

# 循环结构程序设计

程 森

E-mail: mew\_cheng@outlook.com

# 循环结构

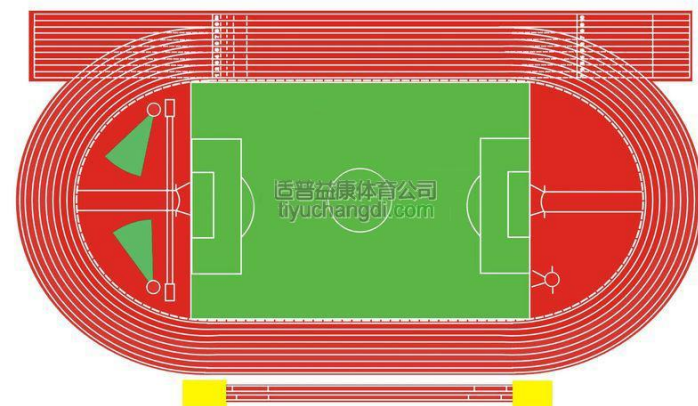
## 为什么需要循环结构

要向计算机输入全班50个学生的成绩

分别统计全班50个学生的平均成绩

求30个整数之和

检查30个学生的成绩是否及格



# 循环结构

**解决办法：分别编写若干个相同或相似的语句或程序段进行处理**

```
scanf("%f, %f, %f, %f, %f", &score1, &score2, &score3, &score4, &score5);  
aver = (score1+score2+score3+score4+score5)/5;  
printf("aver=%7.2f", aver);
```

# 循环结构

在C语言中，用循环语句来处理上面的问题

```
i = 1;
while (i <= 50)
{
    scanf("%f, %f, %f, %f, %f", &score1, &score2, &score3, &score4, &score5);
    aver = (score1+score2+score3+score4+score5)/5;
    i++;
}
```

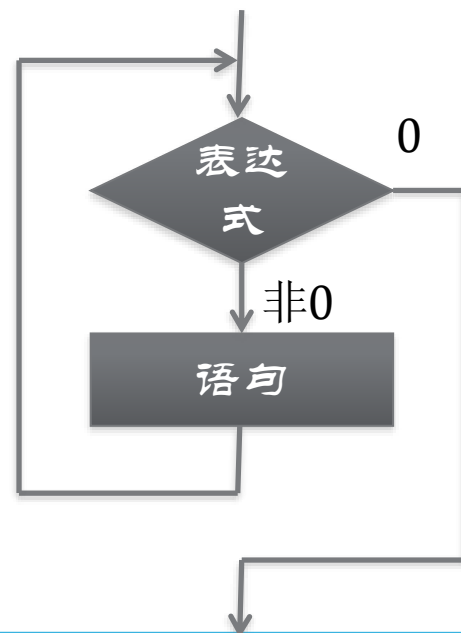
# WHILE语句

while语句的一般形式:

**while (表达式) 语句**

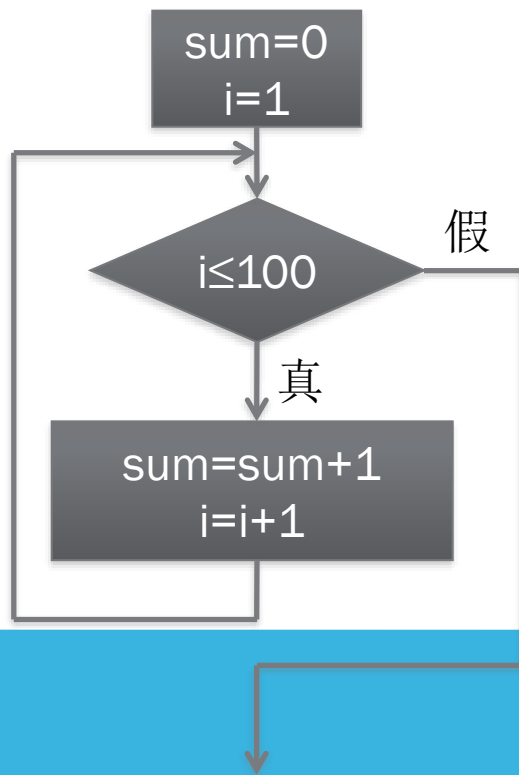
只要当循环条件表达式为真(即给定的条件成立), 就执行循环体语句

先判断条件表达式, 后执行循环体语句



# While语句

先看一个例子：求 $1+2+3+ \dots + 100$ ，即  $\sum_{n=1}^{100} n$



# While语句

## 编写程序

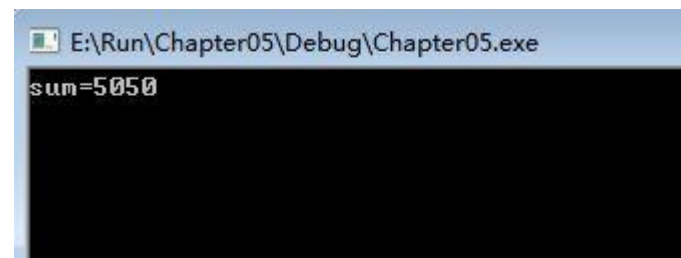
```
#include <stdio.h>

int main()
{
    int i = 1, sum = 0;

    while (i <= 100)
    {
        sum = sum+i;
        i++;
    }

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

    return 0;
}
```



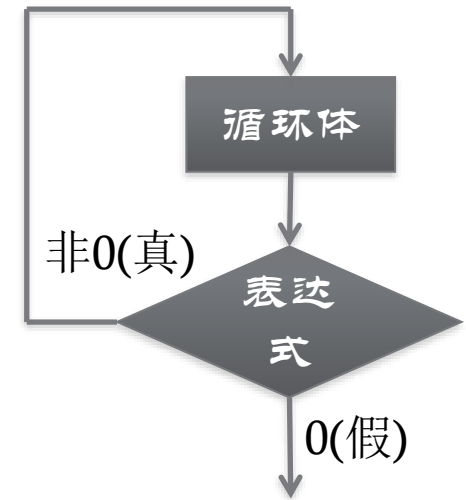
# Do...while语句

do ... while 语句也可以实现循环结构

一般形式为:

**do**  
    **语句**  
**while** (表达式)

特点: 先无条件地执行循环体, 然后判断循环条件是否成立



先跑一圈，热热身

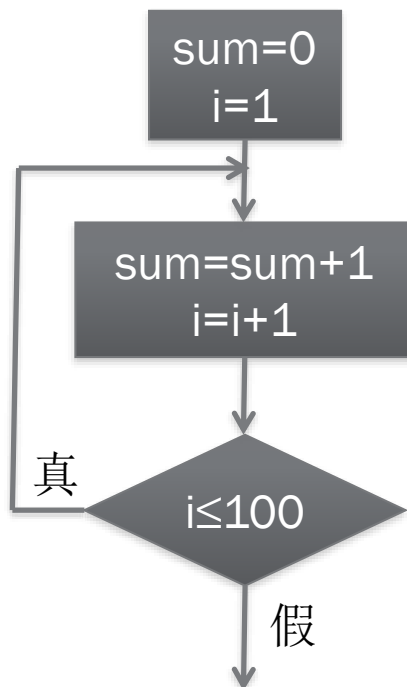




# Do...while语句

用do ... while语句求 $1+2+3+ \dots +100$ ，即

$$\sum_{n=1}^{100} n$$



# Do...while语句

## 编写程序

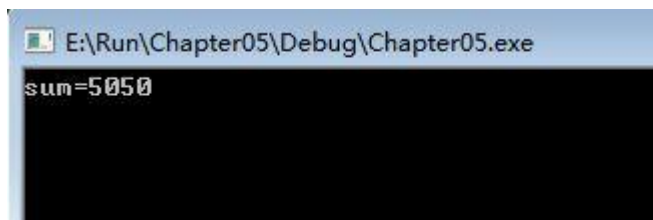
```
#include <stdio.h>

int main()
{
    int i = 1, sum = 0;

    do
    {
        sum = sum + i;
        i++;
    } while (i <= 100);

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

    return 0;
}
```



E:\Run\Chapter05\Debug\Chapter05.exe  
sum=5050

# While和do...while循环

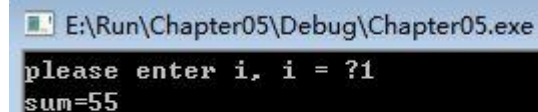
## 用while循环

```
#include <stdio.h>

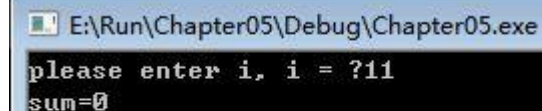
int main()
{
    int i, sum = 0;
    printf("please enter i, i = ?");
    scanf("%d", &i);

    while (i <= 10)
    {
        sum = sum+i;
        i++;
    }

    printf("sum=%d\n", sum);
    return 0;
}
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
please enter i, i = ?1
sum=55
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
please enter i, i = ?11
sum=0
```

