# 流程概述

**概念**：流程是指程序步骤执行的先后顺序，先做什么，后做什么。

**分类**：

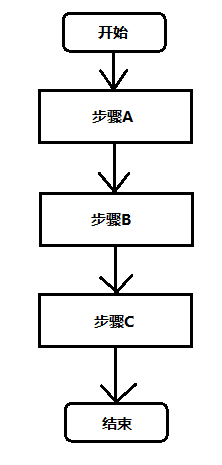
1. 顺序结构：从上到下，从前向后，顺序执行。
2. 选择结构：执行路线分叉，做这个，或者做那个，也叫分支结构。
3. 循环结构：重复做一些事情。

# 顺序结构

程序中最简单最基本的流程控制，没有特定的语法结构，按照代码的先后顺序，依次执行，程序中大多数的代码都是这样执行的。

总地来说：写在前面的先执行，写在后面的后执行。

流程图：



# 选择结构

Java中主要通过if语句来实现选择分支流程，if语句的格式有常用的三种：

* 单if语句：执行某些步骤，或者不执行它们。
* 标准的if-else语句：在2套步骤方案中选择一种执行。
* 扩展的if-else语句：在3套或更多步骤方案中选择一种执行。

## 单if语句

效果：根据条件判断，成立则执行一些代码，不成立则不执行这些代码。

格式：

**if (布尔表达式) {**

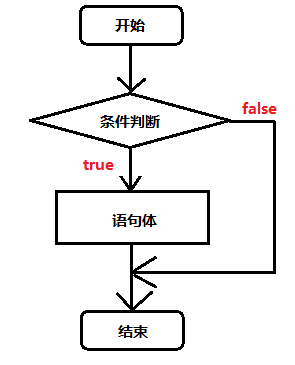
**语句体**

**}**

执行流程：

1. 首先判断小括号之内的条件是true还是false。
2. 如果是**true**，则**执行**大括号之内的语句体，即一些代码。
3. 如果是**false**，则**不执行**大括号之内的语句体。

流程图：



## 标准的if-else语句

效果：根据条件判断，成立则执行if中的代码，不成立则执行else中的代码。

格式：

**if (布尔表达式) {**

**语句体A**

**} else {**

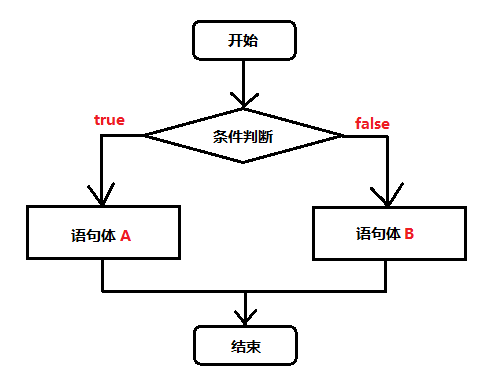
**语句体B**

**}**

执行流程：

1. 首先判断小括号之内的条件是true还是false。
2. 如果是**true**，那么执行if大括号内的**语句体A**，而else的内容将不再执行。
3. 如果是**false**，那么执行else大括号内的**语句体B**，而if的内容将不再执行。

