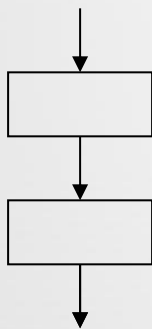
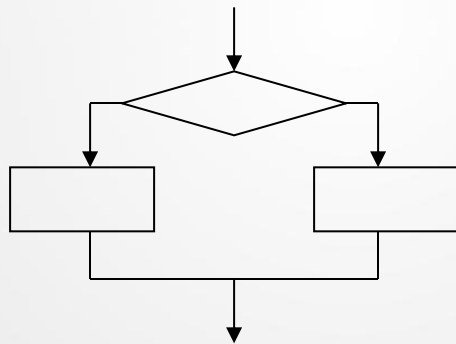


3. 流程控制语句

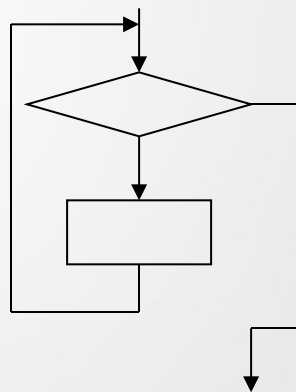
- 流程控制语句是用来控制程序中各语句执行顺序的语句，可以把语句组合成能完成一定功能的小逻辑模块。
- 其流程控制方式采用结构化程序设计中规定的三种基本流程结构，即：顺序结构、分支结构和循环结构，如下图所示：



顺序



分支



循环

本章概述

- 顺序结构
- 选择结构
 - if-else switch
- 循环结构
 - while do-while for
- 跳转
 - break continue return
- 多重循环
- 方法
 - 定义、调用、重载
- 递归算法



本章技能点列表

技能点名称	难易程度	认知程度	重要程度
if语句	中	应用	***
switch语句	中	应用	***
while语句	难	应用	***
do-while语句	难	应用	***
for语句	难	应用	***
跳转语句 break	中	应用	**
跳转语句 continue	中	应用	***
跳转语句 return	中	应用	**



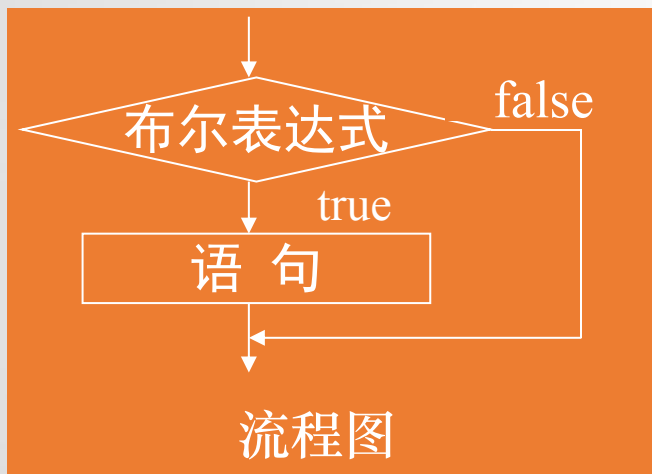
本章技能点列表

技能点名称	难易程度	认知程度	重要程度
多重循环	难	应用	***
多重循环下跳转语句	难	应用	**
方法定义和调用	难	理解	***
方法重载	中	理解	***



if分支选择结构

- if语句对条件表达式进行一次测试，若测试为真，则执行下面的语句，否则跳过该语句



附：

Math类的使用：

```
int i = (int) (6 * Math.random());  
//产生： [0,5]  
//如何产生： 10-15随机数？
```



if 语句

```
double i = 6 * Math.random();
double j = 6 * Math.random();
double k = 6 * Math.random();
int count = (int) (i + j + k);
if(count > 15 ) {
    System.out.println("今天手气不错");
}
if(count >= 10 && count <= 15) { //错误写法: 10<count<15
    System.out.println("今天手气很一般");
}
if(count < 10) {
    System.out.println("今天手气不怎么样");
}
System.out.println("得了" + count + "分"); 要求必须是布尔表达式
```

