

第 2 天 Java 基础语法

今日内容介绍

- ◆ 变量
- ◆ 运算符

第 1 章 变量

1.1 变量概述

前面我们已经学习了常量，接下来我们要学习变量。在 Java 中变量的应用比常量的应用要多很多。所以变量也是尤为重要的知识点！

什么是变量？变量是一个内存中的小盒子（小容器），容器是什么？生活中也有很多容器，例如水杯是容器，用来装载水；你家里的大衣柜是容器，用来装载衣裤；饭盒是容器，用来装载饭菜。

那么变量是装载什么的呢？答案是数据！**结论：变量是内存中装载数据的小盒子，你只能用它来存数据和取数据。**

1.2 计算机存储单元

变量是内存中的小容器，用来存储数据。那么计算机内存是怎么存储数据的呢？无论是内存还是硬盘，计算机存储设备的**最小信息单元**叫“**位（bit）**”，我们又称之为“**比特位**”，通常用小写的字母 b 表示。而计算机最小的存储单元叫“**字节（byte）**”，通常用大写字母 B 表示，字节是由连续的 8 个位组成。

当程序需要使用存储空间时，操作系统最小会分派给程序 1 个字节，而不是 1 个位。你可能会说，如果程序只需要 1 个位的空间，系统分派不能只分派 1 个位吗？答案是不能！这就像你只需要 1 支烟，你到商店去买烟，商店分派的最小单元是 1 盒（20 支），他不可能卖给你 1 支烟。

你可能会想，1 个字节（8 位）可以存储很大的数值了，1 位最大是 9 那么 8 位最大值为 99999999。你错了，因为计算机是采用二进行存储的，而不是我们生活中常用的十进制。所以 1 个字节存储的最大数据是 11111111 的二进制数。

