day02-变量和运算符

今日学习内容：

- 常量
- 数据类型
- 变量的定义和使用
- 基本数据类型的转换
- 算术运算符
- 赋值运算符
- 比较运算符
- 三元运算符
- 逻辑运算符

今日学习目标：

必须掌握变量的定义和赋值

掌握什么是表达式

掌握基本数据类型的自动转换

掌握基本数据类型的自动提升

掌握基本数据类型的强制转换

掌握算术运算符的使用

了解什么是前置++和后置++的区别

掌握赋值运算符的使用，以及它的底层含义

掌握比较运算符的使用

必须掌握三元运算符的语法和使用

掌握逻辑运算符的使用（常用 && 、|| 、！）

了解位与（&）和短路与（&&）的区别，记住结论使用&&即可

了解运算符的优先级

```
学习方法提醒：
1> 课前做好预习
2> 如果遇到听不懂的，记下来，放过去。专心听下一个知识点。
```

2. Java入门基础下

2.1 数据类型、常量、变量

2.1.1.常量（掌握）

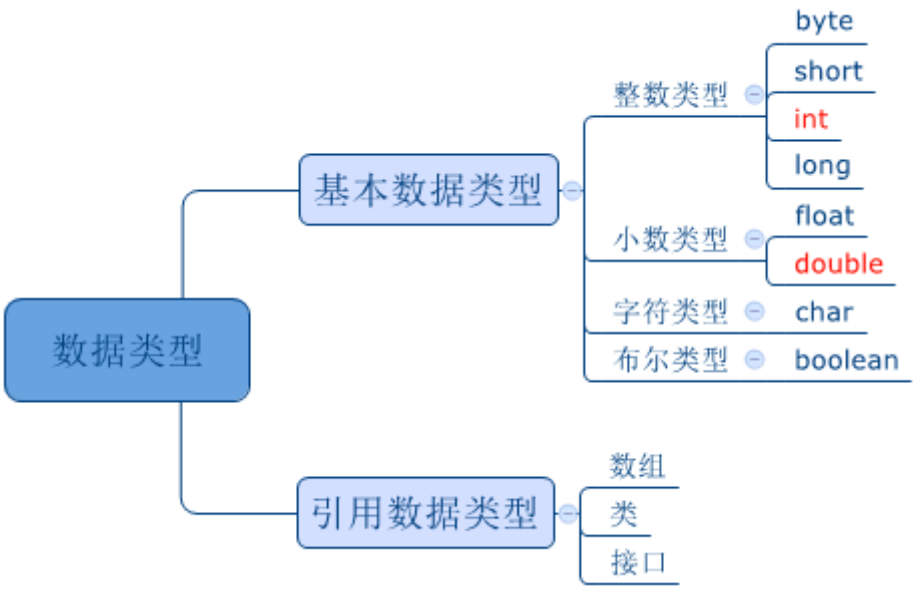
常量，程序运行过程中固定不变化的值。

常量分类：

- 字面量： 就表示直接给出的一个值（可以是整数、小数等），也有人称之为直接量。如整数常量1，2，3，小数常量3.14等。
- 使用final定义的变量（后讲）

2.1.2. 数据类型（重点）

生活中，数据都是有类型这个概念的，比如张三18岁，18这个数字就是整型的，买了2.5斤菜，2.5就是小数类型的，在Java中每一个数据也有数据类型。



8种基本数据数据类型的范围和占内存大小（了解）：

人为的把8个二进制位成为1个字节。（1 byte = 8 二进制位）

NO.	数据类型	占位(字节)	数据范围	默认值
1	byte	1	[-128, 127]	0
2	short	2	$[-2^{15}, 2^{15}-1]$	0
3	int	4	$[-2^{31}, 2^{31}-1]$	0
4	long	8	$[-2^{63}, 2^{63}-1]$	0L
5	char	2	$[0, 2^{16}-1]$	'\u0000'
6	float	4	$[-3.4E38(-3.4*10^{38}), 3.4E38(-1.7*10^{30}]$	0.0F
7	double	8	$[-1.7E308(-1.7*10^{308}), 1.7E308(-1.7*10^{308}]$	0.0D
8	boolean	1位	false、true	false

