define V1 0.8

```

```

#define V2 1.2

```

```

main(){

```

```

    float x=0.0,time,temp,dx=0;

```

```

    time=32768;

```

```

    while(x<=50.0){

```

```

        temp=x/V2+sqrt(WIDTH*WIDTH+(LENGTH-x)*(LENGTH-x))/V1;

```

```

        if(temp<=time){

```

```

            time=temp;

```

```

            dx=x;

```

```

    }
    x=x+0.1;
}
printf("time sec=%f,x=%f\n",time,dx);
printf("(50+25)/1.2=%f\n",(LENGTH+WIDTH)/V2);
}

```

实验结果:

```

time sec=64.959061, x=27.600069
(50+25)/1.2=62.500000

```

```

实验代码: #include<stdio.h>
#include<math.h>
double jisuan(double n) {
    double t;
    t=n/1.2+sqrt(25*25+(50-n))/0.8;
    return(t);
}
main() {
    double x,tmin,i;
    for(i=0.02;i<=50;i=i+0.01)
        if(jisuan(i)>jisuan(i+0.01)) {
            tmin=jisuan(i);
            x=i;
        }
    printf("距 A 点%f, 最短时间%f. \n", x, tmin);
}

```

实验结果:

```

time sec=64.959061, x=27.600069
(50+25)/1.2=62.500000

```

```

实验代码: #include<stdio.h>
#include<math.h>
main() {
    double a,b,c,disc,x1,x2,t1,t2;
    a=1;b=-100;c=2000;
    disc=b*b-4*a*c;
    x1=(-b+sqrt(disc))/(2*a);
    x2=(-b-sqrt(disc))/(2*a);
    t1=x1/1.2+sqrt((50-x1)*(50-x1)+25*25)/0.8;
    t2=x2/1.2+sqrt((50-x2)*(50-x2)+25*25)/0.8;
    if(x1>=0&&x1<=50)
        printf("x1=%f t1=%f\n", x1, t1);
    else
        printf("x2=%f t2=%f\n", x2, t2);
}

```

```
}
```

实验结果:

```
time sec=64.959061, x=27.600069
(50+25)/1.2=62.500000
```

实验代码:

```
#include<stdio.h>
#include<math.h>
main() {
    double AD, bD, bC=25.0, AB=50.0, x, x1, t, ft1, ft2;
    scanf("%lf", &x);
    do{
        x1=x;
        ft1=x1*x1-100*x1+2000;
        ft2=2*x-100;
        x=x1-ft1/ft2;
        printf("time min=%f\n", x);
    }while(fabs(x-x1)>=1e-6);
    t=x/1.2+sqrt((50.0-x)*(50.0-x)+25.0*25.0)/0.8;
    printf("!! time min=%f, d=%f\n", t, x);
    printf("(50+50)/1.2=%f\n", (50+25)/1.2);
}
```

实验结果:

```
time sec=64.959061, x=27.600069
(50+25)/1.2=62.500000
```

实验代码:

```
#include<stdio.h>
#include<math.h>
double fx(double x) {
    double fxy;
    fxy=x*x-100*x+2000;
    return(fxy);
}
main() {
    double x0, x1, x2, fx0, fx1, fx2, t;
    do{
        printf("enter x1, x2:");
        scanf("%lf%lf", &x1, &x2);
        fx1=fx(x1);
        fx2=fx(x2);
    }while(fx1*fx2>0);
    do{
        x0=(x1+x2)/2;
```



```

        fx0=fx(x0);
        if(fx0*fx1<0){
            x2=x0;
            fx2=fx0;
        }
        else{
            x1=x0;
            fx1=fx0;
        }
    }while(fabs(fx0)>=1e-6);
    t=x0/1.2+sqrt((50.0-x0)*(50.0-x0)+25.0*25.0)/0.8;
    printf("距 A 点%f 米处下水, 最短时间为%f 秒。\\n", x0, t);
}

```

实验结果:

```

time sec=64.959061, x=27.600069
(50+25)/1.2=62.500000

```

第六题:

刑侦大队对涉及 6 个嫌疑人的一桩疑案进行分析, 已知:

- (1) A,B 至少有一个人犯罪
- (2) AEF 中至少 2 人参与作案
- (3) AD 不可能是同案犯
- (4) BC 或同时作案, 或与本案无关
- (5) CD 中有且仅有一人作案
- (6) 如果 D 没有参与作案, 则 E 也不可能参与作案。

根据以上条件, 分析哪些人是罪犯

要结合逻辑运算符与逻辑表达式, 将案情的每一条件写成逻辑表达式:

- (1) AB 至少有一人作案
可以用变量 A 代表嫌疑人 A, 变量 A=1 表示嫌疑人 A 作案, 0 表示不作案, 类似 B 也是。关系为: $cc1=(A||B)$
- (2) $Cc2=(A\&\&E)||((A\&\&F)||((E\&\&F)))$
- (3) $Cc3=! (A\&\&D)$
- (4) $Cc4=(B\&\&C)||(!B\&\&!C)$
- (5) CD 中有且仅有一人作案, 可以表示为
 $Cc5=(C\&\&!D)||((D\&\&!C))$
- (6) $Cc6=(D||!E)$

实验代码: #include<stdio.h>

```

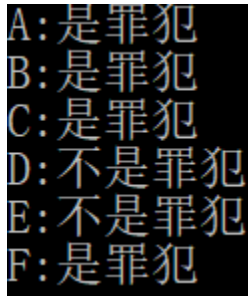
void judge(int x){
    if(x==0)
        printf("不是罪犯\\n");
    else
        printf("是罪犯\\n");
}

```

```

main(){
    int cc1,cc2,cc3,cc4,cc5,cc6;
    int a,b,c,d,e,f;
    for(a=0;a<=1;a++)
    for(b=0;b<=1;b++)
    for(c=0;c<=1;c++)
        for(d=0;d<=1;d++)
            for(e=0;e<=1;e++)
                for(f=0;f<=1;f++){
                    cc1=a||b;
                    cc2=!(a&&d);
                    cc3=(a&&e)||((a&&f)||((e&&f);
                    cc4=(b&&c)||(!b&&!c);
                    cc5=(c&&!d)||((d&&!c);
                    cc6=d||!e;
                    if(cc1&&cc2&&cc3&&cc4&&cc5&&cc6){
                        printf("A:");judge(a);
                        printf("B:");judge(b);
                        printf("C:");judge(c);
                        printf("D:");judge(d);
                        printf("E:");judge(e);
                        printf("F:");judge(f);
                    }
                }
            }
        }
    }
}

```



实验结果：

实验分析：对于这种逻辑性的题目一定要先把所有的取值都自己列出类，然后比遍历，然后选择应该有的结果，然后写出合适的用来判断的逻辑表达式。

实验内容

第一题

实验题目：

输入两个正整数，求其中的最大公约数和最小公倍数

思路：

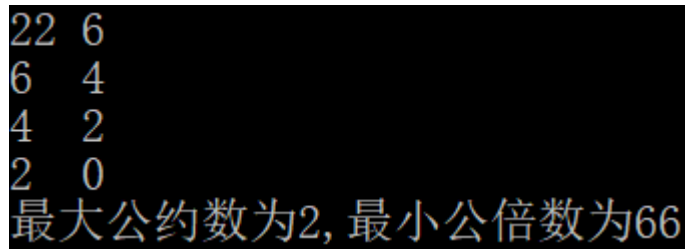
- (1) 求 m 和 n 的余数 r
- (2) 若 r 等于零，则 n 为最大公约数

- (3) 否则令 $m=n, n=r$, 回到第一步
- (4) 最小公倍数就是用两个数除以最大公约数

实验代码:

```
#include<stdio.h>
#include<math.h>
main(){
    int m,n,p,q,r;
    scanf("%d%d",&m,&n);
    p=m;
    q=n;
    do{
        r=m%n;
        m=n;
        n=r;
        printf("%d  %d\n",m,n);
    }while(r);
    printf("最大公约数为%d,最小公倍数为%d",m,p*q/m);
}
```

实验结果:



```
22 6
6 4
4 2
2 0
最大公约数为2, 最小公倍数为66
```

实验心得:

稍微改了一下, 我这里的最小公倍数不是 n 是 m , 要么换一种循环方式也行。

第二题

从键盘输入 50 个学生的学号和三门课的成绩, 计算每个学生的平均成绩并输出其学号和平均成绩, 同时找出其中平均成绩最高和最低的学生并输出其学号和平均成绩。

实验代码:

```
#include <stdio.h>
#include <stdlib.h>
main() {
    FILE *fin;
    int a[50][4];
    int b[50];
    int i,j,k,p,q;
    fin = fopen("data.txt","r"); // 打开文件, 按读的方式打开
    for (i=0;i<50;i++)
        fscanf(fin,"%d %d %d %d", &a[i][0], &a[i][1],&a[i][2],&a[i][3]); // 循环读
    fclose(fin); //关闭文件
    for (i=0;i<50;i++) printf("%d %d %d %d\n",a[i][0],a[i][1],a[i][2],a[i][3]); //输出来看看
```

```
for(i=0;i<50;i++){
    b[i]=(a[i][1]+a[i][2]+a[i][3])/3;
    printf("%d,%d\n",i+1,b[i]);
}
j=b[0];
k=b[0];
for(i=1;i<50;i++){
    if(b[i]>j)
        {j=b[i];p=i;}
    if(b[i]<k)
        {k=b[i];q=i;}
}
printf("平均成绩最高: %d %d, 平均成绩最低: %d %d\n",p+1,j,q+1,k);
return 0;
}
```

实验结果:

1	34	63	78
2	35	62	77
3	52	47	96
4	53	64	57
5	99	65	86
6	77	84	93
7	65	84	88
8	98	97	57
9	47	57	83
10	68	86	88
11	86	48	84
12	68	69	95
13	77	86	66
14	11	35	63
15	99	98	100
16	24	64	84
17	89	88	87
18	69	87	67
19	99	77	88
20	85	81	82
21	83	82	84
22	67	56	55
23	78	88	77
24	77	67	69
25	77	58	55
26	88	83	74
27	74	77	83
28	88	99	77
29	84	84	83
30	89	39	30
31	88	86	65
32	88	85	67
33	88	59	55
34	48	59	30
35	87	97	99
36	95	85	54
37	88	45	45
38	35	69	93
39	98	100	78
40	59	59	59

49	46	43	63
50	66	67	86
1,	58		
2,	58		
3,	65		
4,	58		
5,	83		
6,	84		
7,	79		
8,	84		
9,	62		
10,	80		
11,	72		
12,	77		
13,	76		
14,	36		
15,	99		
16,	57		
17,	88		
18,	74		
19,	88		
20,	82		
21,	83		
22,	59		
23,	81		
24,	71		
25,	63		
26,	81		
27,	78		
28,	88		
29,	83		
30,	52		
31,	79		
32,	80		
33,	67		
34,	45		
35,	94		
36,	78		

```
33, 67
34, 45
35, 94
36, 78
37, 59
38, 65
39, 92
40, 59
41, 78
42, 66
43, 51
44, 55
45, 79
46, 48
47, 56
48, 58
49, 50
50, 73
平均成绩最高: 15 99, 平均成绩最低: 14 36
```

实验总结:

我是从 txt 中读取数据的, 我没有加另一个循环, 对二位数数组不具有通用性, 然后学号也还是特殊取的, 甚至在输出学号的时候都不是从数组取, 然后排列大小的时候初始不应该从放在循环里面, 所以循环要自己做一次, 然后前面算平均数的时候我本来想用 double 的数组的, 结果用了以后发现出来的结果都是没有小数点的, 为了美观我就直接取整数的情况了。

第三题

题目: 写一个程序, 统计输入正文中的英文字母, 空格, 制表符, 换行符和其他字符的个数。输入正文是指从键盘输入的一个字符流。建议采用 getchar 函数读入字符, 并结合循环控制结构完成指定功能。

实验代码:

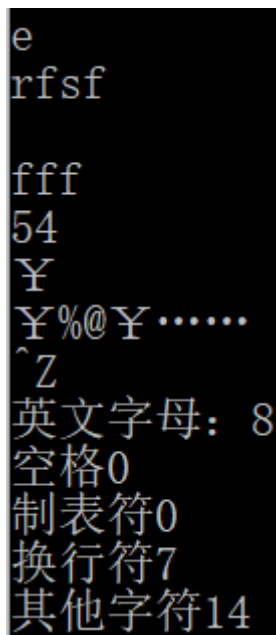
```
#include<stdio.h>
#include<ctype.h>
main(){
    int c1=0,c2=0,c3=0,c4=0,c5=0;
    char c;
    c=getchar();
    while(c!=EOF){
        if(islower(c)||isupper(c))
            c1=c1+1;
        else{
            if(c==' ')
```