除了字节外还有一些常用的存储单位，大家可能比较熟悉，我们一起来看看：

1B（字节）= 8bit

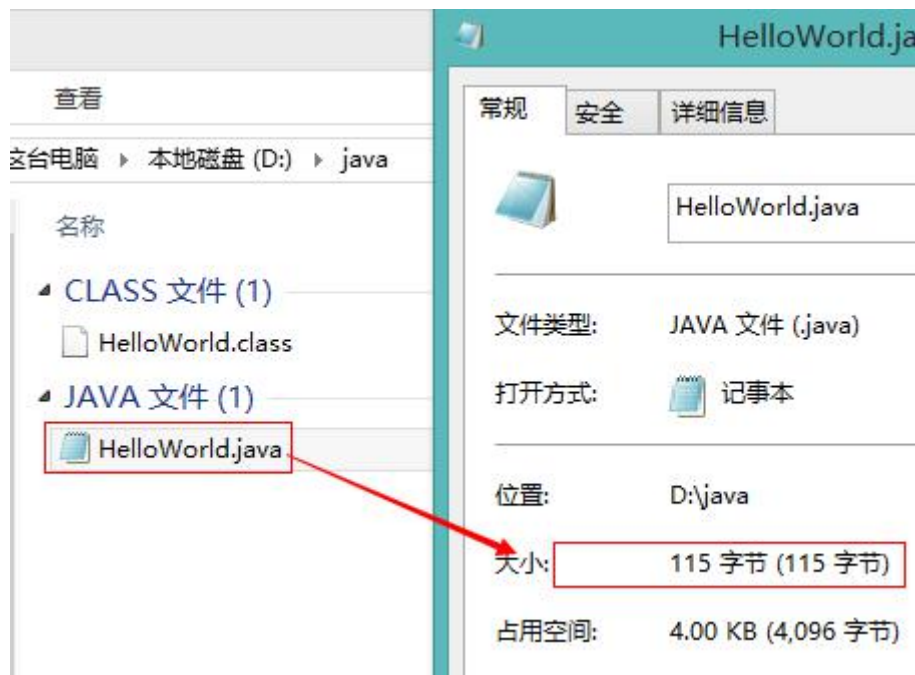
1KB = 1024B

1MB = 1024KB

1GB = 1024MB

1TB = 1024GB

1PB = 1024TB



1.3 基本类型之 4 类 8 种

大衣柜不能用来装载水，水杯也不能用来装载衣裤。这说明不同的容器装载不同的物品。变量

也是如此，在创建变量时需要指定变量的数据类型，例如整型变量、浮点型变量等等。**结论：变量必须要有明确的类型，什么类型的变量装载什么类型的数据。**

水杯是用来装水的，那么水杯能装多少水呢？一吨？我们知道水杯在创建时不只确定了要装载的是水（数据类型），而且还确定了能装多少水（数据类型的具体种类）。变量也是如此，需要指定变量能装载什么类型的数据，同时也要指定变量能装载多大的数据。

Java 中基本类型一共 4 类，把这 4 类展开后共 8 种基本类型。我们今后编写程序时使用的是这 8 种基本类型而不是 4 类，这 8 种基本类型指定了范围。

四类	八种	字节数	数据表示范围
整型	byte	1	-128 ~ 127
	short	2	-32768 ~ 32767
	int	4	-2147483648 ~ 2147483648
	long	8	$-2^{63} \sim 2^{63}-1$
浮点型	float	4	-3.403E38 ~ 3.403E38
	double	8	-1.798E308 ~ 1.798E308
字符型	char	2	表示一个字符，如('a'，'A'，'0'，'家')
布尔型	boolean	1	只有两个值 true 与 false

1.4 常量与类型

前面我们说过 100 是整数常量，但它是 byte、short、int、long 中的哪一种呢？下面我们来聊聊这一常量类型的问题。

整数常量可以根据所在范围来确定类型，例如 100 在 -128~127 之间，所以他是 byte 类型；

500 在-32768~32767 之间，所以它是 short 类型；100000 在-2147483648~2147483648 之间，所以它是 int 类型。[java 中默认的整数类型是 int 类型](#)

你可能会认为 12345678901 在 $-2^{63} \sim 2^{63}-1$ 之间，所以它是 long 类型。注意了，这是错误的 !!! 在 Java 中整数常量如果不在-2147483648~2147483648 之间就必须添加“L”后缀（小写的也可以，但建议使用大写），在-2147483648~2147483648 之间的也可以添加“L”后缀。也就是说 12345678901 不在-2147483648~2147483648 之间，所以它在 Java 中是错误的常量，你必须这样写：12345678901L，这才是正确的常量。所以添加了“L”后缀的整数常量都是 long 类型的，例如：100L、12345678901L 都是 long 类型的常量。

浮点类型的常量也可使用后缀，在 Java 中所有没有后缀以及使用“D”后缀（小写也可以，但建议使用大写）的小数都是 double 类型；float 类型常量必须添加“F”后缀（小写也可以，但建议使用大写）[java 中默认的浮点类型是 double 类型](#)

- 3.14 没有后缀，所以它是 double 类型；
- 5.28D 为 double 类型；
- 1.26F 为 float 类型。

1.5 定义变量（创建变量）

定义变量的语法格式：

```
数据类型 变量名 = 数据值；  
int      a    = 100;
```

其中 int 是数据类型，指定了变量只能存储整数，而且指定了存储范围为-2147483648 ~ 2147483648。

其中 a 表示变量名，变量名是标识符，这说明只要是合法的标识符都可以用来做变量名。在程

序中可以通过变量名来操作变量（内存中的小盒子）。

其中“=100”是给变量赋值，即向 a 变量中写入 100（变量是个小盒子，现在小盒子中保存的是 100）。注意，给变量赋的值一定要与类型符合，也就是说 int 类型只能存储整数，而且必须是在 -2147483648 ~ 2147483648 范围内的整数。100 满足了这两个条件，所以是正确的。

