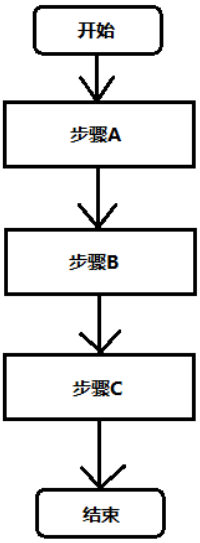
Java Day 04

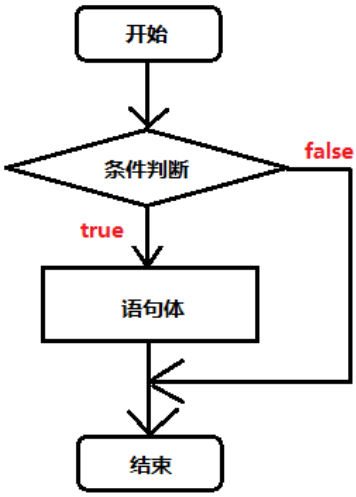
顺序结构：从上到下顺序执行



条件语句

if

```
if(关系表达式) {
    语句体;
}
```



执行流程

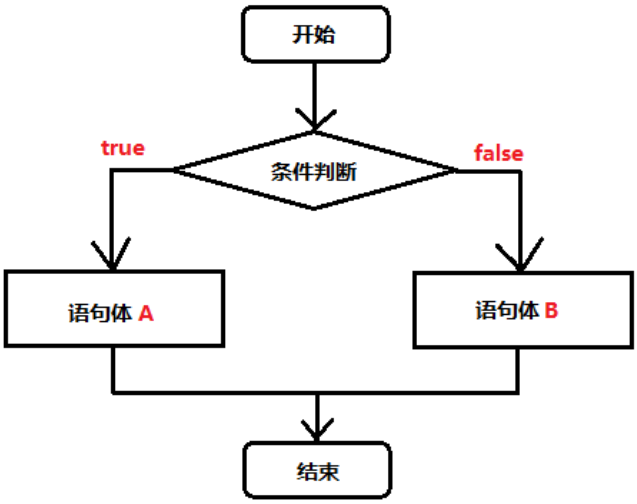
首先判断关系表达式看其结果是 true 还是 false

如果是 true 就执行语句体

如果是 false 就不执行语句体

if…else…

```
if(关系表达式) {
    语句体 1;
}else {
    语句体 2;
}
```



执行流程

首先判断关系表达式看其结果是 true 还是 false

如果是 true 就执行语句体 1

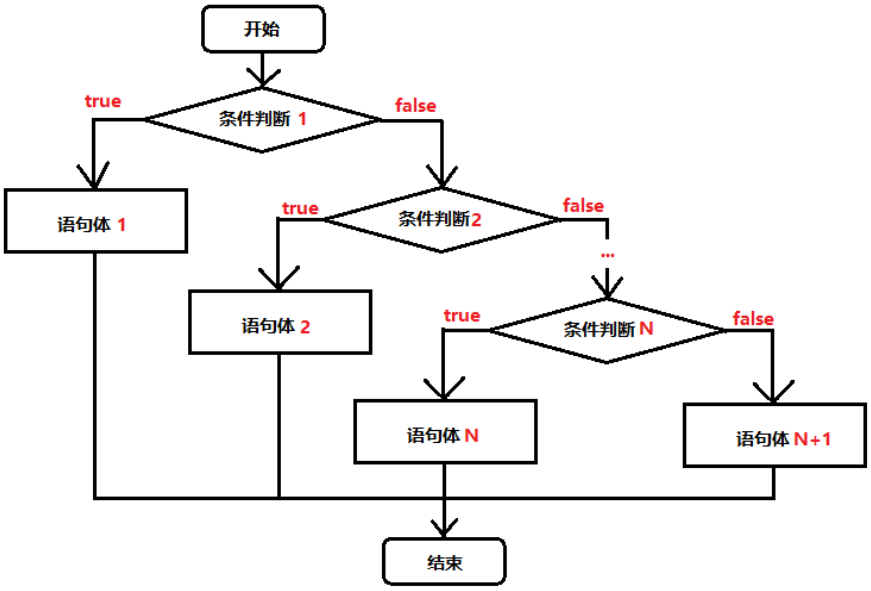
如果是 false 就执行语句体 2

if…else if…else

```
if (判断条件 1) {
    执行语句 1;
} else if (判断条件 2) {
    执行语句 2;
}
...
}else if (判断条件 n) {
    执行语句 n;
} else {
```

执行语句 n+1;

}



执行流程

首先判断关系表达式 1 看其结果是 true 还是 false

如果是 true 就执行语句体 1

如果是 false 就继续判断关系表达式 2 看其结果是 true 还是 false

如果是 true 就执行语句体 2

如果是 false 就继续判断关系表达式...看其结果是 true 还是 false

...

