2. 运算符、表达式

2.1 运算符

基本运算：`+`,`-`,`*`,`/` ； 整数的求余（取模）：`%` ；

整数除以0将会产生一个异常，而浮点数除以0将会得到无穷大或NaN的结果

2.2 数学函数与常量

`math` 类中，包含了多种数学函数：
`sqrt`：计算数值平方根；
`pow`：进行幂运算；`pow`方法有两个`double`类型的参数，其返回结果也为`double`类型。

2.3 结合赋值和运算符

```
x += 4;  // x = x + 4
```

2.4 自增与自减运算符

```
n++;  // n = n + 1
n--;   // n = n - 1
```

> 自增与自减运算符还有一种“前缀”形式：++n，--n；后缀和前缀形式都会使变量值加1或减1。在表达式中，前缀形式会先完成加1，而后缀形式会先使用变量原来的值。

2.5 关系和`boolean`运算符

相等与否判断： `==` ；
不相等与否判断： `!=` ；

大小： `<` , `>` , `<=`, `>=`

逻辑运算符： `!` 逻辑非， `&&` 逻辑与 ， `||` 逻辑或；

2.6 位运算符:

将位模型左移或右移：
处理整数类型时，可以直接对组成整型数值的各个位进行操作。位运算符包括：`&`("and"), `|`("or"), `^`("xor")、`~`("not")。另外还有`>>`和`<<`运算符将位模型左移或右移。

3.控制流程:

条件语句和循环结构确定控制流程.

3.1 块作用域

块 是由一对大括号括起来的若干条简单语句,**块**确定了变量的作用域.块可以嵌套.不可在嵌套的两个块中声明同名的变量.

```
public static void main(String[] args)
{
    int n;
    ...
    {
        int k;
        ...
    }
}
```

3.2 条件语句:

格式1: `if (condition) statement`
格式2: `if (condition) statement1 else statement2`

```
if(yourSales >= target)
{
    performance = "Satisfactory";
    bonus = 100;
}else{
    performance = "Unsatisfactory";
    bonus = 0;
}
```

3.3 循环:

格式: `while (condition)  statement`

3.4 确定循环次数(循环条件):

`for`语句的第一部分通常用于计数器初始化，第二部分给出每次新一轮循环执行前要检测的循环条件，第三部分指示如何更新计数器。`for`循环语句是`while`循环的一种简化形式。
可在各自独立的不同`for`循环中定义同名的变量:

```
for (int i = 1; i <= 10; i++){
    ...
}
for (int i = 11; i <= 20; i++){
    ...
}
```

3.5 中断控制流程语句:

`break` 语句退出循环;
`continue` 语句中断正常的控制流程,语句将控制转移到最内层循环的首部。

```
while(sum < goal){
    n = in.nextInt();
    if(n < 0) continue;     //如果n<0，则continue语句越过了当前循环体的剩余部分，立刻跳到循环首部。
    sum += n;
}
```

3.6 多重选择：`switch` 语句

`switch`语句将从与选项值相匹配的`case`标签处开始执行，直到遇到`break`语句，或执行到`switch`语句的结束处为止。若没有匹配的`case`标签，有`default`子句的话就会返回该默认值。

> 注意：若在`case`分支语句的末尾没有`break`语句，那么就会接着执行下一个`case`分支语句（这时候不看条件是否成立都会执行）。