循环结构又称为重复结构：用来处理需要重复处理的问题，它是程序中一种很重要的结构。

特点是：当给定条件成立时，反复执行某段程序，直到条件不成立时为止。

给定的条件称为循环条件，反复执行的程序段称为循环体。

C 语言提供了以下形式的循环结构。

（**1**）用 **goto** 语句和 **if** 语句构成循环（已不再使用）；

（**2**）用 **while** 语句构成循环；

（**3**）用 **do-while** 语句构成循环；

（**4**）用 **for** 语句构成循环。

第 5 章　循环结构程序设计

5.1　for 语句

5.2　while 语句

5.3　do…while 语句

5.4　break 和 continue 语句

5.5　程序综合举例

目标

- 理解为什么使用循环结构
- 熟练掌握 **while** 循环的使用
- 熟练掌握 **do-while** 循环的使用
- 理解 **while** 和 **do-while** 循环的区别

问题

1、盈盈为了考验令狐冲夺冠的决心，要他说十遍“我能行！”

10 条

```
printf(“ 第 1 遍说：我能行！ ");
printf(" 第 2 遍说：我能行！ ");
……
printf(" 第 10 遍说：我能行！ ");
```

```
第 2遍说：我能行！
第 3遍说：我能行！
第 4遍说：我能行！
第 5遍说：我能行！
第 6遍说：我能行！
第 7遍说：我能行！
第 8遍说：我能行！
第 9遍说：我能行！
第10 遍说：我能行！
```

问题

2、盈盈要他说 1000 遍“我是最棒的！”，怎么办？

为什么需要循环 2-2

没有使用循环结构

```
printf(" 我是最棒的！ ");
printf(" 我是最棒的！ ");
printf(" 我是最棒的！ ");
printf(" 我是最棒的！ ");
printf(" 我是最棒的！ ");
printf(" 我是最棒的！ ");
… …
printf(" 我是最棒的！ ");
```

使用 while 循环

```
int i = 1;
while ( i<=1000 )
{
   printf( " 我是最棒的！ ");
   i ++;
}
```

