第一章 流程控制

在程序执行的过程中，每条语句的执行顺序对程序的结果都有着直接的影响，所以，我们必须清楚每条语句的执行流程，并且，通过控制语句的执行顺序来实现功能。

顺序结构

顺序执行，更具编写的顺序，从上到下运行

```
public static void main(String[] args) {
    System.out.println(1);
    System.out.println(2);
    System.out.println(3);
}
```

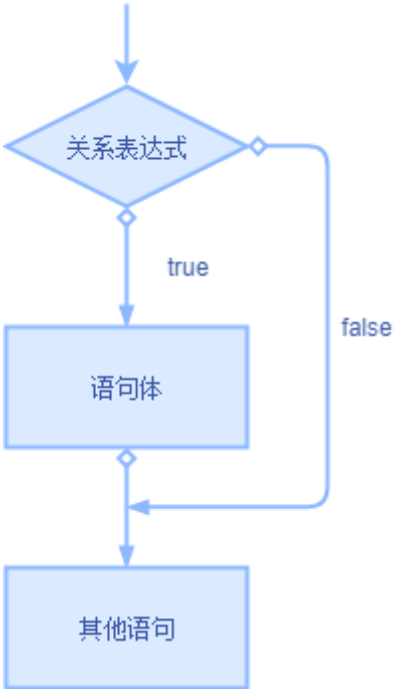
第二章 判断语句

2.1 if

语句格式：

```
if(关系表达式) {
    语句体;
}
```

执行流程：



例：

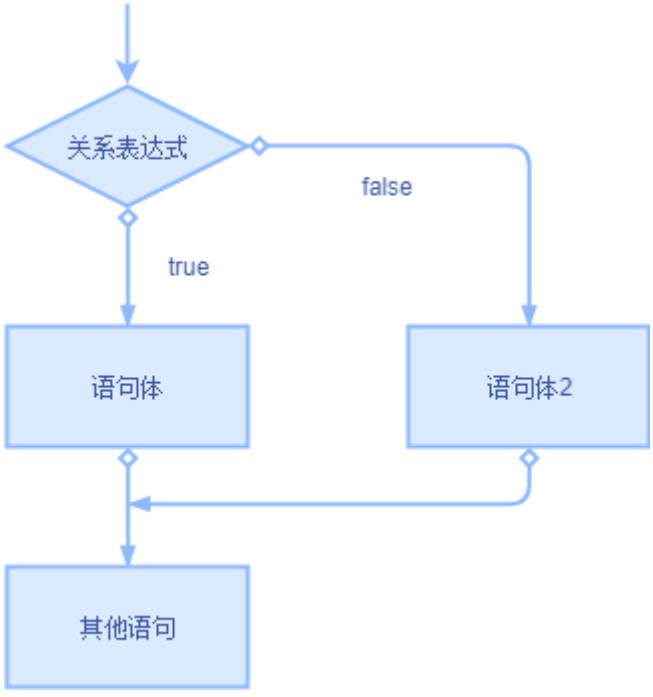
```
public static void main(String[] args) {
    System.out.println("开始");
    int a = 10;
    int b = 20;
    if(a==b){
        System.out.println("a等于b");
    }
    int c = 10;
    if(a == c){
        System.out.println("a等于c");
    }
    System.out.println("结束");
}
```

2.2 if...else

语句格式:

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

执行流程:



例:

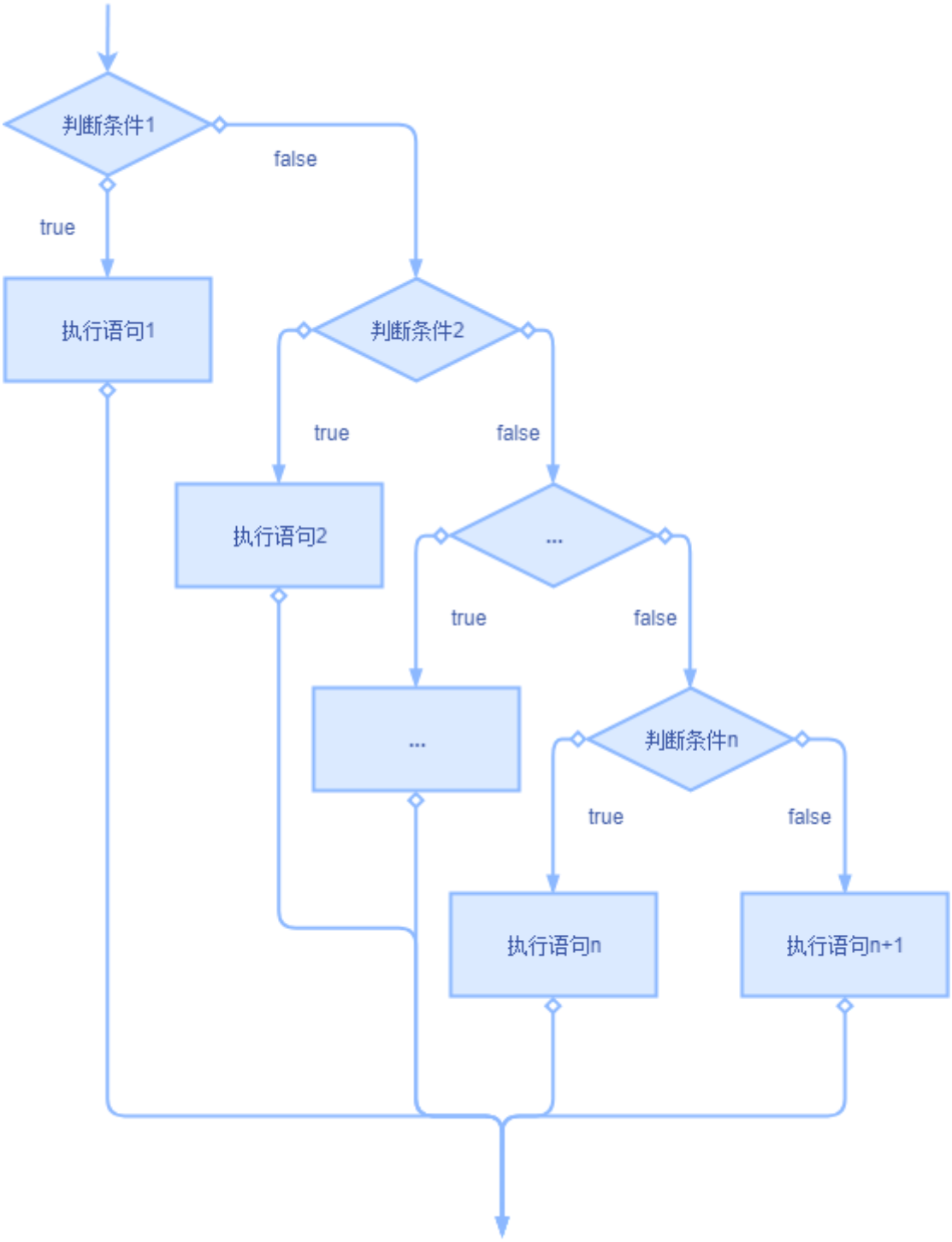
```
public static void main(String[] args) {
    int a = 1;
    if(a % 2 == 0) {
        System.out.println("a是偶数");
    }else {
        System.out.println("a是奇数");
    }
    System.out.println("结束");
}
```

2.3 if...else if...else

语句格式：

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

执行流程：



例:

```
public static void main(String[] args) {
    int x = 5;
    int y;
    if(x >= 3) {
        y = 2 * x + 1;
    }else if(x >= -1 && x < 3) {
        y = 2 * x;
    }else {
        y = 2 * x - 1;
    }
    System.out.println("y的值是: " + y);
}
```

2.4 练习

指定考试成绩，判断学生等级

- 90-100：优秀
- 80-89：好
- 70-79：良
- 60-69：及格
- 60以下：不及格

```
public static void main(String[] args) {
    int score = 100;
    if(score<0 || score>100) {
        System.out.println("你的成绩是错误的");
    }else if(score>=90 && score<=100) {
        System.out.println("你的成绩是优秀");
    }else if(score>=80 && score<90) {
        System.out.println("你的成绩是好");
    }else if(score>=70 && score<80) {
        System.out.println("你的成绩是良");
    }else if(score>=60 && score<70) {
        System.out.println("你的成绩是及格");
    }else {
        System.out.println("你的成绩是不及格");
    }
}
```