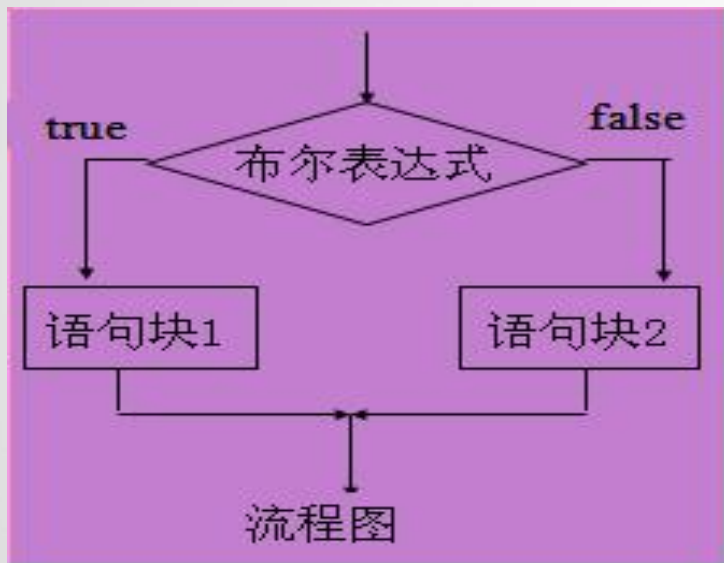
条件结果必须是布尔值

建议都加上花括号。 如果不加花括号, 则只对第一句话有效!



if-else双分支选择结构

- 当条件表达式为真时，执行语句块1，否则，执行语句块2。也就是else部分



```
double r = 4 * Math.random();
double area = Math.PI * Math.pow(r, 2);
double circle = 2 * Math.PI * r;
System.out.println("半径为: " + r);
System.out.println("面积为: " + area);
System.out.println("周长为: " + circle);
if(area >= circle) {
    System.out.println("面积大于等于周长");
} else {
    System.out.println("周长大于面积");
}
```




if-else if-else多分支选择结构

- if(布尔表达式1) {
 - 语句块1;
- } else if(布尔表达式2) {
 - 语句块2;
- }.....
- else if(布尔表达式n){
 - 语句块n;
- } else {
 - 语句块n+1;
- }
- 逐条if语句进行判断
 - 条件匹配, 进入语句体
 - 否则对if语句继续匹配

```
int age = (int) (100 * Math.random());
System.out.print("年龄是" + age + ", 属于");
if (age < 15) {
    System.out.println("儿童, 喜欢玩! ");
} else if (age < 25) {
    System.out.println("青年, 要学习! ");
} else if (age < 45) {
    System.out.println("中年, 要工作! ");
} else if (age < 65) {
    System.out.println("中老年, 要补钙! ");
} else if (age < 85) {
    System.out.println("老年, 多运动! ");
} else {
    System.out.println("老寿星, 古来稀! ");
}
```



switch多分支选择结构

- 根据表达式值的不同执行许多不同的操作

- switch (表达式) {

- case 值1 :

- 语句序列;

- [break];

- case 值2:

- 语句序列;

- [break] ;

-

- [default:

- 默认语句 ;]

- }

1.switch语句会根据表达式的值从相匹配的执行，一直执行到break标签处开始ak语句处或者是switch语句的末尾。与任一case值不匹配，则进入default语句(如果有的话)

2.只能处理等值条件判断的情况，且表达式必须为byte，short，int或char类型，不能是String或double,float.

3.常量值必须是与表达式类型兼容的特定的一个常量

4.不允许有重复的case值

5.default子句为可选



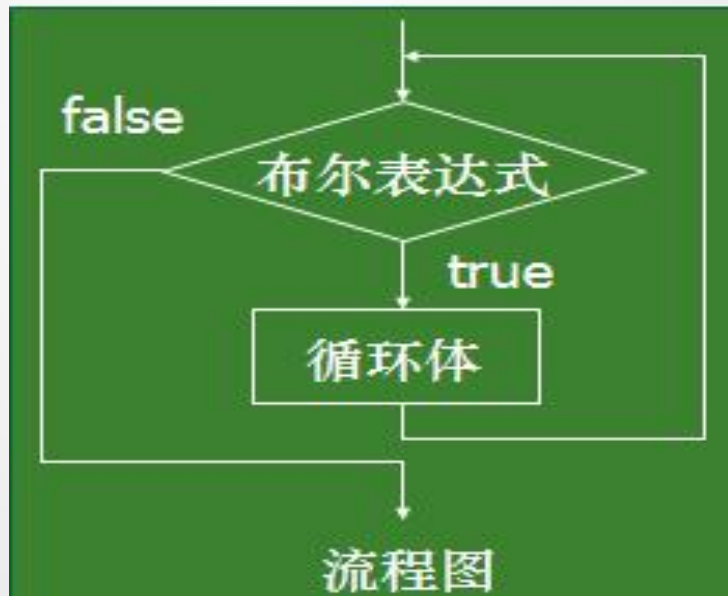
switch多值选择结构

```
char c = 'a';
int rand =(int) (26*Math.random());
char c2 = (char)(c+rand);
System.out.print(c2 + ": ");
switch (c2) {
case 'a':
case 'e':
case 'i':
case 'o':
case 'u':
    System.out.println("元音"); break;
case 'y':
case 'w':
    System.out.println("半元音"); break;
default:
    System.out.println("辅音");
}
```