演示示例 2：使用循环结构解决问题 1

什么是循环

❖ 生活中的循环

打印 50 份试卷

10000 米赛跑

锲而不舍地学习

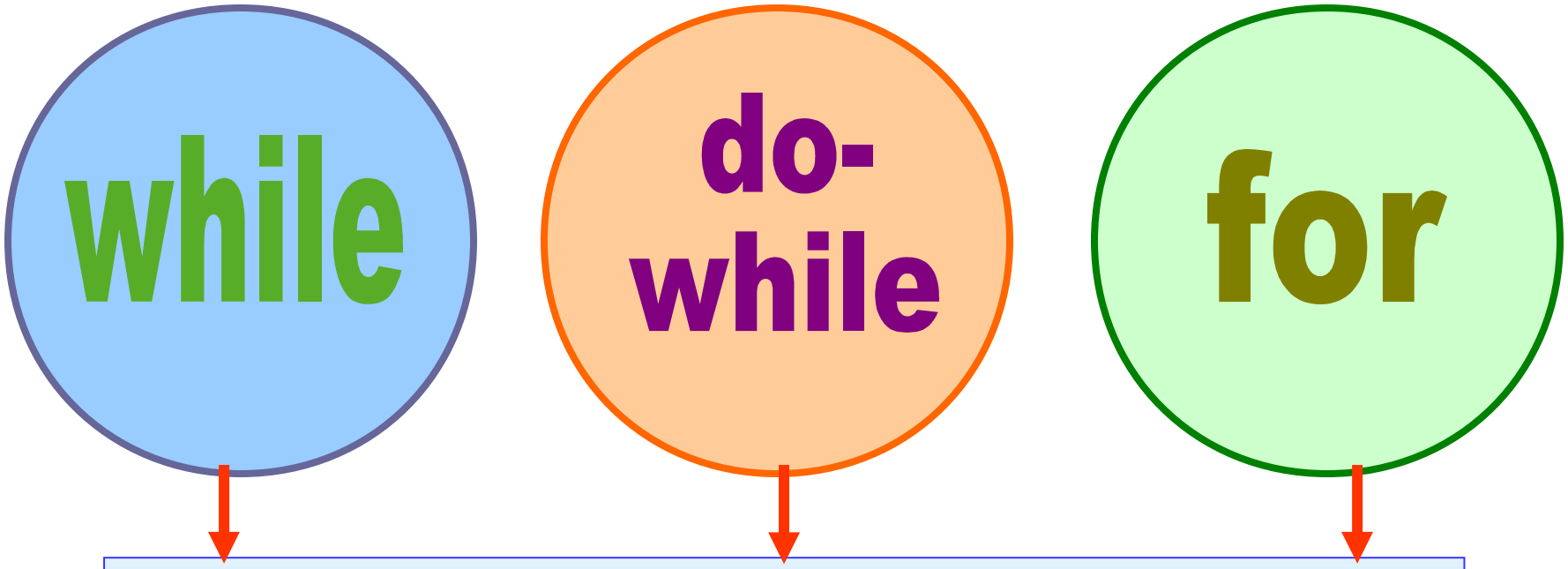
旋转的车轮

循环结构的特点

循环结构

- 循环条件
- 循环操作

C 语言中的各种循环



需要多次重复执行一个或多个任务的问题考虑使用循环来解决

什么是 while 循环

语法

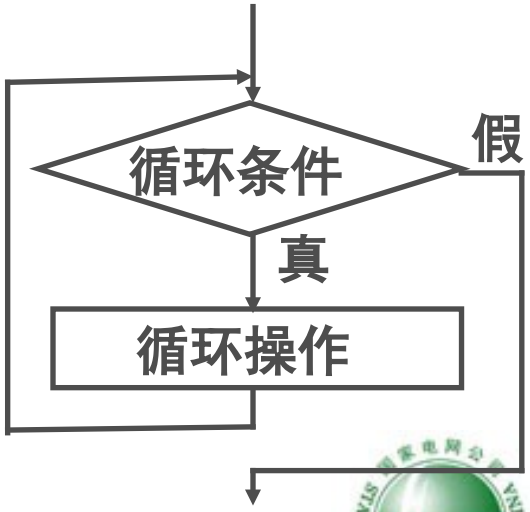
符合条件，循环继续执行；否则，循环退出

```
while ( 循环条件 ) {

        循环操作

}
```

循环中被重复执行的操作

```
int i = 1;
while ( i <= 10 ) {
    printf(" 我能行！ ");
    i ++;
}
```

❖ 特点：先判断，再执行



编码规范：缩进、换行

规则1：

```
[< 初始化 >]

while( 循环条件 )
{
    < 循环体 >
}
```

循环条件中使用的变量需要经过初始化

规则2：

```
while (index < 100)
{
    . . .
    . . .
    index++;
}
```

while 循环主体中的语句必须修改循环条件的值，否则会形成死循环

while (1)　——　真

while 循环常见问题 3-1

```
/*  打印 4 次“欢迎新同学”    */
int main(){
        int i  = 0;
        while (i < 4 ){
                printf(“ 欢迎新同学 ");
                i ++ ;
        }
}
```

永远都不会退出的循环称为**死循环**

while 循环常见问题 3-2


```
/*  打印 4 次“欢迎新同学”   */
int main( ){
        int i= 1;
        while ( i <= 4 ){
            printf(“ 欢迎新同学 ");
            i ++;
        }
}
```

注意检查循环次数是否满足需求

while 循环常见问题 3-3

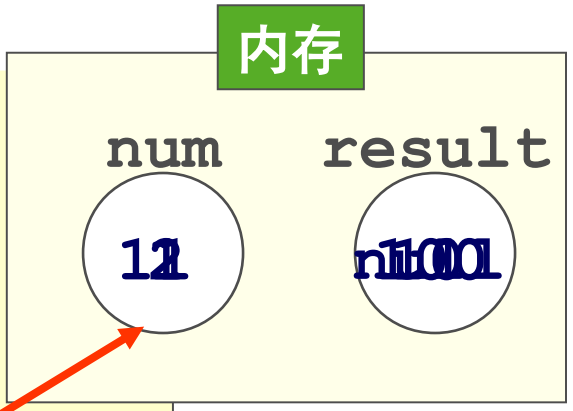
一次都没有打印，
哪里出错了？

```
/*  打印 4 次“欢迎新同学”   */
int main( ){
      int i = 1;
      while ( i < 5 ){
          printf(“ 欢迎新同学 ");
          i ++;
      }
}
```

注意：如果一开始条件就不满足， while 循环一次都不执行

while 循环示例 1

```
#include<stdio.h>
void main ()
{
    int num=1,result;
    while (num<=10)
    {
        result=num*10;
        printf("%d × 10 ＝ %d \n",num,resu
        num++;
    }
}
```