开发者需要明确记住每个类型所占字节数（内存大小）。

整数类型常量

在 java 中，整数类型的常量JVM默认使用 int 类型来存储 。

```java
public class Test01Int {
    public static void main (String[] args) {
        // 控制台输出10，这个10是一个常量，默认使用int存储。
        System.out.println(10);

        // 控制台输出20，这个20如果想以long类型存储，需要加L
        System.out.println(20L);
    }
}
```

所以，如果要存储long类型常量，要加L或者l，建议加L。

常用的整数类型是int和long，byte和short基本不用。

小数类型类型

在 java 中，小数类型的常量JVM默认使用 double 类型来存储 。

```java
public class Test02Double {
    public static void main (String[] args) {
        // 控制台输出3.14，这个3.14是一个小数常量，默认使用double存储。
        System.out.println(3.14);

        // float类型常量,使用F后缀
        System.out.println(3.14F);
    }
}
```

如果要存储 float 类型常量，要加f或者F。

float类型又被称作单精度类型，尾数可以精确到6-7位有效数字，在很多情况下，float类型的精度很难满足需求，double更常用，double表示小数的数值精度是float类型的两倍，又被称作双精度，绝大部分应用程序都采用double类型。尾数可以精确到15-16位有效数字.

不管是float 类型和double类型在有效数字内都存在精确问题。

```java
// float 类型只能精确到6-7，其中第六位是一定可以精确到的，第7可能精确到，也可能精确不到。
float a = 3.141592653f;
System.out.println("a = " + a);
// a = 3.1415927
```

所以，实际开发过程中，小数类型一定要在有限数字范围内使用。

float，double 的数据不适合在不容许舍入误差的金融计算领域。如果需要进行不产生舍入误差的精确数字计算，需要使用 BigDecimal 类 (后面学) 。

字符类型常量

字符表示Unicode（万国码）编码表中的每一个符号，每个符号使用单引号引起来，其中前128个符号和ASCII表相同，如下图。

ASCII 字符代码表 一

高四位 / 低四位		ASCII非打印控制字符										ASCII 打印字符															
		0000					0001					0010		0011		0100		0101		0110		0111					
		0					1					2		3		4		5		6		7					
		十进制	字符	ctrl	代码	字符解释	十进制	字符	ctrl	代码	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	ctrl			
0000	0	0	BLANK NULL	^@	NUL	空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p				
0001	1	1	☺	^A	SOH	头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q				
0010	2	2	☻	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r				
0011	3	3	♥	^C	ETX	正文结束	19	‼	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s				
0100	4	4	♦	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t				
0101	5	5	♣	^E	ENQ	查询	21	§	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u				
0110	6	6	♠	^F	ACK	确认	22	▬	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v				
0111	7	7	•	^G	BEL	震铃	23	↨	^W	ETB	传输块结束	39	'	55	7	71	G	87	W	103	g	119	w				
1000	8	8	◘	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x				
1001	9	9	○	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y				
1010	A	10	◙	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z				
1011	B	11	♂	^K	VT	竖直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{				
1100	C	12	♀	^L	FF	换页/新页	28	∟	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|				
1101	D	13	♪	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}				
1110	E	14	♫	^N	SO	移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~				
1111	F	15	☼	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	Back space			

这张表要记住的几个符号，A在码表的顺序是65，a在码表的顺序是97。

```
public class Test03Char {
    public static void main (String[] args) {
         // 控制台输出字符A，字符在内存中以两个字节存储。
        System.out.println('A');
    }
}
```

布尔类型常量

boolean 类型只有两个值，true 和 false , 在未来的开发中用于逻辑判断。

```
public class Test04Boolean {
    public static void main (String[] args) {
        System.out.println(true);
    }
}
```

字符串类型

所谓字符串就是多个字符合在一起，使用双引号引起来。例如 : 我们在开发中要输出一个用户的名字就可以使用字符串了，因为一个用户的名字是由多个字符构成的。

```
public class Test05String {
    public static void main (String[] args) {
        System.out.println("wolfcode");

        // 需求：尝试输出你的姓名？
        // 思考：一个字符一个字符的输出，还是一下子就输出？
    }
}
```