while循环

- 在循环刚开始时，会计算一次“布尔表达式”的值，若条件为真，执行循环体。而对于后来每一次额外的循环，都会在开始前重新计算一次。
- 语句中应有使循环趋向于结束的语句，否则会出现无限循环——“死”循环。





while循环

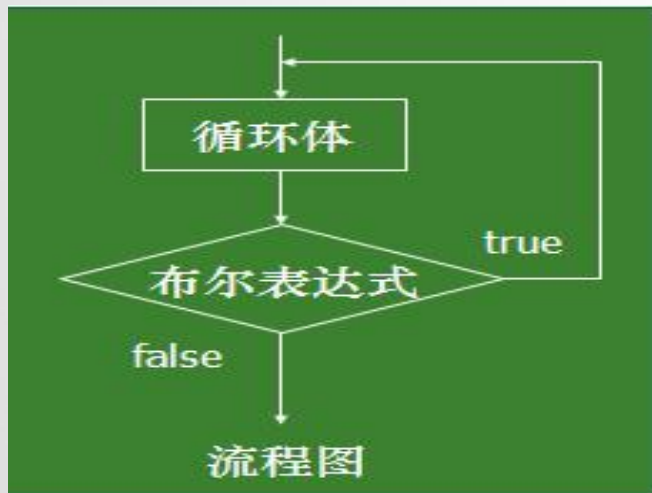
```
public class WhileTest {  
    public static void main(String[] args) {  
        int i = 0;  
        int sum = 0;  
        while (i <= 100) {  
            sum += i; //sum = sum+i;  
            i++;  
        }  
        System.out.println("Sum= " + sum);  
    }  
}
```

循环结构都由如下四个结构组成：
初始化、条件判断、循环体、迭代



do-while循环

- do-while:
 - 先执行，后判断。
- while:
 - 先判断，后执行。



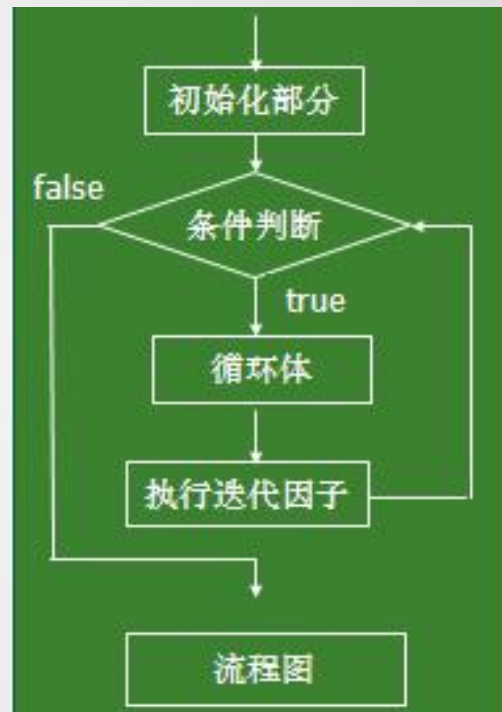
```
int a = 0;
while(a<0){
    System.out.println(a);
    a++;
}
System.out.println("-----");
a=0;
do{
    System.out.println(a);
    a++;
} while (a<0);
```

While和dowhile的区别：
Dowhile总是保证循环体会被至少执行一次！
这是他们的主要差别



for 循环

- for 循环语句是支持迭代的一种通用结构，是最有效、最灵活的循环结构
- 语法形式
 - for (初始表达式;布尔表达式;步进) {
 - 循环体;
 - }
- 注意事项
 - for 循环在执行条件测试后，先执行程序部分，再执行步进。
 - 在for语句的初始化部分声明的变量，其作用域为整个for循环体
 - “初始化”和“循环条件表达式”部分可以使用逗号来执行多个操作
 - 如果三个部分都为空语句（分号不能省），相当于一个无限循环