# While和do...while循环

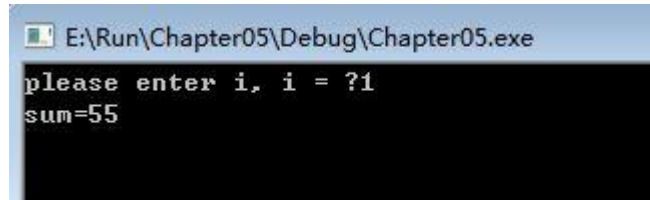
## 用do...while循环

```
#include <stdio.h>

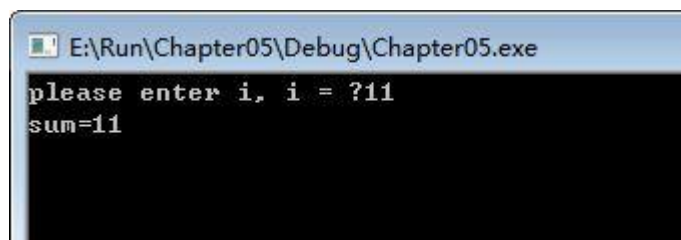
int main()
{
    int i, sum = 0;
    printf("please enter i, i = ?");
    scanf("%d", &i);

    do
    {
        sum = sum+i;
        i++;
    }while(i <= 10);

    printf("sum=%d\n", sum);
    return 0;
}
```