2.5 使用三元运算符代替if语句

简单情况下，三元运算符可以与if语句互换使用

```
public static void main(String[] args) {
    int a = 10;
    int b = 20;
    int c;
    //if语句
    if(a>b){
        c=a;
    }else{
        c=b;
    }
    //三元运算符
    c=a>b?a:b;
}
```

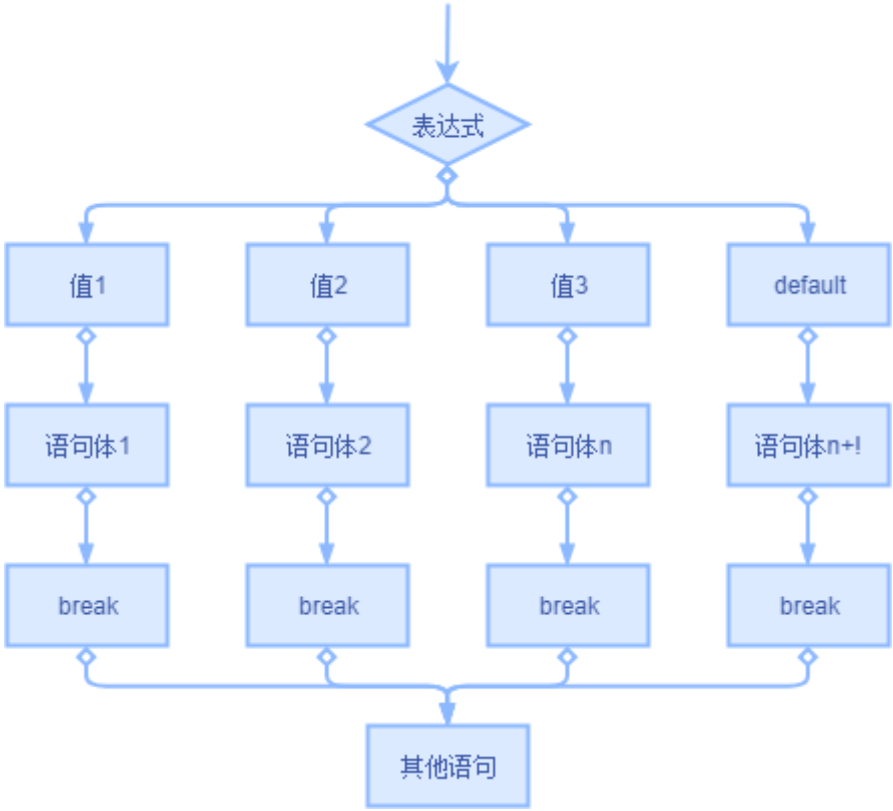
第三章 选择语句

3.1 switch

语句格式：

```
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：



例：

```
public static void main(String[] args) {
    //定义变量，判断星期几
    int weekday = 6;
    switch(weekday){
        case 1:
            System.out.println("星期一");
            break;
        case 2:
            System.out.println("星期二");
            break;
        case 3:
            System.out.println("星期三");
            break;
        case 4:
            System.out.println("星期四");
            break;
```

```
            case 5:
                System.out.println("星期五");
                break;
            case 6:
                System.out.println("星期六");
                break;
            case 7:
                System.out.println("星期七");
                break;
            default:
                System.out.println("你输入的数字有误");
                break;
        }
    }
```

注：switch语句中，表达式的数据类型，可以是byte、short、int、char、enum（枚举），JDK7之后可以接收字符串

3.2 case的穿透性

```
public static void main(String[] args) {
    int i=5;
    switch(i){
        case 0:
            System.out.println("case 0");
        case 5:
            System.out.println("case 5");
            break;
        case 10:
            System.out.println("case 10");