练习：

Variable.java

```
/*
变量定义格式：
数据类型 变量名 = 变量值;
*/
public class Variable {
    public static void main(String[] args) {
        int a = 10;
        double b = 3.14;
        char c = 'z';
        String s = "i love java";

        a = 20;
        System.out.println(a);
    }
}
```

1.6 变量使用的注意事项

我们使用变量的时候需要注意，要满足变量的使用规则才可以使用的，我们来看看都有哪些注意事项。

- 变量使用的注意事项

- 变量定义后可以不赋值，使用时再赋值。不赋值不能使用。

```
public static void main(String[] args) {
    int x;
    x = 20; //为 x 赋值 20
    System.out.println(x); //读取 x 变量中的值，再打印
}
```

- 变量使用时有作用域的限制。

```
public static void main(String[] args) {  
    int x = 20;  
    {  
        int y = 20;  
    }  
    System.out.println(x); // 读取 x 变量中的值，再打印  
    System.out.println(y); // 读取 y 变量中的值失败，失败原因，找不到 y 变量，因为超出了 y 变量作用范围，所以不能使用 y 变量  
}
```

- 变量不可以重复定义。

```
public static void main(String[] args){  
    int x = 10;  
    double x = 5.5; // 编译失败，变量重复定义  
}
```

1.7 数据类型转换

不同类型的变量是否可以在一起运算呢？答案是可以的，但要先进行类型转换再运算。下面我们来学习一下类型转换。

其实，我们所学习的数据，它的表示方式是可以灵活多变的，比如把小数转换成整数的操作

转换的过程中，数据遵循一个原则：

范围小的数据类型值（如 byte），可以直接转换为范围大的数据类型值（如 int）；

范围大的数据类型值（如 int），不可以直接转换为范围小的数据类型值（如 byte）

那么，大家还记得每种类型表示数据的范围吗？忘记了不要紧，我来告诉大家，将各种数据类型按照数据范围从小到大依次列出：

```
byte -> short -> int -> long -> float -> double
```

关于数据类型转换有两种方式，我们来学习一下：

- 自动类型转换

表示范围小的数据类型转换成范围大的数据类型，这种方式称为自动类型转换

自动类型转换格式：

范围大的数据类型 变量 = 范围小的数据类型值；

如：

```
double d = 1000;
```

或

```
int i = 100;  
double d2 = i;
```

- 强制类型转换

表示范围大的数据类型转换成范围小的数据类型，这种方式称为强制类型转换

强制类型转换格式：

范围小的数据类型 变量 = (范围小的数据类型) 范围大的数据类型值;

如：

```
int i = (int)6.718; //i 的值为 6
```

或

```
double d = 3.14;  
int i2 = (int)d; //i2 的值为 3
```

第 2 章 运算符

2.1 算术运算符

运算符是用来计算数据的符号。数据可以是常量，也可以是变量。被运算符操作的数我们称为操作数。

算术运算符最常见的操作就是将操作数参与数学计算，具体使用看下图：

运算符	运算规则	范例	结果
+	正号	+3	3
+	加	2+3	5
+	连接字符串	“中”+“国”	“中国”
-	负号	int a=3;-a	-3
-	减	3-1	2
*	乘	2*3	6
/	除	5/2	2
%	取模	5/2	1
++	自增	int a=1;a++/++a	2
--	自减	int b=3;a--/--a	2

我们在使用算术运算符时，记得要注意下列事项：

- 加法运算符在连接字符串时要注意，只有直接与字符串相加才会转成字符串。
- 除法“/”当两边为整数时，取整数部分，舍余数。当其中一边为浮点型时，按正常规则相除。
- “%”为整除取余符号，小数取余没有意义。结果符号与被取余符号相同。
- 整数做被除数，0 不能做除数，否则报错。