流程图：



## 扩展的if-else语句

作用：在N个语句体当中选择其中的一个执行。

格式：

**if (条件判断1) {**

**语句体1**

**} else if (条件判断2) {**

**语句体2**

**}**

**...**

**else if (条件判断N) {**

**语句体N**

**} else {**

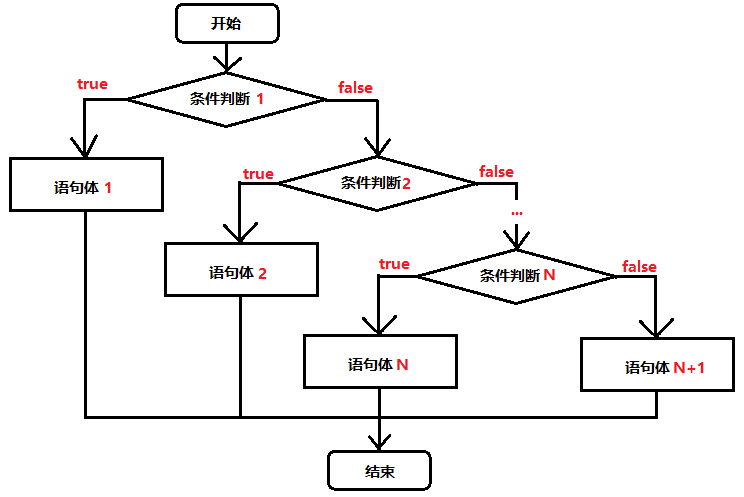
**语句体N+1**

**}**

执行流程：

1. 如果条件判断1成立，执行语句体1，后续代码不再执行。
2. 如果条件判断1不成立，跳过语句体1，继续执行条件判断2。
3. 如果条件判断2成立，执行语句体2，后续代码不再执行。
4. 如果条件判断2不成立，跳过语句体2，继续执行条件判断N。
5. 如果条件判断N成立，执行语句体N，后续代码不再执行。
6. 如果条件判断N不成立，跳过语句体N，执行else中的代码。

流程图：



## 练习1：考试成绩划分

提示：总共有5种情况，5者选其一。成绩划分的标准为：

* 优秀：90-100分
* 良好：80-89分
* 中等：70-79分
* 及格：60-69分
* 不及格：0-59分

## 练习2：求两者最大值

提示：有两个数字a和b，要么a大，要么一定是b大，所以应该使用标准的if-else语句，二者选其一。对于a和b一样大的情况，可以归并为两种情况中的任何一种。

## 练习3：求三者最大值

提示：已经掌握了两者最大值，则可以将三个数字的大小比较，拆分为两两比较。

# 循环结构

## 组成部分

一个循环结构的典型组成部分有：

1. 初始化语句（准备工作）。这部分内容最先执行，而且仅执行一次。
2. 条件判断（布尔结果）。如果成立，则循环继续；如果不成立，则循环退出。
3. 循环体（重复的内容）。每次循环都将重复执行循环体的代码内容。
4. 步进语句（扫尾工作）。每次循环体执行后，都会执行一次步进语句。

