Java程 序 设 计

第 5 讲: 循环结构与算法控制 (for, while 与转向控制)

厦门大学嘉庚学院

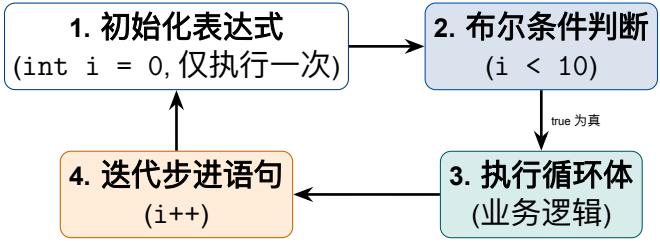
XIAMEN UNIVERSITY TAN KAH KEE COLLEGE

CONTENTS

三大循环结构语法与执行时序

循环语句	执行时序特征	最适用业务场景
`for` 循环	循环前先判断条件，执行后步进	**已知循环次数**（如遍历数组）
`while` 循环	循环前先判断条件，可能一次都不执行	**条件驱动/事件监听**（如网络等待）
`do-while` 循环	**先执行一次循环体**，随后再判断条件	**至少执行一次**（如控制台输入重试）

for 循环执行四步曲


1. 初始化表达式
(int i = 0, 仅执行一次)
2. 布尔条件判断
(i < 10)
true 为真
3. 执行循环体
(业务逻辑)
4. 迭代步进语句
(i++)

while 与 do-while 循环结构演练

while 循环 (先判后跑)

```
int retry = 0;
while (retry < 3) {
    if (connect()) break;
    retry++;
}
```

do-while 循环 (先跑一次)

```
int choice;
do {
    showMenu();
    choice = sc.nextInt();
} while (choice != 0);
```

随堂概念抢答 1：循环执行次数判定

【随堂抢答·概念检测】问题：以下代码执行完毕后，控制台共输出多少次”Java”？

```java
int count = 0;
do {
    System.out.println("Java");
    count++;
} while (count < 0);
```

A. 0 次

B. 1 次

C. 无限次（死循环）

D. 编译报错

随堂概念抢答 1:答案与深度解析

【知识要点】正确答案:【B 】

原理解析:

- do-while 的核心特征:**无论判定条件真假,循环体代码至少无条件执行 1 次**;
- 执行完一次打印后,count 变为 1,随后判断 count < 0 为 false,循环终止;
- 最终精确输出 1 次!

break vs continue 核心机制对比

break (强行跳出)

- 立即无条件**彻底终止当前所在的整层循环**;
- 跳出循环体,执行循环后的下一条语句。

continue (提前步进)

- 仅提前**终止当前这一轮迭代**;
- 跳过本轮剩余语句,直接进入下一次循环条件判定与步进。

【代码演练】标签跳出外层双重循环

```java
outerLoop: // 定义外层循环标签
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 5; j++) {
        if (i * j > 10) {
            System.out.printf("触发阈值: %d * %d\n", i, j);
            break outerLoop; // 直接强行跳出外层 outerLoop 循环!
        }
    }
}
```

随堂概念抢答 2：break 与 continue 轨迹追踪

【随堂抢答·概念检测】问题：执行以下代码后，控制台输出什么内容？

```java
for (int i = 1; i <= 5; i++) {
    if (i == 2) continue;
    if (i == 4) break;
    System.out.print(i + " ");
}
```

A. 1 2 3

B. 1 3

C. 1 3 4

D. 1 3 5

随堂概念抢答 2:答案与深度解析

【知识要点】正确答案:【B 】

原理解析:

- $i = 1$:正常打印 `"1 "`;
- $i = 2$:触发 `continue`,直接跳过打印进入下一轮;
- $i = 3$:正常打印 `"3 "`;
- $i = 4$:触发 `break`,彻底终止整层循环,循环立即结束!
- 最终输出:`"1 3 "`。

经典算法 1：九九乘法表双重循环

【代码演练】格式化九九乘法表 (MultiplicationTable.java)

```java
public class MultiplicationTable {
    public static void main(String[] args) {
        for (int i = 1; i <= 9; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.printf("%d*%d=%-2d  ", j, i, (i * j));
            }
            System.out.println(); // 换行
        }
    }
}
```