输出：

```
1 × 10 ＝ 10
2 × 10 ＝ 20
3 × 10 ＝ 30
4 × 10 ＝ 40
5 × 10 ＝ 50
6 × 10 ＝ 60
7 × 10 ＝ 70
8 × 10 ＝ 80
9 × 10 ＝ 90
10 × 10 ＝ 100
```

while 循环示例 2

```
#include <stdio.h>
void main ()
{
    int c=0,count=0;
    double f;
    while (c <= 250 && cou
    {
        count++;
        printf("%d: ",count);
        f=c * 9 / 5.0 + 32.0;
        printf("C = %d, F = %
        c = c + 20;
    }
}
```

输出:

```
1: C = 0, F =  32.00
2: C = 20, F =  68.00
3: C = 40, F =  104.00
4: C = 60, F =  140.00
5: C = 80, F =  176.00
6: C = 100, F =  212.00
7: C = 120, F =  248.00
8: C = 140, F =  284.00
9: C = 160, F =  320.00
10: C = 180, F =  356.00
```

while 循环练习

1、 求 1+2+3+...+100 之和；

2、求 1*2*3*...*10 之积；

3、显示 1-10 的平方

4、模拟超市收银系统，计算多件商品的价格，当输入 0 时结束，统计总金额。

4、从键盘输入若干学生的成绩，当输入负数时结束输入，计算平均分。

什么是 do-while 循环

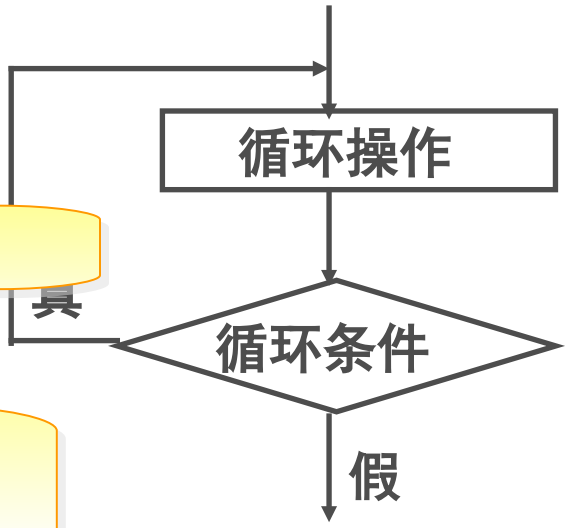
语法

```
do {
    循环操作
} while ( 循环条件 );
```

先执行一遍循环操作

此处有分号！

符合条件，循环继续执行；
否则，循环退出



- 特点：先执行，再判断

do-while 循环示例

```
int number=5,guess;
  printf (" 猜一个介于  1  与  10 之
  do
  {
      printf(" 请输入您猜测的数：
      scanf("%d",&guess);
      if (guess > number)
          printf(" 太大 \n");
      else if (guess < number)
          printf(" 太小 \n");
   }  while (guess != number);
   printf(" 您猜中了！  答案为  %d\n",number);
```

```
猜一个介于  1  与  10  之间的数
请输入您猜测的数：  3
太小
请输入您猜测的数：  5
您猜中了！ 答案为  5
```