代码演示

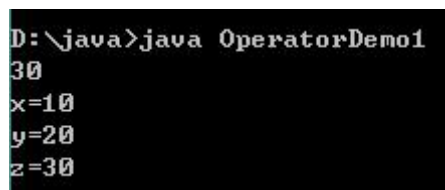
```
/*
 * 算术运算符
 */
public class OperatorDemo1 {
    public static void main(String[] args) {
```



```
/*
 * 常量使用算数运算符
 */
System.out.println(10+20);

/*
 * 变量使用算数运算符
 */
int x = 10;
int y = 20;
// "+" 作为加法运算使用
int z = x + y;
// "+" 作为连接字符串使用
System.out.println("x="+x);
System.out.println("y="+y);
System.out.println("z="+z);
}
}
```

运行结果如下图所示。



```
D:\java>java OperatorDemo1
30
x=10
y=20
z=30
```

图 1-1 运行结果

2.2 算数运算符++、--的使用

算数运算符在前面我们已经学习过了，这里进行一些补充。

在一般情况下，算数运算符不会改变参与计算的变量值。而是在原有变量值不变的情况下，计算出新的值。但是有些操作符会改变参与计算的变量的值，比如++，--。

我们来看一段代码：

```
int a = 3;
int b = 3;
a++;
b--;
System.out.println(a);
System.out.println(b);
```

上面代码的输出结果 a 值为 4，b 值为 2；

这说明 a 的原有值发生了改变，在原有值的基础上自增 1；b 的原有值也发生了改变，在原有值的基础上自减 1；

- ++运算符，会在原有值的基础上自增 1；
- --运算符，会在原有值的基础上自减 1。

我们再看一段代码：

```
int a = 3;
int b = 3;
++a;
--b;
System.out.println(a);
System.out.println(b);
```

上面代码的输出结果 a 值为 4，b 值为 2；

这说明++/--运算符单独使用，不参与运算操作时，运算符前后位置导致的运算结果是一致的。

接下来，介绍下++，--运算符参与运算操作时，发生了怎样的变化，我们来看一段代码：

```
int a = 3;
int b;
b = a++ + 10;
System.out.println(a);
System.out.println(b);
```

上面代码的输出结果 a 值为 4，b 值为 13；

这里我要强调一下了，当++，--运算符参与运算操作时，后置++，--的作用：

- ++/--运算符后置时，先使用变量 a 原有值参与运算操作，运算操作完成后，变量 a 的值自增 1 或者自减 1；

再介绍下++，--运算符前置时，参与运算操作的变化，我们来看一段代码：

```
int a = 3;
int b;
b = ++a + 10;
System.out.println(a);
System.out.println(b);
```

上面代码的输出结果 a 值为 4，b 值为 14；

这里我强调一下，当++，--运算符参与运算操作时，前置++，--的作用：

- ++，--运算符前置时，先将变量 a 的值自增 1 或者自减 1，然后使用更新后的新值参与运算操作。

2.3 赋值运算符

我们来学习一下赋值运算符，赋值运算符就是为变量赋值的符号，赋值运算符的使用看下图：

运算符	运算规则	范例	结果
=	赋值	int a=2	2
+=	加后赋值	int a=2，a+=2	4
-=	减后赋值	int a=2，a-=2	0
=	乘后赋值	int a=2，a=2	4
/=	整除后赋值	int a=2，a/=2	1
%=	取模后赋值	int a=2，a%=2	0

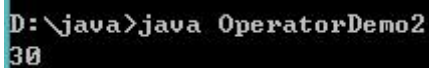
注意：诸如+=这样形式的赋值运算符，会将结果自动强转成等号左边的数据类型。

写一个代码，我们看一下赋值运算符的使用

```
/*
* 赋值运算符
```

```
* +=, -=, *=, /=, %= :
* 上面的运算符作用：将等号左右两边计算，会将结果自动强转成等号左边的数据类型,再赋值给等号左边的
* 注意：赋值运算符左边必须是变量
*/
public class OperatorDemo2 {
    public static void main(String[] args) {
        byte x = 10;
        x += 20; // 相当于 x = (byte)(x+20);
        System.out.println(x);
    }
}
```