如果没有任何关系表达式为 true，就执行语句体 n+1。

注意：

1）如果体部就一条语句，花括号可以省略（一般不建议省略）

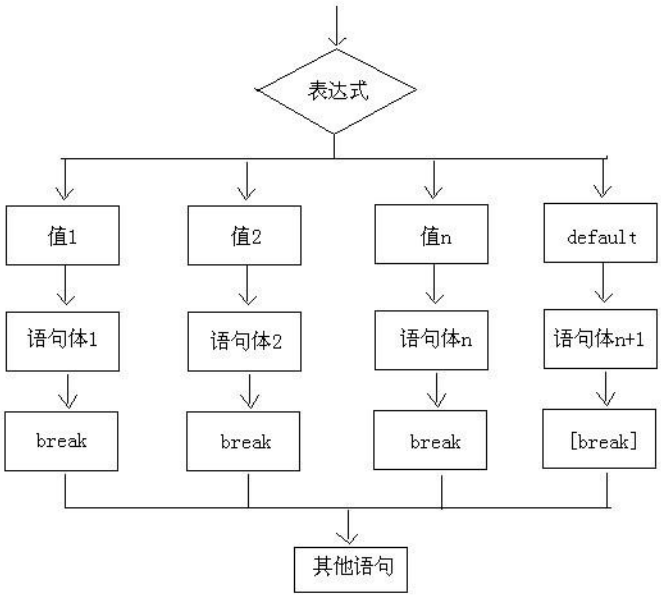
2）如果多个条件分支都满足，只有第一个会被执行

分支语句

---

switch

```
switch(表达式) {

case 常量值 1:

    语句体 1;

    break;

  case 常量值 2:

    语句体 2;

    break;

  ...

  default:

    语句体 n+1;

    break;
}
```



执行流程

首先计算出表达式的值

其次，和 case 依次比较，一旦有对应的值，就会执行相应的语句，在执行的过程中，遇到 break 就会结束。

最后，如果所有的 case 都和表达式的值不匹配，就会执行 default 语句体部分，然后程序结束。

注意

1）表达式的数据类型，可以是 byte，short，int，char，enum（枚举），JDK7 后可以接收字符串。

2）如果 case 的后面不写 break，将出现穿透现象，也就是不会再判断下一个 case 的值，直接向后运行，直到遇到 break，或者整体 switch 结束。

3)default默认匹配分支,当所有case分支不匹配时执行;

default 分支可以放到 switch 语句的任意位置（建议放在末尾)；default 是可选的。

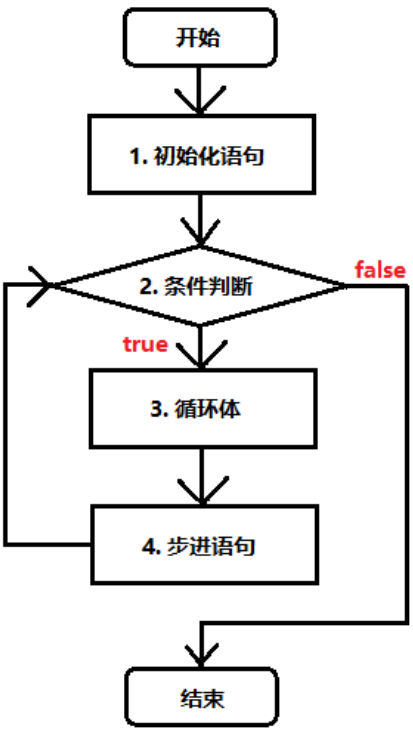
4）可以多个分支匹配一个操作

循环语句

---

for

```
for(初始化表达式①; 布尔表达式②; 步进表达式④){
        循环体③
}
```



执行流程

执行顺序：①②③④>②③④>②③④…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

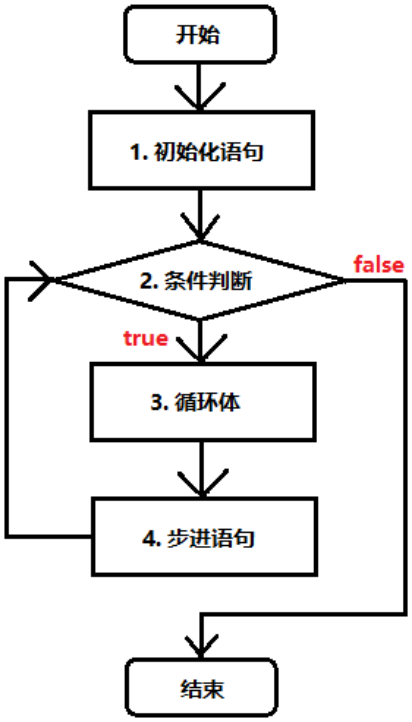
③ 具体执行的语句

④ 循环后，循环条件所涉及变量的变化情况

while

初始化表达式①

```
while(布尔表达式②){
    循环体③
    步进表达式④
}
```



执行流程

执行顺序：①②③④>②③④>②③④…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。

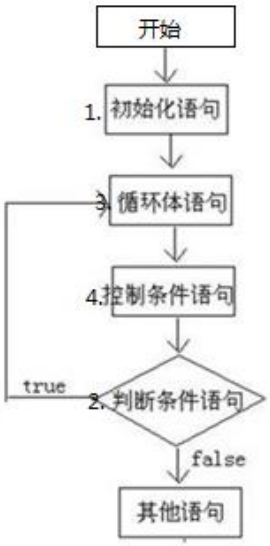
③ 具体执行的语句。

④ 循环后，循环变量的变化情况。

do…while

初始化表达式①

```
do{
    循环体③

    步进表达式④

}while(布尔表达式②);
```



执行流程

执行顺序：①③④>②③④>②③④…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，循环变量的变化情况

for、while 区别

1）控制条件语句所控制的那个变量，在 for 循环结束后，就不能再被访问到了，而 while 循环结束还可以继续使用，如果你想继续使用，就用 while，否则推荐使用 for。原因是 for 循环结束，该变量就从内存中消失，能够提高内存的使用效率。

2）在已知循环次数的时候使用推荐使用 for，循环次数未知的时推荐使用 while。

break、continue

break[label]：终止 switch 或者循环

continue[label]：结束本次循环，继续下一次的循环

Java 中的标签是为循环设计的，是为了在多重循环中方便地使用 break 和 continue 跳到标签处。

死循环

也就是循环中的条件永远为 true，死循环的是永不结束的循环。

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