

第一章 流程控制语句

在一个程序执行的过程中，各条语句的执行顺序对程序的结果是有直接影响的。所以，我们必须清楚每条语句的执行流程。而且，很多时候要通过控制语句的执行顺序来实现我们想要的功能。

1.1 流程控制语句分类

顺序结构

判断和选择结构(if, switch)

循环结构(for, while, do...while)

1.2 顺序结构

顺序结构是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。

顺序结构执行流程图：

1545615769372

第二章 判断语句：if语句

2.1 if语句格式1

格式：

```
if (关系表达式) {  
    语句体;  
}
```

执行流程：

- ①首先计算关系表达式的值
- ②如果关系表达式的值为true就执行语句体
- ③如果关系表达式的值为false就不执行语句体
- ④继续执行后面的语句内容

1545616039363

示例：

```
public class IfDemo {  
    public static void main(String[] args) {  
        System.out.println("开始");  
        //定义两个变量  
  
        int a = 10;
```



```
//需求：判断a和b的值是否相等，如果相等，就在控制台输出：a等于b
if(a == b) {
    System.out.println("a等于b");
}
//需求：判断a和c的值是否相等，如果相等，就在控制台输出：a等于c
int c = 10;
if(a == c) {
    System.out.println("a等于c");
}
System.out.println("结束");
}
```

练习1：老丈人选女婿

需求：

键盘录入女婿的酒量，如果大于2斤，老丈人给出回应，否则没有任何回应

代码示例：

```
//分析：
//1.键盘录入女婿的酒量
Scanner sc = new Scanner(System.in);
System.out.println("请输入女婿的酒量");
int wine = sc.nextInt();//5
//2.对酒量进行一个判断即可
if(wine > 2) {
    System.out.println("不错哟，小伙子！");
}
```

练习2：考试奖励

需求：

键盘录入一个整数，表示小明的考试名次，如果名次为1，小红可以当小明的女朋有了。

代码示例：

```
//分析：
//1.键盘录入一个整数，表示小明的考试名次
Scanner sc = new Scanner(System.in);
System.out.println("请输入小明的名次");
int rank = sc.nextInt();
//2.对小明的考试成绩进行判断即可
if(rank == 1){
    System.out.println("小红成为了小明的女朋友");
}
```

第一种格式的细节：

2. 如果大括号中的语句体只有一条，那么大括号可以省略不写
如果大括号省略了，那么if只能控制距离他最近的那一条语句。
建议：自己不要去写，如果别人这么写了，你要能看懂即可。

2.2 if语句格式2

格式：

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

执行流程：

- ①首先计算关系表达式的值
- ②如果关系表达式的值为true就执行语句体1
- ③如果关系表达式的值为false就执行语句体2
- ④继续执行后面的语句内容

1545616221283

示例：

```
public class IfDemo02 {  
    public static void main(String[] args) {  
        System.out.println("开始");  
        //定义两个变量  
        int a = 10;  
        int b = 20;  
        //需求：判断a是否大于b，如果是，在控制台输出：a的值大于b，否则，在控制台输出：a的值不大于b  
        if(a > b) {  
            System.out.println("a的值大于b");  
        } else {  
            System.out.println("a的值不大于b");  
        }  
        System.out.println("结束");  
    }  
}
```

练习1：吃饭

需求：

键盘录入一个整数，表示身上的钱。

如果大于等于100块，就是网红餐厅。



代码示例：

```
//分析：
//1.键盘录入一个整数。表示身上的钱。
Scanner sc = new Scanner(System.in);
System.out.println("请输入一个整数表示身上的钱");
int money = sc.nextInt();
//2.对钱进行判断
if(money >= 100){
    System.out.println("吃网红餐厅");
}else{
    System.out.println("福建大酒店");
}
```

练习2：影院选座

需求：

在实际开发中，电影院选座也会使用到if判断。

假设某影院售卖了100张票，票的序号为1~100。

其中奇数票号坐左侧，偶数票号坐右侧。

键盘录入一个整数表示电影票的票号。

根据不同情况，给出不同的提示：

如果票号为奇数，那么打印坐左边。

如果票号为偶数，那么打印坐右边。

代码示例：

```
//分析：
//1.键盘录入票号
Scanner sc = new Scanner(System.in);
System.out.println("请输入票号");
int ticket = sc.nextInt();
if(ticket >= 1 && ticket <= 100){
    //合法
    //2.对票号进行判断
    if (ticket % 2 == 0) {
        //偶数
        System.out.println("坐右边");
    } else {
        //奇数
        System.out.println("坐左边");
    }
}else{
    //票号不合法
    System.out.println("票号不合法");
}
```

2.3 if语句格式3

格式：

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

执行流程：

- ①首先计算关系表达式1的值
- ②如果值为true就执行语句体1；如果值为false就计算关系表达式2的值
- ③如果值为true就执行语句体2；如果值为false就计算关系表达式3的值
- ④...
- ⑤如果没有任何关系表达式为true，就执行语句体n+1。