运行结果如下图所示。



```
D:\java>java OperatorDemo2
30
```

图 1-2 运行结果

2.4 比较运算符

比较运算符，又叫关系运算符，它是用来判断两个操作数的大小关系及是否相等关系的，结果是布尔值 true 或者 false。

运算符	运算规则	范例	结果
==	相等于	4==3	False
!=	不等于	4!=3	True
<	小于	4<3	False
>	大于	4>3	True
<=	小于等于	4<=3	False
>=	大于等于	4>=3	True

这里需要注意一下：

- 赋值运算符的 = 符号与比较运算符的 == 符号是有区别的，如下：
 - 赋值运算符的 = 符号，是用来将 = 符号右边的值，赋值给 = 符号左边的变量；
 - 比较运算符的 == 符号，是用来判断 == 符号 左右变量的值是否相等的。

我们通过下面的一段代码，我们演示一下这个注意事项：

```
int a = 3;
int b = 4;
System.out.println( a=b );
System.out.println( a==b );
```

上面代码输出的结果第一个值为 4，第二个值为 false。

2.5 逻辑运算符

逻辑运算符，它是用于布尔值进行运算的，运算的最终结果为布尔值 true 或 false。

运算符	运算规则	范例	结果
&	与	false&true	False
	或	false true	True
^	异或	true^flase	True
!	非	!true	Flase
&&	短路与	false&&true	False
	短路或	false true	True

看完图后，我们来看一下逻辑运算符的常规使用方式：

- 逻辑运算符通常连接两个其他表达式计算后的布尔值结果

- 当使用短路与或者短路或时，只要能判断出结果则后边的部分就不再判断。

我们通过代码演示一下：

```
boolean b = 100>10;
boolean b2 = false;
System.out.println(b&&b2); // 打印结果为 false
System.out.println(b||b2); //打印结果为 true
System.out.println(!b2); //打印结果为 true
System.out.println(b && 100>10); //打印结果为 true，本行结果的计算方式稍后讲解运算符优先级时解
```

答

好了，我们来总结一下运算符的结果规律吧：

- 短路与&&:参与运算的两边数据，有 false，则运算结果为 false；
- 短路或||:参与运算的两边数据，有 true，则运算结果为 true；
- 逻辑非!: 参与运算的数据，原先是 true 则变成 false，原先是 false 则变成 true。

2.6 三元运算符

接下来我们要学习的三元运算符与之前的运算符不同。之前学习的均为一元或者二元运算符。

元即参与运算的数据。

- 格式：

(条件表达式) ? 表达式 1 : 表达式 2 ;

- 表达式：通俗的说，即通过使用运算符将操作数联系起来的式子，例如：
 - $3+2$ ，使用算数运算符将操作数联系起来，这种情况，我们称为算数表达式。
 - $3>2$ ，使用比较运算符（也称为条件运算符）将操作数联系起来，这种情况，我们称为条件表达式。
 - 其他表达式，不再一一举例。
- 三元运算符运算规则：

先判断条件表达式的值，若为 true，运算结果为表达式 1；若为 false，运算结果为表达式 2。

通过代码演示，我们来学习下三元运算符的使用：

```
方式一：
    System.out.println( 3>2 ? “正确” : “错误” );
    // 三元运算符运算后的结果为 true，运算结果为表达式 1 的值“正确”，然后将结果“正确”，在控制台输出打印

方式二：
    int a = 3;
    int b = 4;
    String result = (a==b) ? “相等” : “不相等”;
    //三元运算符运算后的结果为 false，运算结果为表达式 2 的值“不相等”，然后将结果赋值给了变量 result

方式三：
    int n = (3>2 && 4>6) ? 100 : 200;
    //三元运算符运算后的结果为 false，运算结果为表达式 2 的值 200,然后将结果 200 赋值给了变量 n
```