经典算法 2：素数筛选与 Math.sqrt 剪枝

【代码演练】素数判定与剪枝优化 (PrimeSieveDemo.java)

```java
public static boolean isPrime(int n) {
    if (n <= 1) return false;
    // 关键优化：因数成对出现，仅需遍历到 sqrt(n)
    int limit = (int) Math.sqrt(n);
    for (int i = 2; i <= limit; i++) {
        if (n % i == 0) return false; // 存在因子，提前返回
    }
    return true;
}
```

随堂找茬 1:浮点数作为循环计数器陷阱

【随堂找茬·避坑挑错】看看以下代码为什么会导致死循环?

```
// 致命错误: 浮点数存在二进制精度截断, d 极可能永远不严格等于 1.0!
for (double d = 0.0; d != 1.0; d += 0.1) {
    System.out.println("数据采样中: " + d);
}
```

【知识要点】避坑指南

严禁使用浮点数作为循环计数器或等值判断条件! 循环计数器一律使用 int 或 long 整型!

随堂找茬 2：while 循环漏写步进语句

【随堂找茬·避坑挑错】看看以下代码为什么会卡死无法退出？

```java
int i = 0;
while (i < 10) {
    System.out.println("处理中...");
    // 致命错误：忘记书写 i++ 步进语句，导致 i 永远为 0，陷入死循环！
}
```

随堂实操练笔 1：控制台猜数字互动小游戏

【随堂实操·动手练笔】任务 1：编写 NumberGuessGame.java

编写 `NumberGuessGame.java`，实现：

1. 生成一个 $1 \sim 100$ 的随机目标数字；
2. 使用 `while` 循环引导用户输入猜测，提示“猜大了”或“猜小了”；
3. 猜中后统计并输出总共尝试的猜测次数。

随堂实操练笔 2：区间素数统计分析器

【随堂实操·动手练笔】任务 2：编写 PrimeSieveDemo.java

编写 `PrimeSieveDemo.java`，实现：

1. 找出 $1 \sim 1000$ 之间的所有素数；
2. 每行格式化输出 10 个素数；
3. 统计素数总个数并计算素数密度。

工程视野与系统防死锁

【工程视野】死循环防范与看门狗机制

在嵌入式与通信服务器中死循环会导致瘫痪。设计最大循环次数与看门狗(Watchdog)熔断，是保障系统可用性的底线。

【实战技巧】代码审查与辅助开发

实战建议：利用 AI 对多重嵌套循环进行复杂度评估，检测冗余计算与潜在死循环隐患。

【随堂实操与课后思考】课后思考与巩固

- **练习**：使用嵌套循环打印倒直角三角形与对称菱形图案。
- **思考**：通信网络轮询中，`while(true)` 须配合何种机制避免 CPU 占满 100%?

for 循环省略表达式的特殊形态

【知识要点】死循环标准写法与变种

- `for (;;) { ... }`：省略全部三个表达式，等价于 `while(true);`
- 省略初始化表达式：在外部提前声明循环变量；
- 省略步进表达式：在循环体内部按需手动调整步进。

随堂抢答 3：无限死循环写法辨析

【随堂抢答·概念检测】问题：以下 4 种循环写法中，哪一种不是标准的无限循环？

A. `for (;;)`

B. `while (true)`

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

D. `for (int i = 0; i < 10; i--)` (假设 i 不会发生下溢)

随堂抢答 3：答案与深度解析

【知识要点】正确答案：【D】

原理解析：

- 选项 A、B、C 均为标准语法的无条件死循环；
- 选项 D 中 `i` 递减到 `Integer.MIN_VALUE` 后若发生下溢可能回绕为正数，不是标准死循环写法！

课程互动与讲师主页

在线主页与课程资源

- **个人主页**：
  `https://matnoble.top`
- **课程专栏**：
  `https://matnoble.top/courses/`
- **GitHub 仓库**：
  `https://github.com/MatNoble`

思考：Questions & Answers

“学贵善疑，逻辑筑基”

感谢大家的专注聆听！

课件最新版本与补充资料将持续同步至课程主页。

现在进入自由提问与交流时间。