```

        c2=c2+1;
    else{
        if(c=='\t')
            c3=c3+1;
        else{
            if(c=='\n')
                c4=c4+1;
            else
                c5=c5+1;
        }
    }
}
c=getchar();//这一行是为了进行下一次的输入而必要的
}
printf("英文字母: %d\n",c1);
printf("空格%d\n",c2);
printf("制表符%d\n",c3);
printf("换行符%d\n",c4);
printf("其他字符%d\n",c5);
}

```

实验结果：



```

e
rfsf

fff
54
Y
Y%@Y.....
^Z
英文字母: 8
空格0
制表符0
换行符7
其他字符14

```

实验总结：

这个实验主要就是 ctype, getchar 和 ctrl+z

第四题

题目：

编写程序，从键盘输入 x，利用幂级数展开计算 $\sin x$ 的近似值，要求误差小于 10^{-6} 。

实验代码：

```
#include<stdio.h>
```

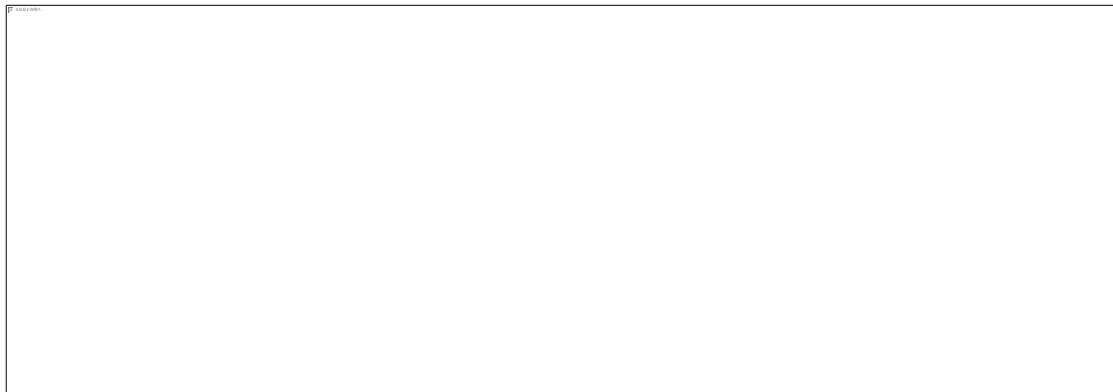


```

#include<math.h>
main(){
    double x;
    int xixi;
    xixi=1;
    int i;
    scanf("%lf",&x);
    double f,sinx;
    f=x;
    sinx=x;
    for(i=3;f>=pow(10,-6);i++){
        printf("3\n");
        if(!(i%2))
        {continue;
        printf("2\n");
        }
        xixi*=i*(i-1);
        f=pow(x,i)/xixi;
        sinx+=pow(-1,(i-1)/2)*f;
        printf("1\n");
    }
    printf("结果是: %lf\n",sinx);
    printf("精确结果是: %12lf,误差是: %12lf",sin(x),sin(x)-sinx);
}

```

实验结果:



实验总结:

Scanf 不知道是什么神奇的存在，在这里面的 bug 多半都可以编译通过；永远!!! 记住永远!!! 确定循环的变量一开始赋值是是正确的!!!!!!!!!!!!!! 我改了 i 改了 xixi (我一点都不想笑嘻嘻了) 改了 f!!!!!!!!!!!!!! 我只想让 dev 狗带。; 论如何愉快的使用 continue; 还有循环体变一点，整个给我重新查一遍。。。疯了疯了

第五题

按下面给出的求 pi 近似值的公式，计算并输出 pi 的值，设 n=1000

实验代码:

```

#include<stdio.h>

```

```
#include<math.h>
main(){
    double pi;
    int i;
    pi=2.0;
    for(i=1;i<1001;i++){
        pi*=pow(2*i,2)/((2*i-1)*(2*i+1));
    }
    printf("%lf",pi);
}
```

实验结果：

3.140808

实验总结：

很好很好

第六题

实验代码：