E:\Run\Chapter05\Debug\Chapter05.exe  
please enter i, i = ?1  
sum=55



E:\Run\Chapter05\Debug\Chapter05.exe  
please enter i, i = ?11  
sum=11

# for语句

for语句的一般形式为：

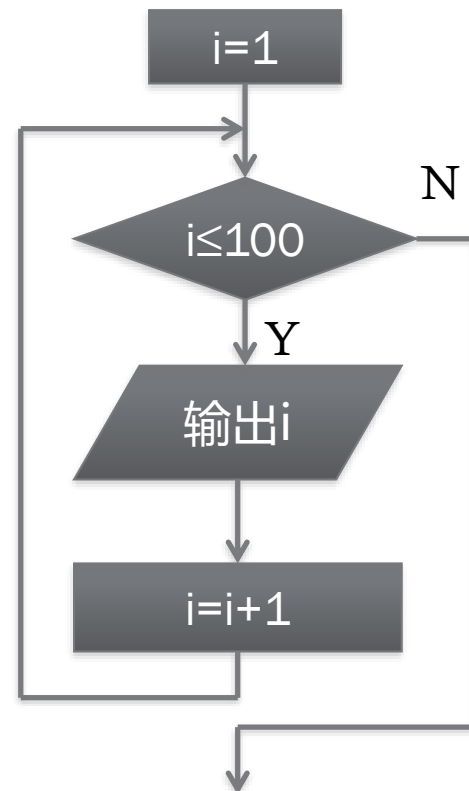
**for(表达式1； 表达式2； 表达式3)**

**语句**

表达式 1： 设置初始条件，只执行一次。可以为零个、一个或多个变量设置初值

表达式 2： 是循环条件表达式，用来判定是否继续循环

表达式3： 作为循环的调整



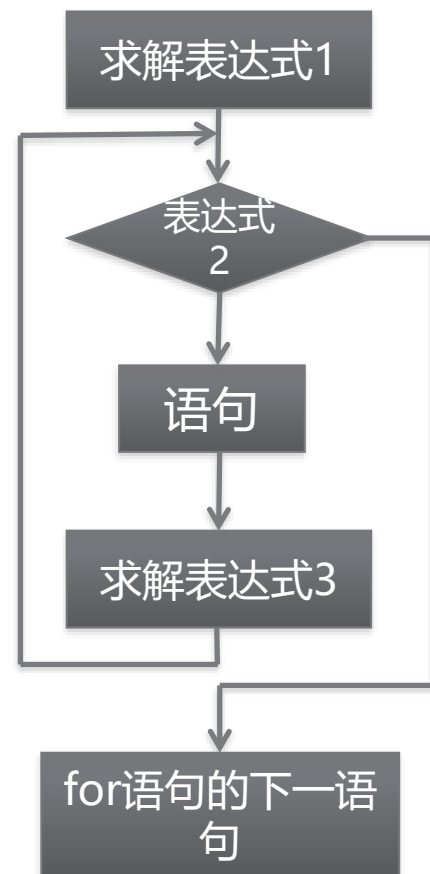
# for语句

for (循环变量赋初值; 循环条件; 循环变量增值)  
语句

```
for (i=1; i<100; i++)  
    sum=sum+i;
```



```
i = 1;  
while (i <= 100)  
{  
    sum=sum+i;  
    i++;  
}
```



# for语句

for语句的等价形式:

```
for (表达式1; 表达式2; 表达式3) 语句
```

```
表达式 1;  
while 表达式 2  
{  
    语句  
    表达式 3  
}
```

# for语句

“表达式1”可以省略，即不设置初值，但“表达式1”后的分号不能省略

```
for (; i<=100; i++) sum=sum+i;
```

✘

```
i = 1;  
for (; i<=100; i++) sum=sum+i;
```

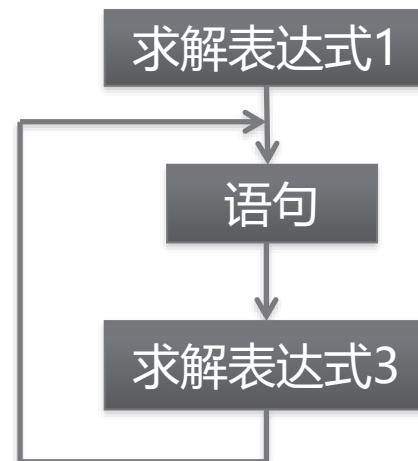


# for语句

“表达式2”也可以省略，即不用“表达式2”来作为循环条件表达式，不设置和检查循环的条件

```
for (i=1; ; i++)    sum=sum+i;
```

```
i = 1;  
while (1)  
{  
    sum=sum+i;  
    i++;  
}
```



# for语句

**表达式3也可以省略**，但此时程序设计者应另外设法保证循环能正常结束

```
for(i=1; i<=100; )  
{  
    sum=sum+i;  
    i++;  
}
```

# for语句

如果表达式1和表达式3都没有，只有表达式2，即只给出循环条件

```
for( ; i<100; )  
{  
    sum=sum+i;  
    i++;  
}
```



```
i = 1;  
for( ; i<100; )  
{  
    sum=sum+i;  
    i++;  
}
```

# For语句

甚至可以将3个表达式都省略

```
for(;;)    printf(“%d\n”, i);
```

```
while(1)    printf(“%d\n”, i);
```

# for语句

表达式1可以是设置循环变量初值的赋值表达式，也可以是与循环变量无关的其他表达式

```
for (sum=0; i<=100; i++)    sum=sum+i;
```

表达式3也可以是与循环控制无关的任意表达式。但不论怎样写for语句，都必须使循环能正常执行

# for语句

表达式1和表达式3可以是一个简单的表达式，也可以是逗号表达式，即包含一个以上的简单表达式，中间用逗号间隔

```
for (sum=0, i=1; i<=100; i++)    sum=sum+i;
```

```
for (i=1, j=100; i<=j; i++, j--)    k=i+j;
```

在逗号表达式内按自左至右顺序求解，整个逗号表达式的值为最右边的表达式的值

# For语句

表达式2一般是关系表达式(如  $i \leq 100$ )或逻辑表达式(如  $a < b \ \&\& \ x < y$ ), 但也可以是数值表达式或字符表达式

只要其值为非零, 就执行循环体

```
for(i=0; (c=getchar())!='\n'; i+=c)    ;
```

此for语句的循环体为空语句, 把本来要在循环体内处理的内容放在表达式3中, 作用是一样的

```
for( ; (c=getchar())!='\n'; )  
    printf("%c", c);
```

# For语句

C99允许在for语句的“表达式1”中定义变量并赋初值

```
for (int i=1; i<=100; i++)  
    sum=sum+i;
```



# 循环的嵌套

3种循环(while循环、do...while循环和for循环)可以互相嵌套

```
(1) while()  
{  
  :  
  while()  
  {...}  
}
```