        default:
            System.out.println("default");
    }
}
```

运行结果：

```
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\Je
case 0
case 5

进程已结束，退出代码 0
```

- 执行case后，由于没有break语句，程序会一直向后走，不再判断case，一直到下一个break出现，否则将运行完整个程序

第四章 循环语句

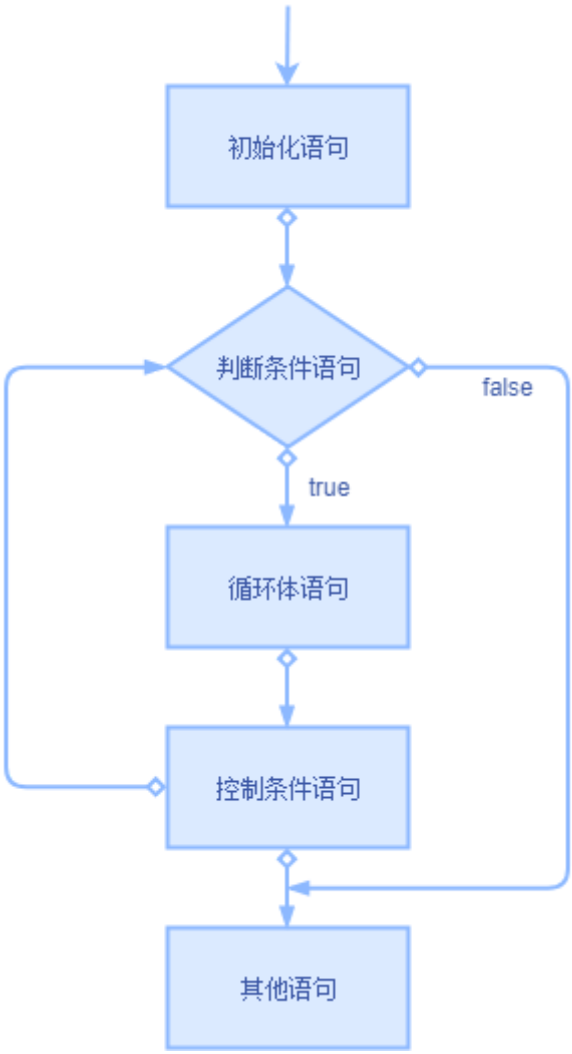
循环语句可以在满足循环条件的情况下，反复执行一段代码，这段被重复执行的代码称之为循环体语句，当反复执行这个循环体时，需要在合适的时候把循环条件修改为false，从而结束循环，否则循环将一直执行下去，形成死循环

4.1 for

语句格式：

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

执行流程：



例：

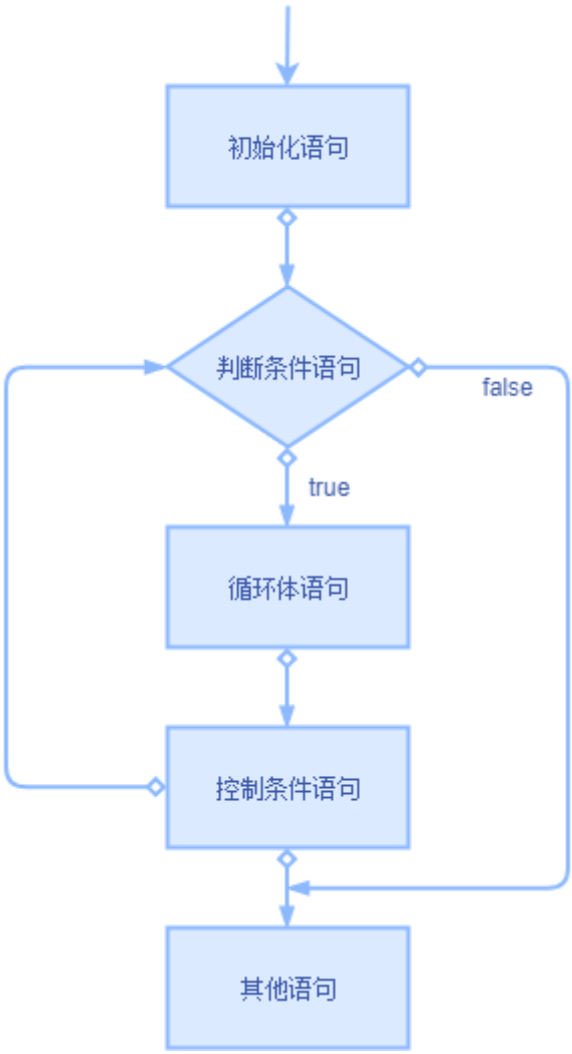
```
public static void main(String[] args) {
    int sum=0;
    for(int i=1;i<=100;i++) {
        if(i%2==0){ //偶数就累加求和
            sum+=1;
        }
    }
    System.out.println("sum="+sum);
}
```

4.2 while

语句格式：