输入数字 5 后，do…while 循环中的条件为假， 输出结果消息后，程序终止。

比较 while 和 do-while

❖ **while 循环和 do-while 循环的区别**

- 语法不同

```
while ( 循环条件 ) {
        循环操作
}
```

先判断，再执行

```
do {
        循环操作
} while( 循环条件 );
```

先执行，再判断

- 初始情况不满足循环条件时
  - while 循环一次都不会执行
  - do-while 循环不管任何情况都至少执行一次

为什么使用 for 循环

示例

while 循环结构

```
int i=0;
while(i<100){
    printf(" 我最棒 ");
    i++;
}
```

for 循环结构

```
int i ;
for(i=0;i<100;i++){
    printf(" 我最棒 ");
}
```

for 比 while 更简洁

❖ for 循环的语法和执行顺序

语法

```
for( 1 参数初始化 ; 2 条件判断 ; 4 更新循环变量 ){
3     循环操作 ;
}
```

条件为 true

循环体被执行

```
for ( i = 0 ;  < 100 ;  i++ ) {
        printf(“ 好好学习！ \n");
}
```

代码规范：**格式对齐、代码的缩进**

❖ for 循环的语法和执行顺序

① 首先**计算表达式 1 的值**。

② 再**计算表达式 2 的值**，若值为真（非 0）则执行循环体一次，然后执行下面的第 3 步；若值为假（值为 0）则结束循环，转到第 4 步。

③ **计算表达式 3 的值**，转回第 2 步继续执行。

④ 循环结束，执行 for 语句下面的一个语句。

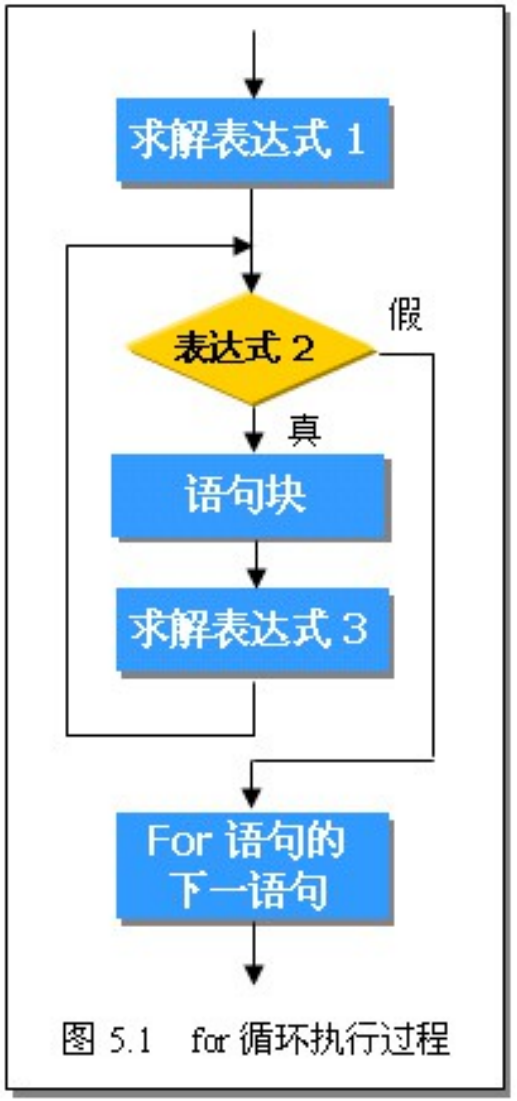
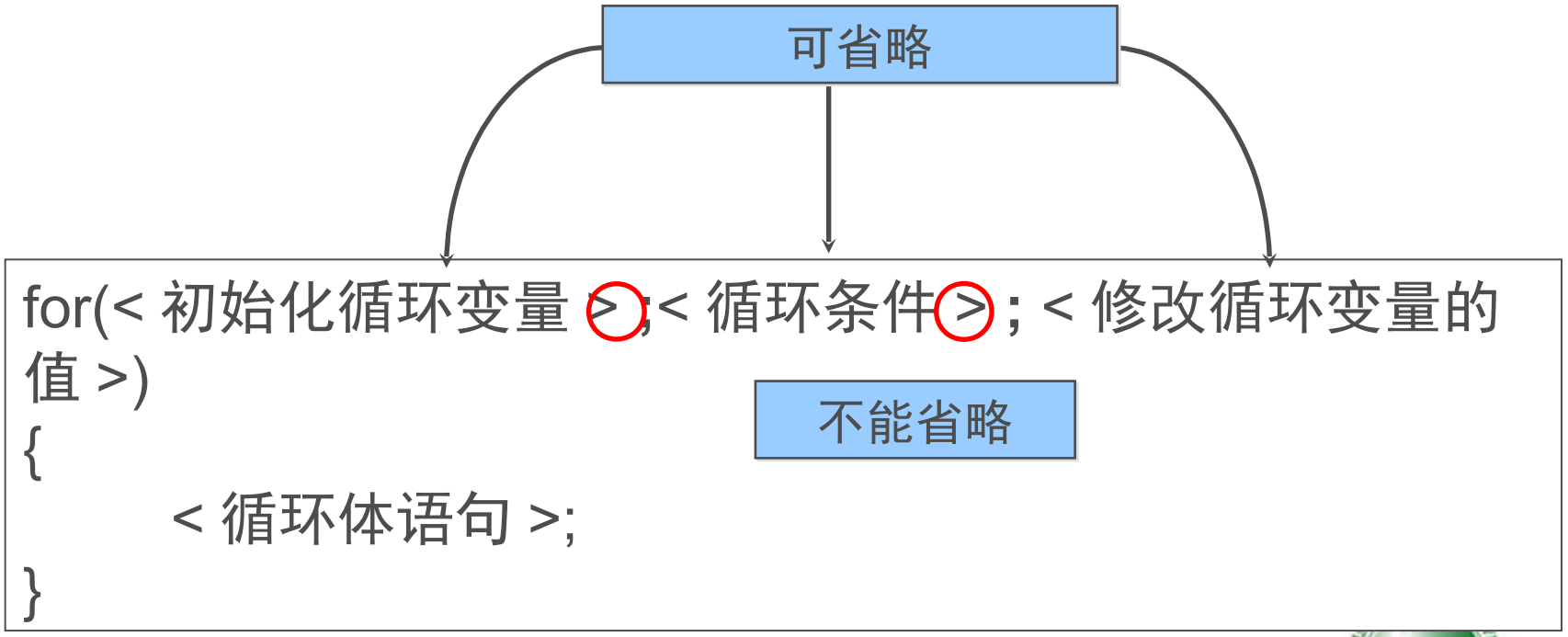