} 内层循环

```
(3) for(;;)  
{  
  for(;;)  
  {...}  
}
```

} 内层循环

```
(5) for(;;)  
{  
  :  
  while()  
  {...}  
  :  
}
```

} 内层循环

```
(2) do  
{  
  :  
  do  
  {...}  
  while()  
} while()
```

} 内层循环

```
(4) while()  
{  
  :  
  do  
  {...}  
  while();  
  :  
}
```

} 内层循环

```
(6) do  
{  
  :  
  for(;;)  
  {...}  
} while();
```

} 内层循环

# 循环执行的状态

用**break**语句提前终止循环

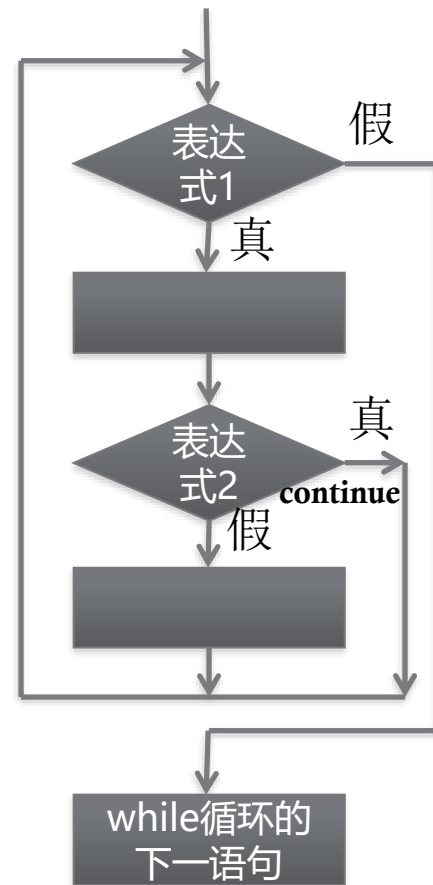
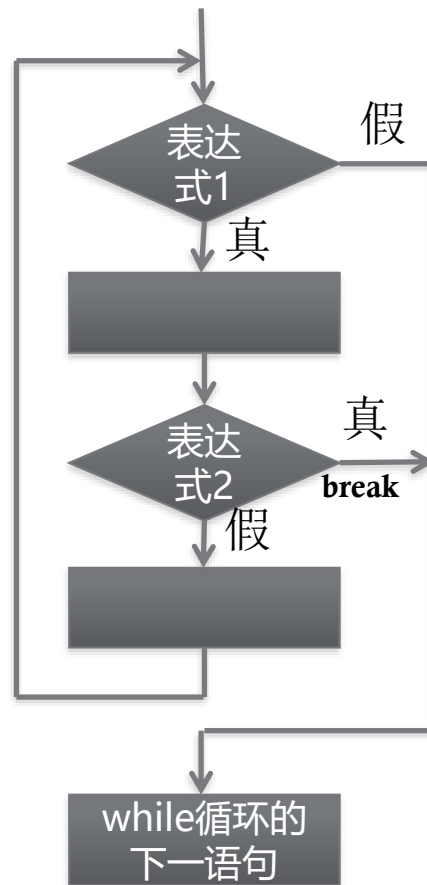
用**continue**语句提前结束本次循环



# 循环执行的状态

```
while (表达式1)
{
    ...
    if(表达式 2)  break;
    ...
}
```

```
while (表达式 1)
{
    ...
    if(表达式 2)  continue;
    ...
}
```



# 循环执行的状态

例子：在全系1000学生中，征集慈善募捐，当总数达到10万元时就结束，统计此时捐款的人数，以及平均每人捐款的数目

```
#include <stdio.h>

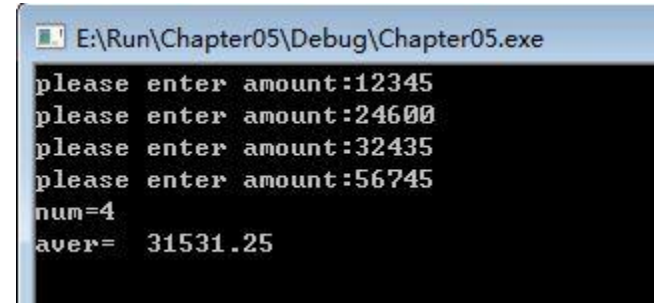
#define SUM 100000

int main()
{
    float amount, aver, total;
    int i;

    for (i = 1; total = 0; i <= 1000; i++)
    {
        printf("please enter amount:");
        scanf("%f", &amount);
        total = total+amount;
        if (total >= SUM) break;
    }

    aver = total/i;
    printf("num=%d\naver=%10.2f\n", i, aver);

    return 0;
}
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
please enter amount:12345
please enter amount:24600
please enter amount:32435
please enter amount:56745
num=4
aver= 31531.25
```