```
初始化表达式;
while(布尔表达式) {
    循环体;
    步进表达式;
}
```

执行流程:



例:

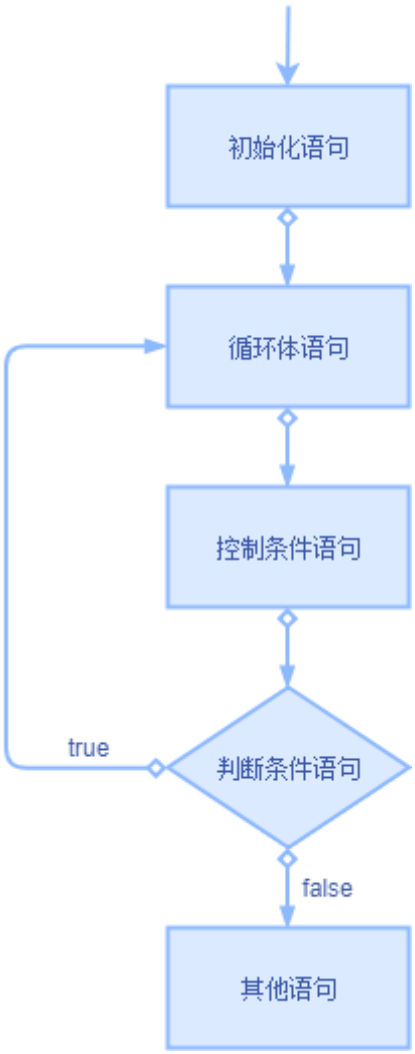
```
public static void main(String[] args) {
    int sum=0;
    int i=1;
    while(i<=100){
        sum+=i;
        i++;
    }
    System.out.println("1-100的和是："+sum);
}
```

4.3 do...while

语句格式:

```
初始化表达式;
do{
    循环体;
    步进表达式;
}while(布尔表达式);
```

执行流程：



例：

```
public static void main(String[] args) {
    do{
        System.out.println("无条件执行一次");
    }while(false);
}
```

- 注：`do...while`循环的特点就是不管循环条件是啥，会无条件执行一次循环体

4.4 循环语句的区别

- 控制条件语句所控制的变量，在for循环结束之后，就不能被访问到了，而while循环结束后还可以继续使用，如果想继续使用变量，就用while，否则推荐使用for，因为for循环结束后，该变量就从内存中消失，能够提高内存的使用效率
- 在已知循环次数的时候推荐使用for，循环次数未知的时候推荐使用while

4.5 跳出语句

4.5.1 break

使用场景：**终止switch或者循环**

例：

```
public static void main(String[] args) {
    //打印完两次HelloWorld之后结束循环
    for(int i=1;i<=10;i++) {
        if(i==3) {
            break;
        }
        System.out.println("HelloWrold"+i);
    }
}
```

4.5.2 continue

使用场景：**结束本次循环，继续下一次的循环**

例：

```
public static void main(String[] args) {
    //不打印第三次HelloWorld
    for(int i=1;i<=10;i++) {
        if(i==3) {
            continue;
        }
        System.out.println("HelloWrold"+i);
    }
}
```

4.6 扩展知识点

4.6.1 死循环

- 死循环也就是循环条件永远为true，永不结束的循环。例如：`while(true){}`
- 在后期的开发中，会出现使用死循环的场景，例如：需要读取用户的输入，但是用户输入多少数据我们并不清楚，也只能使用死循环，当用户不想输入数据了，就可以使用跳出语句，结束死循环。

4.6.2 嵌套循环

嵌套循环是指一个循环的循环体是另一个循环，总循环次数=外循环次数*内循环次数

例：

```java
public static void main(String[] args) {
    //嵌套循环打印5*8的矩阵
    for(int i=0;i<=5;i++) {
        for(int j=0;j<=8;j++) {
            //不换行打印星号
            System.out.print("*");
        }
        //内循环打印8个星号之后，需要一次换行
        System.out.println();
    }
}
```