图 5.1　for 循环执行过程

for 循环的表达式

- for 循环中有三个表达式
- for 语句中的各个表达式都可以省略
- 分号分隔符不能省略



1．”表达式 1”省略：但分号不能省略。

例如：

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

又如：

```
#include"stdio.h"
void main()
{  int n=0;
   printf("input a string:\n");
   for(  ;  getchar()!='\n' ; n++) ;
   printf("%d",n);
}
```

2．”表达式 2” 可以省略：这时循环体中必须加入条件判断语句来判断是否结束循环，否则循环会无终止地进行下去形成死循环。分号不能省略。

例如：

```
for ( i=1 ; ; i++ )
  {  sum=sum+i ;
     if ( i>100 ) break ;
  }
```

其中，break 语句的意义是结束循环。如果循环体中去掉 if 语句，则上面的循环为死循环。

3．“表达式 3”可以省略：这时循环体内应加入使循环变量变化的语句，否则也会造成死循环。

例如：

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

4．循环体可以是空语句：可以把循环体要处理的内容放到表达式 3 中，效果是一样的。

例如：

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

循环程序举例

【例 5.1 】用 for 循环语句计算 s=1+2+3+…+9+10 。

【例 5.2 】用 for 语句编写程序：求 100-999 之间的所有水仙花数。

水仙花数：如果一个数的各个数位上的数字之立方和等于该数本身，则此数为水仙花数。如 $153=1^3 + 5^3 + 3^3$

【例 5.3 】求 1-100 之间能被 2 和 3 整数的数，要求每行输出 10 个数字。

【例 5.4 】用 for 语句编写程序。从 0 开始，输出 n 个连续的偶数。

【例 5.5 】用 while 语句修改【例 5.2 】题：从 0 开始，输出 n 个连续的偶数。

```
int main()
{   int a=0,n;
    printf("\n input n: ");
    scanf("%d",&n);
    while (n--)
        printf("%d ",a++*2);
 }
```

【例 5.6 】用 do-while 语句修改【例 5.2 】题：从 0 开始，输出 n 个连续的偶数。

```
int  main()
  {  int a=0,n;
     printf("\n input n: ");
     scanf("%d",&n);
     do{   printf("%d ",a++*2);
        }while (--n);
  }
```

试问：**本例中循环条件继续用（ n-- ）会出现什么问题？**

答：将会多执行一次循环。

为什么：这是由于先执行后判断而造成的。

为什么需要 break 语句

- 描述 10000 米长跑比赛 for 语句

```
for (i = 0; i<10; i++) {
        跑 400 米;
    if ( 不能坚持 ) {
        break;    // 退出比赛
    }
}
// 其他语句
```

第 8 圈，
快累死了…
我要退出…

遇到 break ， 立即跳出 switch 语句

什么是 break 语句

- **break ：改变程序控制流**
  - 用于 do-while 、 while 、 for 中时，可跳出循环而执行循环后面的语句

```
while(…) {
        ……
        ……
        ……
    break;
        ……
        ……
        ……
}
```

break 通常在循环中与条件语句一起使用

跳出整个循环

break 语句 2-1

- break 语句可以改变程序的控制流
- break 语句用于 do-while 、 while 、 for 循环中时，可使程序终止循环而执行循环后面的语句
- break 语句通常在循环中与条件语句一起使用。若条件值为真，将跳出循环，控制流转向循环后面的语句
- 如果已执行 break 语句，就不会执行循环体中位于 break 语句后的语句
- 在多层循环中，一个 break 语句只向外跳一层

break 语句 2-2

跳出 for 循环

```
for( ; ; )
{
    printf(" 这将一直进行下去 ");
    i = getchar();
    if(i == 'X' || i == 'x')
        break;
}
```