字符串类型属于引用数据类型，不属于8大基本数据类型范畴，不在今天课程讨论范围内，后续会学习。但今天我们会写即可。

不同数据类型的常量：

- 整数常量，所有整数，如1、2、3、100、200等
- 小数常量，所有小数，如1.2、2.7、3.14等
- 字符常量，0~65535之间的整数或用单引号括起来的符号如，‘A’、‘a’、‘龙’等
- 布尔常量，只有true和false，分别表示对与错
- 字符串常量，使用双引号括起来的内容如：“Will”、“wolfcode”等

需求：定义每一种数据类型的常量

```
public class TypeDemo{
    public static void main(String[] args) {
        //byte类型常量
        System.out.println(20);

        //short类型常量
        System.out.println(20);

        //int类型常量
        System.out.println("十进制" + 20);
        System.out.println("二进制" + 0B00010100);
        System.out.println("八进制" + 024);
        System.out.println("十六进制" + 0x14);

        //long类型常量,使用L后缀
        System.out.println(20L);

        //float类型常量,使用F后缀
        System.out.println(3.14F);

        //double类型常量
        System.out.println(3.14);

        //char类型常量
        System.out.println(65);
        System.out.println('A');

        //boolean类型常量
        System.out.println(true);
        System.out.println(false);

        //String类型常量
        System.out.println("你好");
    }
}
```

2.1.3.变量（重点）

通过一张不完整的房屋租赁合同，引出变量。

案例：张三需要租赁李四的房屋，租赁合同如下：

张三、李四双方就下列房屋的租赁达成如下协议：
第一条 房屋押金
张三、李四双方自本合同签订之日起，由张三支付李四一个月房租的金额作为押金。
第二条 租赁期满。
1、租赁期满后，如张三要求继续租赁，李四则优先同意继续租赁；
2、租赁期满后，如李四未明确表示不续租的，则视为同意张三继续承租；
3、租赁期限内，如张三明确表示不租的，应提前一个月告知李四，李四应退还张三已支付的租房款及押金。

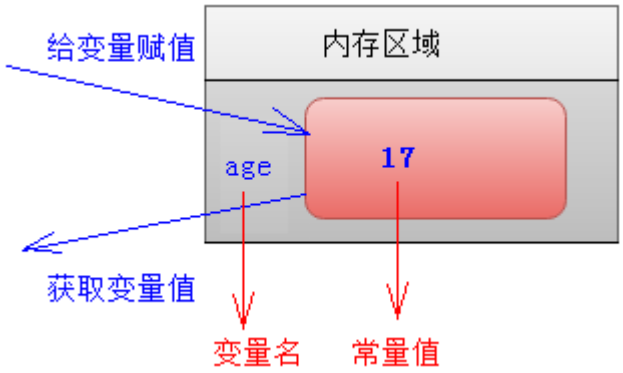
上述合同，相当不正规，因为正规的合同上，租客和房东都是有变动的，不能写死，在整个合同中应该是使用甲方来表示房东，乙方来表示租客，只会在最后的时候签名甲方是谁，乙方是谁。

甲、乙方双方就下列房屋的租赁达成如下协议：
第一条 房屋押金
甲、乙双方自本合同签订之日起，由乙方支付甲方一个月房租的金额作为押金。
第二条 租赁期满。
1、租赁期满后，如乙方要求继续租赁，甲方则优先同意继续租赁；
2、租赁期满后，如甲方未明确表示不续租的，则视为同意乙方继续承租；
3、租赁期限内，如乙方明确表示不租的，应提前一个月告知甲方，甲方应退还乙方已支付的租房款及押金。
甲方（签章）：李四　　乙方（签章）：张三