2.7 运算符优先级

在学习运算符的过程中，我们发现，当多个运算符一起使用的时候，容易出现不清晰先后运算顺序的问题，那么，在这里，我们来学习下，运算符之间的运算优先级。

下图是每种运算符的优先级，按照运算先后顺序排序（优先级相同的情况下，按照从左到右的顺序依次运算）

优先级	描述	运算符
1	括号	()、[]
2	正负号	+, -
3	自增自减，非	++, --, !
4	乘除，取余	*, /, %

5	加减	+, -
6	移位运算	<<、>>、>>>
7	大小关系	>、>=、<、<=
8	相等关系	==、!=
9	按位与	&
10	按位异或	^
11	按位或	
12	逻辑与	&&
13	逻辑或	
14	条件运算	?:
15	赋值运算	=、+=、-=、*=、/=、%=
16	位赋值运算	&=、 =、 <<=、>>=、>>>=

优先级顺序看完了，我们来通过代码，加强一下：

```
int n = 3>4 ? 100 : 200;
```

这行的代码运算执行顺序我们来写一下：

- 1.执行 3>4 操作，得到布尔类型 false 的结果
- 2.通过结果 false，将对应的表达式 2 的结果 200，作为运算的最终结果
- 3.把 200 赋值给变量 n

接下来，我们看一个比较复杂的代码：

```
int a = 5;
int b = 3;
int c = 1;
int n2 = (a>b && b>c) ? (c++) : (++c);
```


这段代码运算执行顺序我们也写一下：

- 1.小括号优先级高，我们先运算第一组小括号中的代码
- 1.1. 比较运算符”>” 优先级大于 逻辑运算符”&&”
 - 先执行 `a>b`,得到结果 `true`；
 - 再执行 `b>c`,得到结果 `true`；
 - 最后执行 `a>b` 的结果 `&& b>c` 的结果，即 `true && true`，结果为 `true`
- 2.三元运算符中的条件判断结果为 `true`，返回表达式 1 的结果 `c++`
 - 先将变量 `c` 的原有值赋值给变量 `n2`，即 `n2` 值为 1；
 - 再将变量 `c` 的值自增 1,更新为 2。

运算符我们学到这里就结束了，稍后在“趣味乐园”中，我们可以运用运算符进行练习。

第 3 章 商场库存清单案例

3.1 案例介绍

现在我们来做一个复杂点的案例——商场库存清单案例，这个案例最终会在控制台输出如下结

果：

```
-----商城库存清单-----
品牌型号      尺寸      价格      库存数
MacBookAir    13.3      6988.88  5
ThinkpadT450  14.0      5999.99  10
ASUS-FL5800   15.6      4999.5   18
-----
总库存数: 33
库存商品总金额: 184935.3
```

3.2 案例需求分析

- 观察清单后，可将清单分解为三个部分（清单顶部、清单中部、清单底部）

品牌型号	尺寸	价格	库存数
MacBookAir	13.3	6988.88	5
ThinkpadT450	14.0	5999.99	10
ASUS-FL5800	15.6	4999.5	18

总库存数: 33
库存商品总金额: 184935.3

- 1.清单顶部为固定的数据，直接打印即可
- 2.清单中部为商品，为变化的数据，需要记录商品信息后，打印

经过观察，我们确定一项商品应该有如下几个属性：

品牌型号：即商品名称，String 型

尺寸：物品大小，double 型

价格：物品单价，double 型

配置：这一项为每种商品的配置信息，String 型

库存数：这一项为每种商品的库存个数，int 型

- 3.清单底部包含了统计操作，需经过计算后，打印

我们发现两个单独的可变化量

总库存数：所有商品总个数，int 型

库存商品总金额：所有商品金额，double 型

3.3 实现代码步骤

一起分析完毕了，我们开始完成案例代码的编写：

- 创建 Demo01 库存清单.java 文件，编写 main 主方法

```
public class Demo01 库存清单 {
```

```
public static void main(String[] args) {  
    }  
}
```