跳出 while 循环

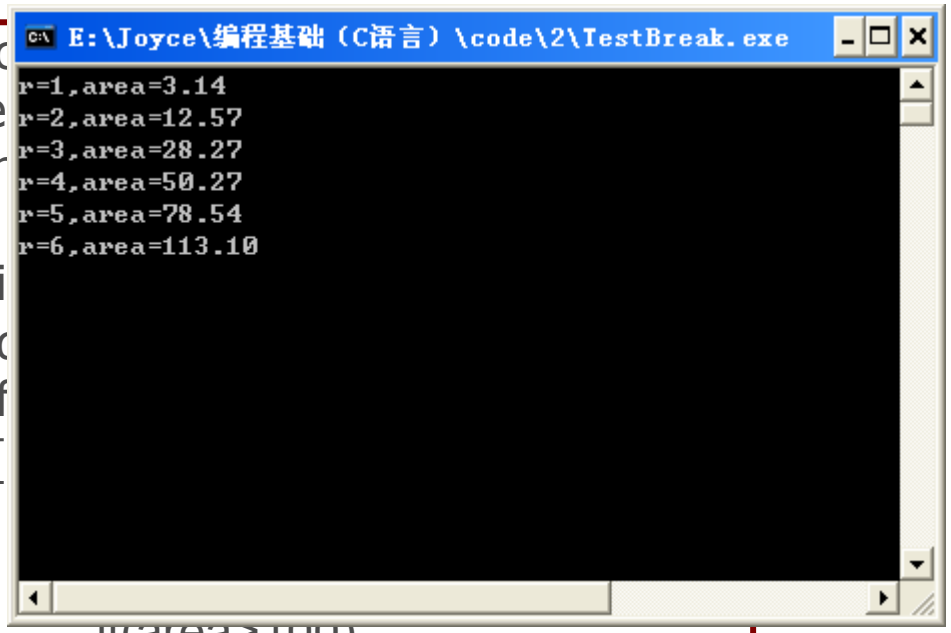
```
while(1)
{
    if(x == 10)
        break;
}
```

跳出 do-while 循环

```
do
{
    if (x == 10)
        break;
}while (x< 15);
```

break 语句示例

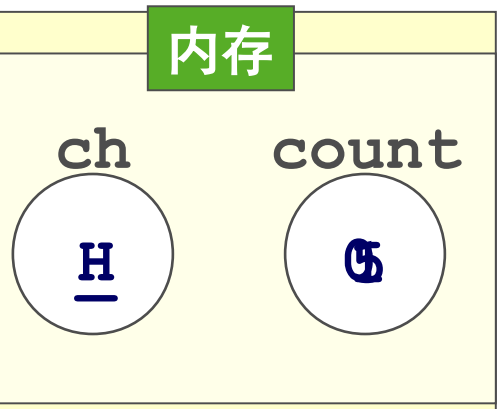
- 输出半径为 1 到 10 的圆的面积，若面积超过 100 ，则结束输出 。




```
#inc
#de
int r
{
        i
        c
        f
      {
            if(area>100)
                break;
      }
      return 0;
}
```

break 语句示例

```
#include<stdio.h>
void main()
{
        int count=0,ch;
        printf("\n  请输入一行字符：  ");
        while((ch=getchar())!='\n')
        {
                if(ch==' ')
                        break;
                count++;
        }
        printf("\n  共有  %d  个有效字符。 \n",count);
}
```



循环执行　5　次

请输入一行字符： Hello world
共有　5　个　有效字符

什么是 continue

- **continue** ：只能用在循环里
- **continue** 作用：跳过循环体中剩余的语句而执行下一次循环

```
while(…) {
        ……
        ……
        ……
    continue;
        ……
        ……
}
```

继续下一次循环

示例

```
for(int i = 0; i<10;i++){
        跑 400 米;
    if （！口渴）{
      continue; // 不喝水，继续跑
    }
        接过水壶，喝水 ;
}
```