2.1.3.1 变量概述（了解）

变量指在程序运行过程中，值可以发生变化的量。

变量表示一个存储空间，可用来存放某一类型的常量，没有固定值，并可以重复使用。变量是内存中一块区域，可以往该区域存储数据，修改里面的数据，也可以获取里面的数据。



变量声明的语法

```
数据类型 变量名 [= 初始值];
```

变量的特点：

- 占据着内存中的某一块存储区域
- 该区域有自己的名称（变量名）和类型（数据类型）
- 可以被重复使用
- 该区域的数据可以在同一类型范围内不断变化

2.1.3.2 变量定义和赋值（重点）

需求：定义一个int类型变量，并赋值。

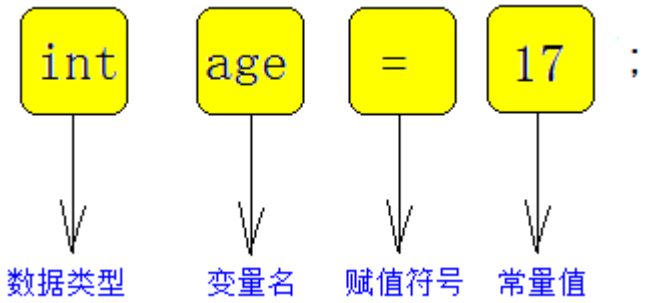
```
public class VarDemo{
    public static void main(String[] args) {
        // 方式一，先变量，后赋值，再使用
        // 1.数据类型  变量名;
        int age;
        // 2.变量名 = 常量值;
        age = 17; // 定义一个int类型变量,初始值为17

        // 修改age变量的值为17
        age = 22;
        // 3. 使用定义的变量
        System.out.println(age);


        // 方式二，在声明时同时赋值（推荐）
        // 数据类型 变量名 = 初始化值;
        // 定义一个String类型的变量，初始值为wolf
        String name = "wolf";
    }
}
```

使用变量注意：

- 变量必须先声明，并且初始化后才能使用
- 定义变量必须有数据类型
- 变量从开始定义到所在的花括号结束之内可以使用，离开花括号就不能使用了
- 同一作用域内，变量名不能重复定义

记：语法格式

- String，表示类型，这里可以写任何的类型
- name：变量名，和我们的姓名一样理解， 没有为什么
- =：赋值运算符，后面会讲，意思是将右边的值赋值给左边的变量
- “wolf”：一个字符串类型的值，如果是其他类型，不要加引号

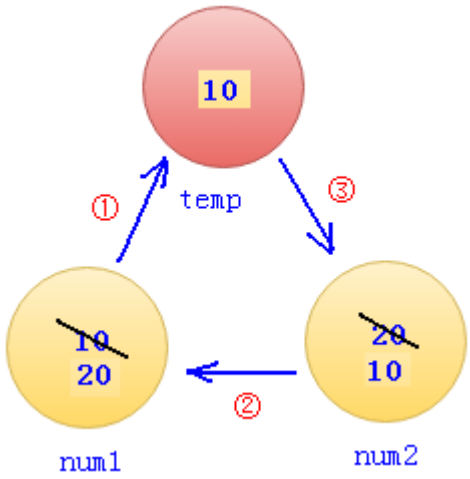


需求：定义每一种数据类型的变量

```
public class VarDemo{

    public static void main(String[] args) {

        // byte类型变量
        byte b = 20;
        System.out.println(b);
```

```
        // short类型变量
        short s = 20;
        System.out.println(s);

        //int类型变量
        int i = 20;
        System.out.println(i);

        //long类型变量,使用L后缀
        long l = 20L;
        System.out.println(l);

        //float类型变量,使用F后缀
        float f = 3.14F;
        System.out.println(f);

        //double类型变量
        double d = 3.14;
        System.out.println(d);

        //char类型变量
        char c = 'A';
        System.out.println(c);

        //boolean类型变量
        boolean bb = true;
        System.out.println(bb);

        //String类型变量
        String str = "你好";
        System.out.println(str);
    }
}
```

需求：交互两个相同类型变量的值



1、把num1的值存储到临时变量temp中去

2、把num2的值赋给num1变量

3、把temp存储的值赋给num2变量

```
public class ChangVarDemo{
```

```
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;
        System.out.println("num1=" + num1);
        System.out.println("num2=" + num2);
        // ---------------------------------
        // 交换过程
        int temp = num1;
        num1 = num2;
        num2 = temp;
        //---------------------------------
        System.out.println("num1=" + num1);
        System.out.println("num2=" + num2);
    }

}
```