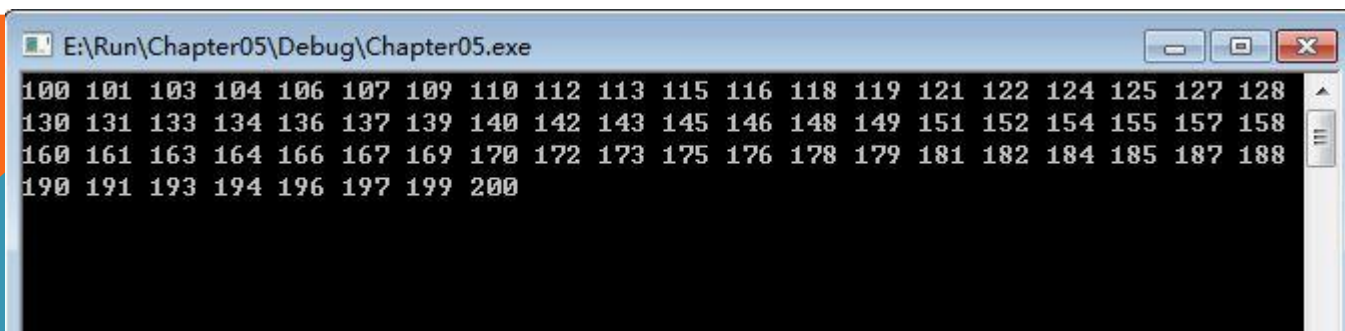
# 循环执行的状态

要求输出100~200之间的不能被3整除的数

```
#include <stdio.h>

int main()
{
    int n;
    for (n = 100; n <= 200; n++)
    {
        if (n%3 == 0)
            continue;
        printf("%d", n);
    }

    printf("\n");
    return 0;
}
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
100 101 103 104 106 107 109 110 112 113 115 116 118 119 121 122 124 125 127 128
130 131 133 134 136 137 139 140 142 143 145 146 148 149 151 152 154 155 157 158
160 161 163 164 166 167 169 170 172 173 175 176 178 179 181 182 184 185 187 188
190 191 193 194 196 197 199 200
```

# 循环执行的状态

输出以下4\*5的矩阵

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20

# 循环执行的状态

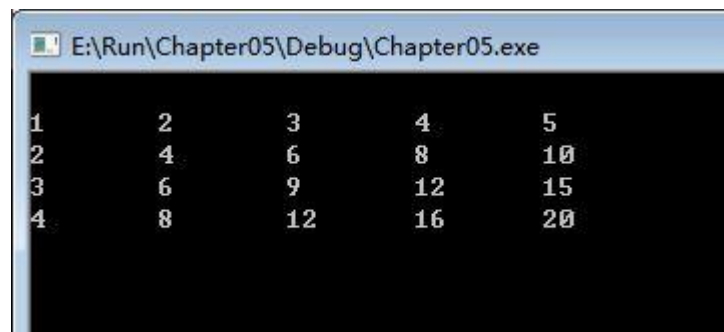
## 编写程序

```
#include <stdio.h>

int main()
{
    int i, j, n = 0;
    for (i = 1; i <= 4; i++)
        for (j = 1; j <= 5; j++, n++)
        {
            if (n%5==0) printf("\n");

            printf("%d\t", i*j);
        }

    printf("\n");
    return 0;
}
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
1      2      3      4      5
2      4      6      8      10
3      6      9      12     15
4      8      12     16     20
```

# 循环执行的状态

用  $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$  公式求的近似值，直到发现某一项的绝对值小于  $10^{-6}$  为止(该项不累加)