通常与条件语句一起使用，加速循环

Continue 语句

- **continue** 语句只能用在循环里
- **continue** 语句的作用是跳过循环体中剩余的语句而执行下一次循环
- 对于 **while** 和 **do-while** 循环，**continue** 语句执行之后的动作是条件判断；对于 **for** 循环，随后的动作是变量更新（表达式 **3**）

continue 语句示例

❖ 例：说出下列程序的功能：

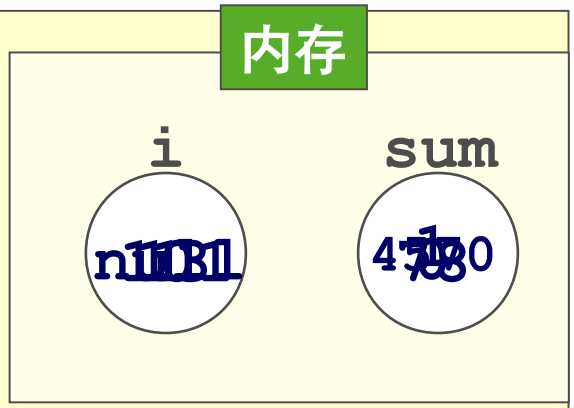
```
#include<stdio.h>
int main()
{
        int num;
        for(num=100;num<=450;num++)
        {
                if(num>150 && num<400)
                        continue;
                if(num%9==0)
                        printf("%6d",num);
        }
        return 0;
}
```

输出 100~150 之间和 400~450 之间能被 9 整除的数

完成下列程序：输出 100~200 之间不能被 3 整除的数

```
#include<stdio.h>
int main()
{
    int k;
    for(k=100;k<=200;k++)
    {
            if(k%3==0)
               continue;
            printf(" %d ",k);
    }
    return 0;
}
```

```
#include<stdio.h>
void main()
{
        int i,sum = 0;
        for(i=1; i<=100;i++)
        {
                if( i % 10 == 3)
                        continue;
                sum += i;
        }
        printf("sum = %d \n",sum);
}
```



循环执行到　i = 13

循环执行到　i = 101

输出：
sum = 4570

循环语句的嵌套

❖ **循环语句的嵌套**：一个循环语句的循环体中又包含循环语句

```
while()
{   ......
    while()
     {
      ......
     }
     ......
}
```

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

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

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

- 三种循环可互相嵌套，层数不限。
- 外层循环可包含两个以上内循环，但不能相互交叉。
- 嵌套循环的执行流程：外层循环执行一次，内层循环要执行完。
- 嵌套循环的跳转：只能跳转出本层循环。
- 禁止从外层跳入内层、禁止跳入同层的另一循环和向上跳转。

嵌套循环示例 1

```
void main()
{
    int i,j,k;
    for(i=1;i<=4;i++)
    {
        for(j=1;j<=4-i;j++)
            printf(" ");
        for(k=1;k<=2*i-1;k++)
            printf("*");
        printf("\n");
    }
    for(i=1;i<=3;i++)
    {
        for(j=1;j<=i;j++)
            printf(" ");
        for(k=1;k<=7-2*i;k++)
            printf("*");
        printf("\n");
    }
}
```

控制打印的行数

控制每行打印的空格数

控制每行打印的 * 号数

控制打印的行数

控制每行打印的空格数

控制每行打印的 * 号数

输出：

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

嵌套循环示例 2

```
void main()
{
    int i,j,n;
    n=0;
   for(i=100;i<=200;i++)
   {
      j=2;
      while(i%j!=0)
          j++;
      if(i==j)
      {
          printf("%4d",i);
          n++;
          if(n%8==0)
              printf("\n");
      }
   }
   printf("\n");
}
```