课堂练习

- 用while和for循环分别计算100以内奇数和偶数的和，并输出。
- 用while和for循环输出1-1000之间能被5整除的数，且每行输出3个。
- 使用循环分别实现将10进值整数和小数变成二进制数



跳转语句---break和continue

- 在任何循环语句的主体部分，均可用break控制循环的流程。break用于强行退出循环，不执行循环中剩余的语句。(break语句还可用于多支语句switch中)
- continue 语句用在循环语句体中，用于终止某次循环过程，即跳过循环体中尚未执行的语句，接着进行下一次是否执行循环的判定。

生成0-100随机数，直到生成88为止，停止循环！

```
int total = 0;
System.out.println("Begin");
while(true) {
    total++;
    int i = (int)Math.round(100 * Math.random());
    if(i == 88) break;
}
System.out.println("Game over, used " + total + " times.");
```

把100~150之间不能被3整除的数输出：

```
for (int i = 100; i < 150; i++) {
    if (i % 3 == 0)
        continue;
    System.out.println(i);
}
```



跳转语句---return

- return语句从当前方法退出，返回到调用该方法的语句处，并从该语句的下条语句处继续执行程序。
- 返回语句的两种格式（具体到方法时详细讲解）
 - 1、return expression
 - 返回一个值给调用该方法的语句。
 - 返回值的数据类型必须和方法声明中的返回值类型一致或是精度低于声明的数据类型。
 - 2、return
 - 当方法声明中用void声明返回类型为空时，应使用这种返回类型，它不返回任何值。



跳转语句总结

- break
 - switch语句
 - 循环语句
- continue
 - 循环语句
- return
 - 任何语句中，结束当前方法，和循环其实没有什么关系



多重循环

- 三种循环方式

- while
- do-while
- for

- 多重循环（循环嵌套）

- 一个循环体内又包含另一个完整的循环结构
- 任何两种循环都可以相互嵌套
- 可以任意层次循环，但是一般不超过3层

- 多重循环执行过程

- 外层循环变量变化一次，内层循环变量要变化一遍

```
for(循环条件1){  
    //循环操作1  
    for(循环条件2){  
        //循环操作2  
    }  
}
```

```
while(循环条件1){  
    //循环操作1  
    for(循环条件2){  
        //循环操作2  
    }  
}
```



多重循环

- 打印矩形
- 打印平行四边形
- 打印等腰三角形
- 打印菱形



多重循环

- 多重循环中使用break
 - 示例
 - 有5家衣服专卖店，每家最多购买3件。用户可以选择离开，可以买衣服。最后打印总共买了几件衣服
 - 思路
 - 外层循环控制去每个专卖店
 - 内层循环控制买衣服过程
 - 使用break退出内层循环
 - 可以使用break直接退出外层循环
 - 使用goto
 - 使用符号量



多重循环

- 带标签的break和continue
 - goto关键字很早就出现在程序设计语言中出现。尽管goto仍是Java的一个保留字，但并未在语言中得到正式使用；Java没有goto。然而，在break和continue这两个关键字的身上，我们仍然能看出一些goto的影子---带标签的break和continue。

```
public class PrimeNumber {  
    public static void main(String args[]) {  
        int count = 0;  
        outer: for (int i = 101; i < 150; i++) {  
            for (int j = 2; j < i / 2; j++) {  
                if (i % j == 0)  
                    continue outer;  
            }  
            System.out.print(i+ " ");  
        }  
    }  
}
```