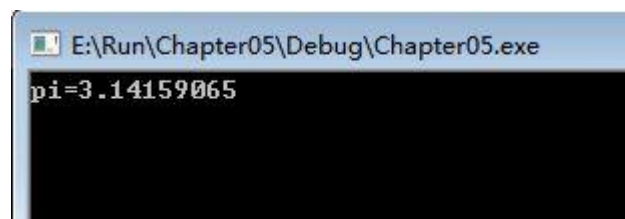
```
#include <stdio.h>
#include <math.h>

int main()
{
    int sign = 1;
    double pi = 0.0, n = 1.0, term = 1.0;

    while (fabs(term) >= 1e-6)
    {
        pi = pi + term;
        n = n + 2;
        sign = -sign;
        term = sign/n;
    }

    pi = pi*4;
    printf("pi=%10.8f\n", pi);

    return 0;
}
```



E:\Run\Chapter05\Debug\Chapter05.exe  
pi=3.14159065



# 循环执行的状态

求Fibonacci数列的前40个数

这个数列有如下特点：第1，2两个数为1, 1。

从第3个数开始，该数是其前面两个数之和。即：

$$\begin{cases} F_1=1 & (n=1) \\ F_2=1 & (n=2) \\ F_n=F_{n-1}+F_{n-2} & (n\geq 3) \end{cases}$$

# 循环执行的状态

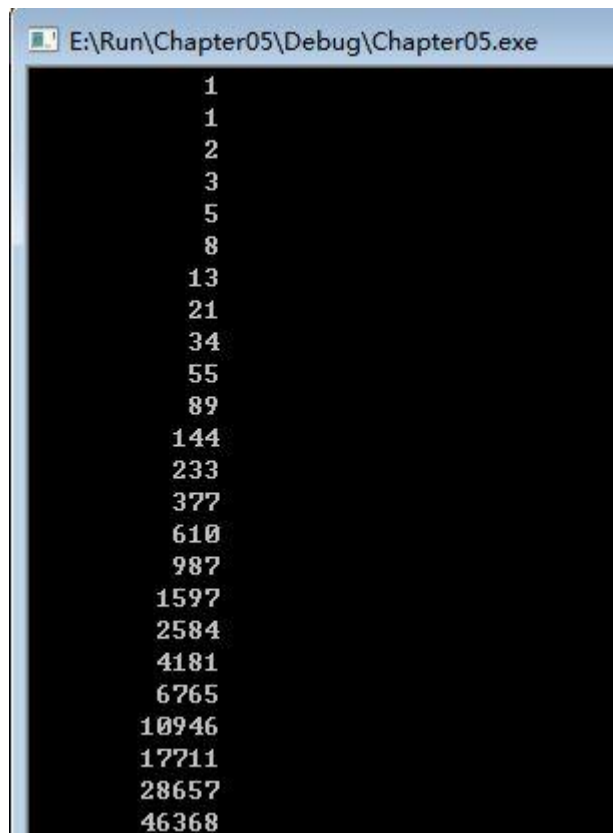
求Fibonacci数列的前40个数

```
#include <stdio.h>

int main()
{
    int f1 = 1, f2 = 1, f3;
    int i;

    printf("%12d\n%12d\n", f1, f2);
    for (i = 1; i <= 38; i++)
    {
        f3 = f1 + f2;
        printf("%12d\n", f3);
        f1 = f2;
        f2 = f3;
    }

    return 0;
}
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
```

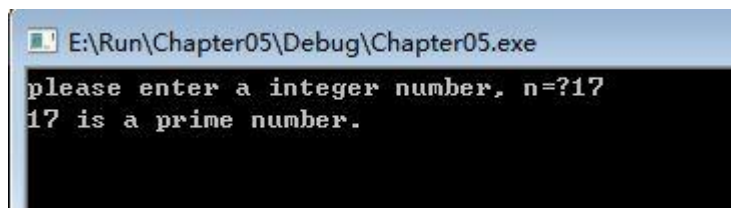
# 循环执行的状态

输入一个大于3的整数n，判定它是否为素数(prime，又称质数)

```
#include <stdio.h>

int main()
{
    int n, i;
    printf("please enter a integer number, n=?");
    scanf("%d", &n);
    for (i = 2; i <= n-1; i++)
        if (n%i == 0) break;
    if (i < n) printf("%d is not a prime number.\n", n);
    else printf("%d is a prime number.\n", n);

    return 0;
}
```



```
E:\Run\Chapter05\Debug\Chapter05.exe
please enter a integer number, n=?17
17 is a prime number.
```

# 循环执行的状态