- 记录每种库存商品信息

```
//苹果笔记本电脑  
String macBrand = "MacBookAir";  
double macSize = 13.3;  
double macPrice = 6988.88;  
int macCount = 5;  
  
//联想 Thinkpad 笔记本电脑  
String thinkpadBrand = "ThinkpadT450";  
double thinkpadSize = 14.0;  
double thinkpadPrice = 5999.99;  
int thinkpadCount = 10;  
  
//华硕 ASUS 笔记本电脑  
String ASUSBrand = "ASUS-FL5800";  
double ASSUSize = 15.6;  
double ASUSPrice = 4999.50;  
int ASUSCount = 18;
```

- 统计所有库存商品数量与金额

```
//统计库存总个数、库存总金额  
int totalCount = macCount + thinkpadCount + ASUSCount;  
double totalMoney = (macCount * macPrice) + (thinkpadCount * thinkpadPrice) +  
(ASUSCount * ASUSPrice);
```

- 打印库存清单顶部信息

```
//列表顶部  
System.out.println("-----商城库存清单-----");  
System.out.println("品牌型号 尺寸 价格 库存数");
```

- 打印库存清单中部信息

```
//列表中部  
System.out.println(macBrand+" "+macSize+" "+macPrice+" "+macCount);  
System.out.println(thinkpadBrand+" "+thinkpadSize+" "+thinkpadPrice+"  
"+thinkpadCount);  
System.out.println(ASUSBrand+" "+ASUSize+" "+ASUSPrice+" "+ASUSCount);
```

- 打印库存清单底部信息

```
//列表底部  
System.out.println("-----");
```

```
System.out.println("总库存数："+totalCount);  
System.out.println("库存商品总金额："+totalMoney);
```

第 4 章 总结

4.1 知识点总结

- 数据类型转换
 - 数据类型范围从小到大排序（byte < char < short < int < long < float < double），布尔类型 Boolean 不能参与类型转换；
 - 自动类型转换，范围小的数据类型向范围大的数据类型转换时使用；
 - 强制类型转换，范围大的数据类型向范围小的数据类型转换时使用。
- 算数运算符
 - 用来完成算数运算（如加减乘除计算等）
 - ++，--运算符的使用
 - ◆ ++，--前置（如++a），当参与运算时，变量 a 的值先自增 1，然后用自增后的新值再参与运算；
 - ◆ ++，--后置（如 a++），当参与运算时，变量 a 先使用原有值参与运算符，变量 a 的值再自增 1。
- 赋值运算符
 - 用来完成数据的赋值（如 int a = 100;）
 - +=, -=, *=, /= 这样的赋值运算符包含了一个强制转换的操作，会将左右两边运算后的结果，强制类型转换后赋值给左边

```
int n = 10;
byte by = 20;
by += n; // 运算完毕后，by 的值为 byte 类型 30，相当于代码 by = (byte)(by + n);
```

- 比较运算符

- 用来比较数据的大小（如 $3 > 4$ ），比较运算符也称为条件运算符。
- 比较后的结果为布尔类型 Boolean 的值
- “==”两个等号代表比较是否相等，“=”一个等号代表赋值。

- 逻辑运算符

- 逻辑与 & 和逻辑短路与 && :代表着并且的意思 ,左右两边都要条件成立 ,结果才为 true ;
- 逻辑或 | 和逻辑短路或 || :代表着或者的意思 ,左边两边有一个条件成立 ,结果就为 true ;
- 逻辑非 ! :代表着相反的意思 ,原先是 false ,结果就为 true ;原先是 true ,结果就为 false ;
- 逻辑异或 ^ : 左右两边条件结果相同 , 结果就为 false , 左右两边条件结果不同 , 结果就为 true ;

- 三元运算符

- 根据判断条件运算结果，执行不同的表达式值；条件为 true，执行表达式 1，否则，执行表达式 2。