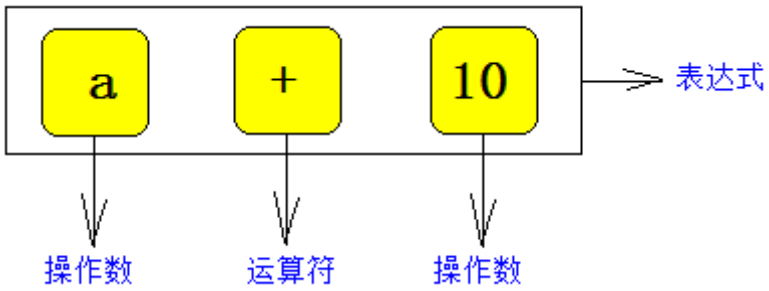
综合案例 : 存储一个手机手机信息

```
public class Test05Var {

    public static void main(String[] args){

        /**
         * 手机信息如下:
         * 名称: iphone xr
         * 价格: 5899.00
         * 重量: 300(g)
         * 颜色: 白
         * 屏幕尺寸: 5.99
         * 是否支持5G : 能
         */

        String phoneName = "iphone xr";
        double phonePrice = 5899.00;
        int phoneWeight = 300;
        char phoneColor = '白';
        float phoneSize = 5.99f;
        boolean isSupport5G = true;


        /**
         * 标识符命名规则:
         * 1> 标识符可以有字母、数字、下划线、$构成
         * 2> 数字不能开头
         * 3> 不能使用java关键字和保留字
         *       // int int = 10; 错误
         */

        // 需求:给班级人数取名字
        int classPersonNum = 100;
        int class_person_num = 100;
        int _classpersonname = 100;
        int $classPersonName = 100;
        int class$person$name = 100;
```

```
        /** 标识符命名规范
         * 1> 使用驼峰命名法（一般第一个单词的首字母小写，后续单词的首字母都大写）
         * 2> 命名时见名知意。
         */

    }
}
```