1545616667104

练习1：考试奖励

需求：

小明快要期末考试了，小明爸爸对他说，会根据他不同的考试成绩，送他不同的礼物，假如你可以控制小明的得分，请用程序实现小明到底该获得什么样的礼物，并在控制台输出。

分析：

- ①小明的考试成绩未知，可以使用键盘录入的方式获取值
- ②由于奖励种类较多，属于多种判断，采用if...else...if格式实现
- ③为每种判断设置对应的条件
- ④为每种判断设置对应的奖励

代码示例：

```
//95~100 自行车一辆  
//90~94 游乐场玩一天  
//80 ~ 89 变形金刚一个  
//80 以下 胖揍一顿  
  
//1.键盘录入一个值表示小明的分数  
  
Scanner sc = new Scanner(System.in);
```

```
int score = sc.nextInt();
//2.对分数的有效性进行判断
if(score >= 0 && score <= 100){
    //有效的分数
    //3.对小明的分数进行判断，不同情况执行不同的代码
    if(score >= 95 && score <= 100){
        System.out.println("送自行车一辆");
    }else if(score >= 90 && score <= 94){
        System.out.println("游乐场玩一天");
    }else if(score >= 80 && score <= 89){
        System.out.println("变形金刚一个");
    }else{
        System.out.println("胖揍一顿");
    }
}else{
    //无效的分数
    System.out.println("分数不合法");
}
```

第三章 switch语句

3.1 格式

```
switch (表达式) {
    case 1:
        语句体1;
        break;
    case 2:
        语句体2;
        break;
    ...
    default:
        语句体n+1;
        break;
}
```

3.2 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，然后程序结束掉。

练习：运动计划

- 需求：键盘录入星期数，显示今天的减肥活动。
周一：跑步
周二：游泳
周三：慢走



周五：拳击

周六：爬山

周日：好好吃一顿

- 代码示例：

```
package a01switch选择语句;

import java.util.Scanner;

public class SwitchDemo2 {
    public static void main(String[] args) {
        //1.键盘录入一个整数表示星期
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入一个整数表示星期");
        int week = sc.nextInt();

        //2.书写一个switch语句去跟week进行匹配
        switch (week){
            case 1:
                System.out.println("跑步");
                break;
            case 2:
                System.out.println("游泳");
                break;
            case 3:
                System.out.println("慢走");
                break;
            case 4:
                System.out.println("动感单车");
                break;
            case 5:
                System.out.println("拳击");
                break;
            case 6:
                System.out.println("爬山");
                break;
            case 7:
                System.out.println("好好吃一顿");
                break;
            default:
                System.out.println("输入错误，没有这个星期");
                break;
        }
    }
}
```

3.3 switch的扩展知识：

- default的位置和省略情况

- case穿透
不写break会引发case穿透现象
- switch在JDK12的新特性

```
int number = 10;
switch (number) {
    case 1 -> System.out.println("一");
    case 2 -> System.out.println("二");
    case 3 -> System.out.println("三");
    default -> System.out.println("其他");
}
```

- switch和if第三种格式各自的使用场景

当我们需要对一个范围进行判断的时候，用if的第三种格式

当我们把有限个数据列举出来，选择其中一个执行的时候，用switch语句

比如：

小明的考试成绩，如果用switch，那么需要写100个case，太麻烦了，所以用if简单。

如果是星期，月份，客服电话中0~9的功能选择就可以用switch

练习：休息日和工作日

需求：键盘录入星期数，输出工作日、休息日。

(1-5) 工作日，(6-7)休息日。

代码示例：

```
//分析：
//1.键盘录入星期数
Scanner sc = new Scanner(System.in);
System.out.println("请输入星期");
int week = sc.nextInt(); //3
//2.利用switch进行匹配

-----
利用case穿透简化代码
switch (week){
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        System.out.println("工作日");
        break;
    case 6:
    case 7:
        System.out.println("休息日");
        break;
```




```
System.out.println("没有这个星期");  
break;  
}  
  
-----  
利用JDK12简化代码书写  
switch (week) {  
    case 1, 2, 3, 4, 5 -> System.out.println("工作日");  
    case 6, 7 -> System.out.println("休息日");  
    default -> System.out.println("没有这个星期");  
}
```

第四章 循环结构

4.1 for循环结构（掌握）

循环语句可以在满足循环条件的情况下，反复执行某一段代码，这段被重复执行的代码被称为循环体语句，当反复执行这个循环体时，需要在合适的时候把循环判断条件修改为false，从而结束循环，否则循环将一直执行下去，形成死循环。

4.1.1 for循环格式：

```
for (初始化语句;条件判断语句;条件控制语句) {  
    循环体语句;  
}
```

格式解释：

- 初始化语句：用于表示循环开启时的起始状态，简单说就是循环开始的时候什么样
- 条件判断语句：用于表示循环反复执行的条件，简单说就是判断循环是否能一直执行下去
- 循环体语句：用于表示循环反复执行的内容，简单说就是循环反复执行的事情
- 条件控制语句：用于表示循环执行中每次变化的内容，简单说就是控制循环是否能执行下去