## for循环

通常用于控制次数的场景，格式为：

**for (①初始化语句; ②条件判断; ④步进语句) {**

**③循环体**

**}**

执行顺序： ①②③④=>②③④=>②③④……直到②不满足为止。

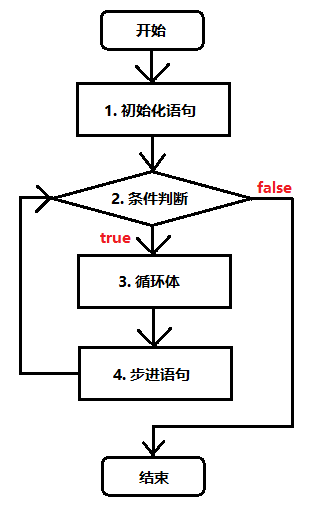
①负责完成循环变量初始化

②负责判断是否满足循环条件，不满足则跳出循环

③具体执行的语句

④循环后，循环条件所涉及变量的变化情况

流程图：



## 练习：求出1-100的偶数和

使用for循环，从1一直到100逐一判断，如果是偶数，则累加起来。最终输出结果。

## while循环

通常用于控制条件的场景，格式有两种。

### 标准格式

**while (①条件判断) {**

**②循环体**

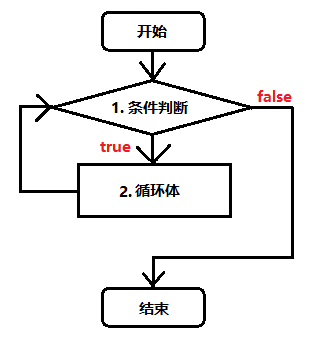
**}**

执行顺序： ①②=>①②=>①②……直到①不满足为止。

①负责判断是否满足循环条件，不满足则跳出循环

②具体执行的语句

流程图：



### 扩展格式

**①初始化语句**

**while (②条件判断) {**

**③循环体**

**④步进语句**

**}**

执行顺序： ①②③④=>②③④=>②③④……直到②不满足为止。

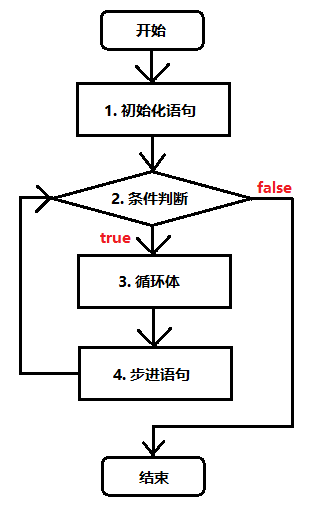
①负责完成循环变量初始化

②负责判断是否满足循环条件，不满足则跳出循环

③具体执行的语句

④循环后，循环条件所涉及变量的变化情况

流程图：



## 练习：求出1-100的奇数和

使用while循环，从1一直到100逐一判断，如果是偶数，则累加起来。最终输出结果。

## for与while的区别

大多数时候，for循环与while循环可以等效替换。但是二者存在几点区别：

1. for循环格式固定，控制次数更方便；而while循环格式灵活，不太在意循环次数。所以次数确定使用for较多；次数不确定，使用while较多。
2. for循环小括号内定义的变量，循环内可用，循环外不可用；while循环的初始化表达式本来就在外面，仍然可用。
3. 跳转语句continue的效果不同。（稍后介绍）

## 跳转控制语句

循环当中可以使用两种跳转控制语句：

* break语句
* continue语句

### break语句

如果希望当前循环立刻结束，可以使用break语句：

**break;**

一旦执行这个语句，整个循环立刻结束。

### continue语句

如果希望循环只是跳过当前次，后续还要继续，那么可以使用continue语句：

**continue;**

一旦执行这个语句，当前次循环剩余内容立刻跳过，马上开始下一次循环。

### break与continue的区别

* break会将整个循环彻底打断，剩余多少次全都没有了；
* continue只是跳过当前次，剩余的更多次仍然还有。

## 死循环

概念：永远无法退出的循环，叫做死循环。

如果希望手写一个死循环，那么可以使用格式：

**标准格式：**

**while (true) {**

**循环体**

**}**

还有另外一种等效的格式：

**for (;;) {**

**循环体**

**}**

PS：死循环当中仍然可以根据实际需要使用break或continue语句。

## 循环嵌套

可以在一个循环的内部，套用又一个循环，行程内外层嵌套的结构。

在循环嵌套中，外层循环的**每一次**执行，都对应这内层循环的**完整多次**执行。

## 练习：打印一天的分钟时刻

提示：外层控制小时，内层控制分钟。

## 循环嵌套中的跳转语句

break或continue只能作用于**当前所属层**循环。如果希望作用于外层循环，可以使用标签的写法：

**标签名称: for (...) {** // 外层循环

**for (...) {** // 内层循环

**break 标签名称;**

**}**

**}**

# 知识总结

1. 流程：也就是代码的若干个步骤，先做什么，后做什么。
2. 流程分类：
   1. 顺序结构
   2. 选择结构
   3. 循环结构
3. 选择结构，if语句的格式三种：
   1. 单if语句：执行代码，或者不执行；有或无
   2. 标准的if-else语句：执行A或者执行B，二者选其一
   3. 扩展的if-else语句：N者选其一
4. 循环结构：
   1. for循环：格式比较死，擅长控制次数
   2. while循环：格式比较灵活，不太在意次数。
      1. 标准格式
      2. 扩展格式
5. 跳转控制语句：
   1. break：一旦执行，整个循环立刻结束
   2. continue：一旦执行，当前单次循环立刻结束，马上开始下一次。
6. 循环嵌套。