```
#include<stdio.h>
#include<math.h>
main() {
    int i;
    double N, j;
    N=1;
    j=1;
    for(i=2;i<=20;i++) {
        j*=i;
        N+=j;
        printf("%lf\n", j);
    }
    printf("%lf", N);
}
```

实验结果：

```
2.000000
6.000000
24.000000
120.000000
720.000000
5040.000000
40320.000000
362880.000000
3628800.000000
39916800.000000
479001600.000000
6227020800.000000
87178291200.000000
1307674368000.000000
20922789888000.000000
355687428096000.000000
6402373705728000.000000
121645100408832000.000000
2432902008176640000.000000
2561327494111820300.000000
```

实验总结:

Int 算不对, 会去掉超出的位数, 这样子有的 j 会是负数, 所以需要 double

第七题

实验代码:

```
#include<stdio.h>
#include<math.h>
main() {
    double x,x1,f,f1;
    int n=1;
    scanf("%lf",&x);
    do{
        x1=x;
        f=2*pow(x1,3.0)-4*pow(x1,2.0)+3*x1-6;
        f1=6*pow(x1,2.0)-8*x1+3;
        x=x1-f/f1;
        printf("n=%4dx1=%-12.7fx=%-12.7f\n",n++,x1,x);
    }while(fabs(x-x1)>=1e-6);
    printf("root=%-12.7f\n",x);
}
```

实验结果:

```

1.5
n=    1x1=1.5000000    x=2.3333333
n=    2x1=2.3333333    x=2.0610022
n=    3x1=2.0610022    x=2.0025569
n=    4x1=2.0025569    x=2.0000047
n=    5x1=2.0000047    x=2.0000000
n=    6x1=2.0000000    x=2.0000000
root=2.0000000

```

实验总结：

不用动脑，完全照搬范例

第八题

用二分法求方程在区间内的根

实验代码：

```

#include<stdio.h>
#include<math.h>
main() {
    double x0, x1, x2, fx0, fx1, fx2;
    do{
        printf("enter x1, x2:");
        scanf("%lf, %lf", &x1, &x2);
        fx1=2*pow(x1, 3.0)-4*pow(x1, 2.0)+3*x1-6;
        fx2=2*pow(x2, 3.0)-4*pow(x2, 2.0)+3*x2-6;
    }while(fx1*fx2>0);
    do{
        x0=(x1+x2)/2;
        fx0=2*pow(x0, 3.0)-4*pow(x0, 2.0)+3*x0-6;
        if(fx0*fx1<0) {
            x2=x0;
            fx2=fx0;
        }
        else{
            x1=x0;
            fx1=fx0;
        }
    }while(fabs(fx0)>=1e-5);
    printf("root=%15.12lf\n", x0);
}

```

实验结果：

```

enter x1, x2:-10.0, 10.0
root= 2.000000476837

```

实验总结：

抄范例

第九题

题目：

输出五百以内的最大十个素数并计算和，输出要有文字说明，要求考虑素数不到十个的情况，因为偶数不是素数所以不用判断。

实验代码：

```
#include<stdio.h>
#include<math.h>
int fun(int n){//判断n是否是素数
    int i;
    for(i=2;i<n;i++)
        if(n%i==0) return 0;
    return 1;
}
main() {
    int i,p=1,j;
    for(i=3;i<=500;i++) {
        if(!(i%2))
            continue;
        if(fun(i)) {
            p++;
            printf("%d\n",i);
        }
    }
    if(p<=10)
        printf("太少了");
    printf("五百以内一共有%d个素数\n",p+1);
    int q=1,m=0;
    for(i=3;i<=500;i++) {
        if(!(i%2))
            continue;
        if(fun(i)) {
            q++;
            if(q>(p-10)) {
                m+=i;
                printf("%d+",i);
            }
        }
    }
    printf("0=%d",m);
}
```

实验结果：

```
353
359
367
373
379
383
389
397
401
409
419
421
431
433
439
443
449
457
461
463
467
479
487
491
499
五百以内一共有96个素数
443+449+457+461+463+467+479+487+491+499+0=4696
```

实验总结：

我感觉我的代码很奇怪，可以出结果但是很不规范，第二个要求没明白什么意思，求助教帮我看看，输出格式我懒得调最后一个0了求放过。。。