执行流程：

- ①执行初始化语句
- ②执行条件判断语句，看其结果是true还是false
如果是false，循环结束
如果是true，继续执行
- ③执行循环体语句
- ④执行条件控制语句
- ⑤回到②继续

for循环书写技巧：

- 确定循环的开始条件
- 确定循环的结束条件
- 确定循环要重复执行的代码



```
//1.确定循环的开始条件
//2.确定循环的结束条件
//3.确定要重复执行的代码

//需求：打印5次HelloWorld
//开始条件： 1
//结束条件： 5
//重复代码：打印语句

for (int i = 1; i <= 5; i++) {
    System.out.println("HelloWorld");
}
```

for循环练习-输出数据

- 需求：在控制台输出1-5和5-1的数据
- 示例代码：

```
public class ForTest01 {
    public static void main(String[] args) {
        //需求：输出数据1-5
        for(int i=1; i<=5; i++) {
            System.out.println(i);
        }
        System.out.println("-----");
        //需求：输出数据5-1
        for(int i=5; i>=1; i--) {
            System.out.println(i);
        }
    }
}
```

for循环练习-求和

- 需求：求1-5之间的数据和，并把求和结果在控制台输出
- 示例代码：

```
public class ForTest02 {
    public static void main(String[] args) {
        //求和的最终结果必须保存起来，需要定义一个变量，用于保存求和的结果，初始值为0
        int sum = 0;
        //从1开始到5结束的数据，使用循环结构完成
        for(int i=1; i<=5; i++) {
            //将反复进行的事情写入循环结构内部
            // 此处反复进行的事情是将数据 i 加到用于保存最终求和的变量 sum 中
            sum = sum + i;
        }
    }
}
```



```
        第一次: sum = sum + i = 0 + 1 = 1;
        第二次: sum = sum + i = 1 + 2 = 3;
        第三次: sum = sum + i = 3 + 3 = 6;
        第四次: sum = sum + i = 6 + 4 = 10;
        第五次: sum = sum + i = 10 + 5 = 15;
    */
}
//当循环执行完毕时，将最终数据打印出来
System.out.println("1-5之间的数据和是: " + sum);
}
}
```

- 本题要点：
 - 今后遇到的需求中，如果带有求和二字，请立即联想到求和变量
 - 求和变量的定义位置，必须在循环外部，如果在循环内部则计算出的数据将是错误的

for循环练习-求偶数和

- 需求：求1-100之间的偶数和，并把求和结果在控制台输出 }
- 示例代码：

```
public class ForTest03 {
    public static void main(String[] args) {
        //求和的最终结果必须保存起来，需要定义一个变量，用于保存求和的结果，初始值为0
        int sum = 0;
        //对1-100的数据求和与1-5的数据求和几乎完全一样，仅仅是结束条件不同
        for(int i=1; i<=100; i++) {
            //对1-100的偶数求和，需要对求和操作添加限制条件，判断是否是偶数
            if(i%2 == 0) {
                //sum += i;
                sum = sum + i;
            }
        }
        //当循环执行完毕时，将最终数据打印出来
        System.out.println("1-100之间的偶数和是: " + sum);
    }
}
```

for循环练习-统计次数

需求：

键盘录入两个数字，表示一个范围。

统计这个范围中。

既能被3整除，又能被5整除数字有多少个？

代码示例：



4.2.1 格式：

```
初始化语句;
while(条件判断语句){
    循环体;
    条件控制语句;
}
```

练习1：打印5次HelloWorld

```
int i = 1;
while(i <= 5){
    System.out.println("HelloWorld");
    i++;
}
System.out.println(i);
```

练习2：珠穆朗玛峰

```
//1.定义一个变量表示珠穆朗玛峰的高度
int height = 8844430;
//2.定义一个变量表示纸张的厚度
double paper = 0.1;

//定义一个计数器（变量），用来统计折叠的次数
int count = 0;

//3.循环折叠纸张
//只有纸张的厚度 < 穆朗玛峰的高度 循环才继续，否则循环就停止
//坑：只有判断为真，循环才会继续
while(paper < height){
    //折叠纸张
    paper = paper * 2;
    count++;
}

//4.打印一下纸张的厚度
System.out.println(count);//27
```

4.3 do...while循环

本知识点了解即可

格式：



```
初始化语句;  
do{  
    循环体;  
    条件控制语句;  
}while(条件判断语句);
```

特点:

先执行，再判断。

4.4 三种格式的区别:

for和while循环，是先判断，再执行。

do...while是先执行，再判断。

当知道循环次数或者循环范围的时候，用for循环。

当不知道循环次数，也不知道循环范围，但是知道循环的结束条件时，用while循环。