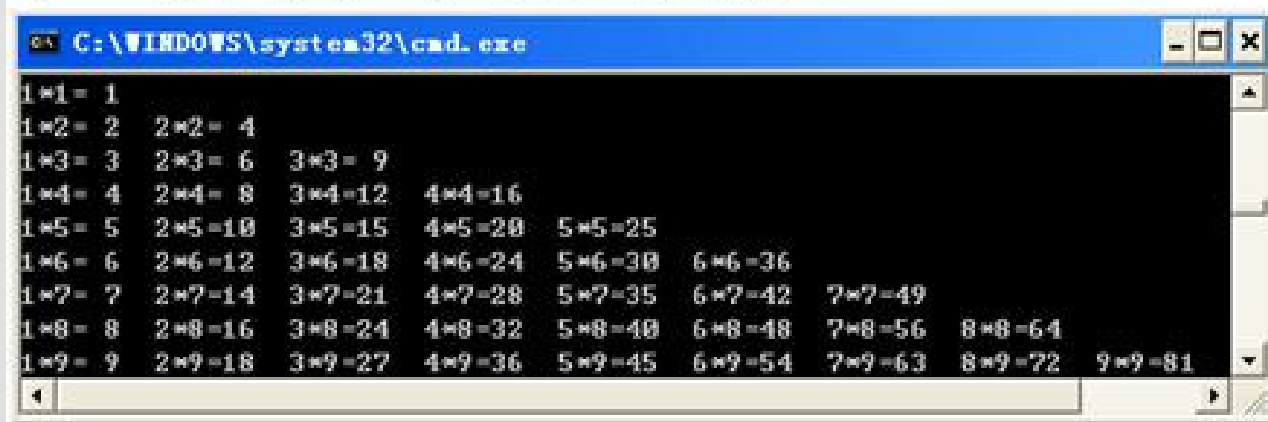
//打印101-150之间所有的质数



课堂作业

- 百钱买白鸡
- 输出九九乘法表：

用for循环输出99乘法表，形式如下：



```
C:\WINDOWS\system32\cmd.exe

1*1= 1
1*2= 2  2*2= 4
1*3= 3  2*3= 6  3*3= 9
1*4= 4  2*4= 8  3*4=12  4*4=16
1*5= 5  2*5=10  3*5=15  4*5=20  5*5=25
1*6= 6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7= 7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8= 8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9= 9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```




方法

- 什么是方法

- 封装在一起来执行操作语句的集合，用来完成某个功能操作

- 在某些语言中称为函数或者过程

- 特殊的方法main，程序执行的入口

- `public static void main(String [] args){`
- 实现功能的语句
- `}`

- 不可能所有的功能都放到main中，需要定义其他方法完成指定功能，需要时调用方法即可



方法

• 定义方法

- [修饰符] 方法返回值类型 方法名(形参列表) {
 - 方法体
 - return 返回值;
- }
- public static int add(int a, int b, int c) {
 - int k = a + b + c;
 - return k;
- }
- 修饰符：封装性时再讲，决定了方法的工作范围
- 返回值类型：必选，如果没有返回值，须写void。方法只能返回一个值
- 方法名：
- 参数列表：可以0个、1个、多个，需要同时说明类型。称为形式参数
- 方法体：完成具体功能。如果有返回值，必须有return语句；如果没有返回值，默认最后一条语句是return，可以省略。



方法

- 方法调用

```
public class MethodTest {
```

```
    public static int add(int a, int b, int c) {  
        int k = a + b + c;  
        return k;  
    }
```

形参列表格式:

数据类型1 形参名1, 数据类型2 形参名2, ...

return 语句用于终止方法的执行并指定要返回的数据,

```
    public static void main(String[] args) {  
        int i = 3, j = 4, k = 5;  
        int result = add(i, j, k);  
    }  
}
```

调用方法调用的形式:
对象引用. 方法名 (实参列表)

实参的数目、数据类型和次序必须和所调用方法声明的形参列表匹配



方法

- 方法重载
 - 一个类中可以定义有相同的名字，但参数不同的多个方法
 - 调用时，会根据不同的参数表选择对应的方法。
- 判断依据
 - 同一个类
 - 同一个方法名
 - 不同：参数列表不同(类型，个数，顺序不同)
- 注意
 - 只有返回值不同不构成方法的重载（
 - `int a(String str){}`,
 - `void a{String i}`,
 - 调用：`a()`， 谁能告诉我是调哪个方法？
 - 只有形参的名称不同，不构成方法的重载



总结

- 选择结构

- if语句 单、双、多分支选择结构，等值、不等值判断均可
- switch语句 只有多分支选择结构 只针对等值判断

- 循环结构

- while循环 先判断再循环 适合循环次数不固定情况
- do-while循环 先循环再判断 适合循环次数不固定情况
- for循环 适合循环次数固定情况

- 循环跳转语句

- break 跳出本层循环，跳出外层循环需要结合标签或符号位实现
- continue 提前结束本次循环
- return 结束当前方法



总结

- 多重循环
 - 任何两种循环都可以相互嵌套
 - 外层循环变量变化一次，内层循环变量要变化一遍
- 方法
 - 定义方法指定形参，调用方法指定实参
 - 方法调用是要求形参实参要求个数相同，类型匹配
 - 方法重载：方法名相同，参数不同