从 2 到 i 之间寻找第一个能被整除的数

如果第一个能被整除的数等于该数本身，则说明该数为素数

控制每行输出 8 个素数

输出：

从 100 到 200 之间所有的素数为：

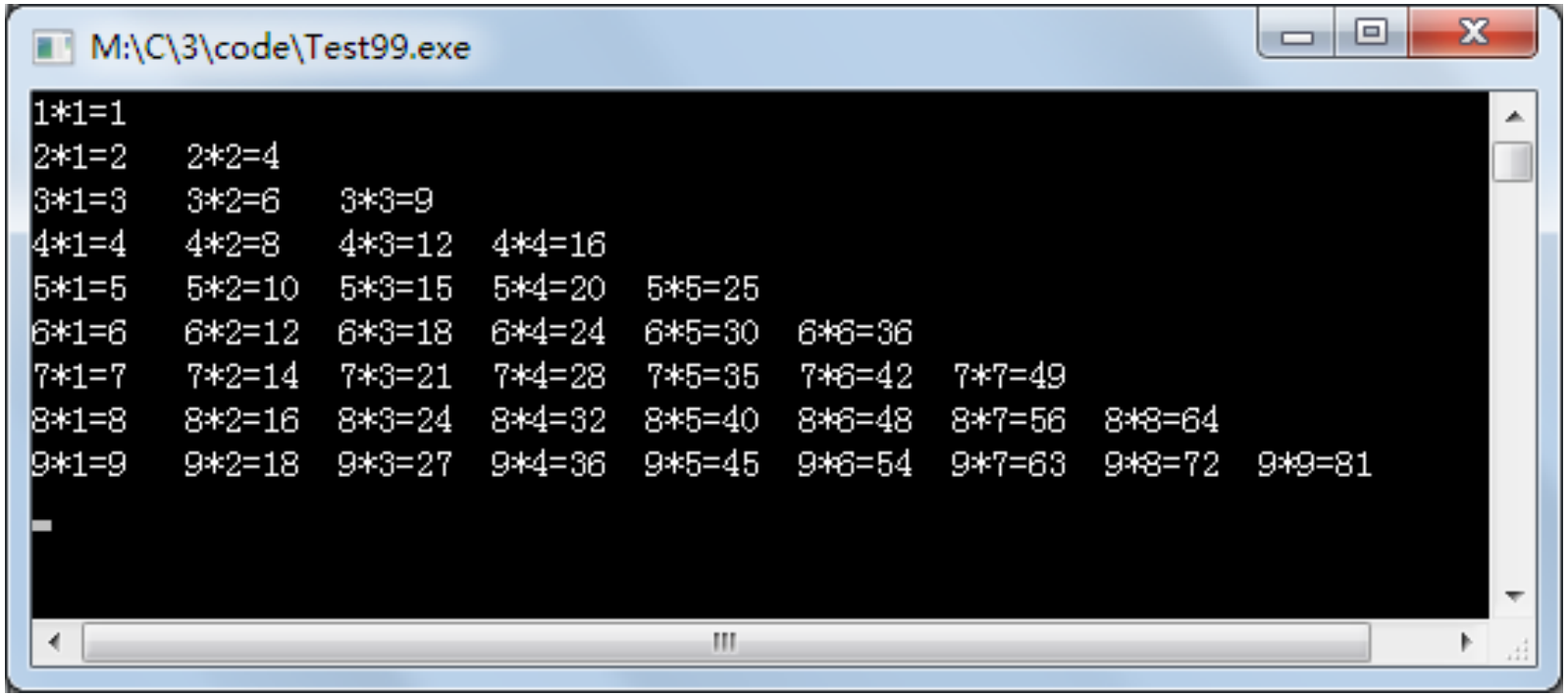
101 103 107 109 113 127 131 137

139 149 151 157 163 167 173 179

181 191 193 197 199

练习

❖ 例 1：编写 C 语言程序，输出乘法九九表。



练习

❖ **例 2：编写 C 程序，实现 100 元人民币换成 1 元、2 元、 5 元的所有兑换方案。**

设 i 、 j 、 k 分别代表 1 元、 2 元和 5 元的数量
则： i+2j+5k=100

```
#include<stdio.h>
int main()
{
  int i,j,k;
  for(i=0;i<=100;i++)
    for(j=0;j<=50;j++)
      for(k=0;k<=20;k++)
        if(i+2*j+5*k==100)
                printf("%d %d %d\n",i,
j,k);
  return 0;
}
```

总结

- 循环结构
  - while 循环
    - 一般形式
    - 注意问题
      - 条件为真执行循环
      - 多个语句应使用大括号
      - 注意循环条件，避免死循环
      - 可以由 while 语句构成多重循环
  - do-while 循环
    - 一般形式
      - do-while 语句后面要加分号
      - 可以和 while 相互嵌套
    - 注意问题
      - 循环体有多个语句也要用大括号
      - 与 while 循环替换时注意修改条件
  - for 循环
    - 书写形式和执行过程
    - 书写格式注意问题
      - 表达式 1，3 可以替换
      - 表达式 2 可以省略，但循环不停止
      - 表达式可以为任意型，执行顺序不变
  - continue 和 break
    - 作用：终止循环
    - 区别：结束本次循环和结束整个循环