2.2 表达式（先掌握概念）

表达式（expression），是由数字、运算符、括号、常量、变量等组合以能求得结果的式子，表达式在开发过程中用于计算结果。

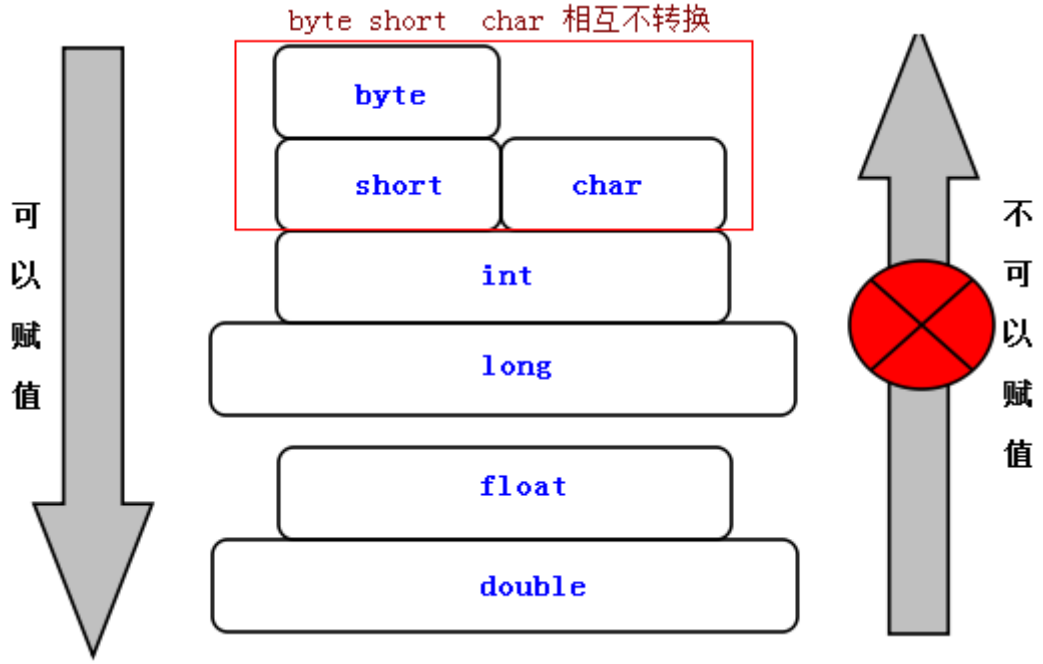


表达式举例（下列a、b、x、y、z都表示变量）。

- a + b
- 3.14 + a
- (x + y) * z + 100

2.3 基本数据类型转换（掌握）

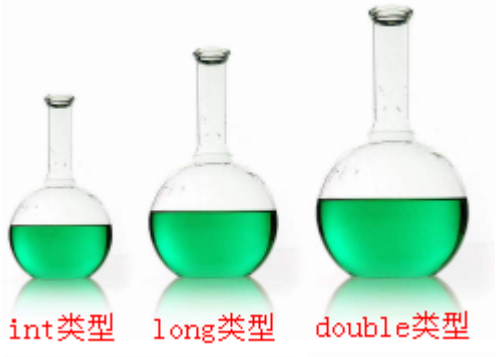
在8大基本数据类型中，boolean不属于数值类型，所以不参与转换，其他类型的转换规则如下图。一般的，byte、short、char三种类型相互之间一般不参与8。



按照转换方式，有两种（注意：boolean类型不参与类型转换）：

- 自动类型转换：范围小的数据类型直接转换成范围大的数据类型 (小 => 大)。

- 强制类型转换：范围大的数据类型强制转换成范围小的数据类型 (大 => 小)。



问题：三个大小不同容器，能相互把盛装的水倒给对方吗？

```
科普了解：
float占4个字节为什么比long占8个字节大？
--->因为底层的实现方式不同
浮点数的32位并不是简单直接表示大小，而是按照一定标准分配的。
第1位，符号位，即S
接下来8位，指数域，即E。
剩下23位，小数域，即M，取值范围为[1 ,2 ) 或[0 , 1)
然后按照公式：V = (-1) ^ S * M * 2 ^ E
也就是说浮点数在内存中的32位不是简单地转换为十进制，而是通过公式来计算而来，通过这个公式虽然，只有4个字节，但浮点数最大值要比长整型的范围要大。
```

2.3.1 自动类型转换（掌握）

自动类型转换，也称为“隐式类型转换，就是把范围小的数据类型直接转换成范围大的数据类型。

转换规则：byte、short、char—>int—>long—>float—>double

注意事项：byte、short、char相互之间不转换，他们参与运算首先转换为int类型

语法格式

```
范围大的数据类型 变量 = 范围小的数据类型值;
```

语法举例：

```
public class Test07TypeConvert {
    public static void main(String[] args){
        // 1> 自动类型转换 ：范围小的数据类型可以直接转换为范围大的数据类型
        int intNum1 = 10;
        long longNum2 = intNum1;

        long longNum3 = 100L;
        float floatNum4 = longNum3;

        float floatNum5 = 3.14f;
        double doubleNum6 = floatNum5;

        /** 总结
         * 自动类型转换 => 范围小的数据类型可以直接转换为范围大的数据类型
         * long类型8字节，float类型4字节，但是float表示的数据范围比long类型大。
         */
```

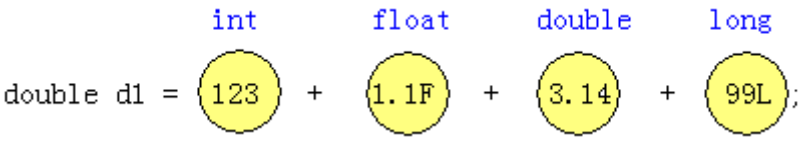
```
        // jvm优化:在java中，可以把范围大的常量直接赋值给范围小(byte,short,char)的变量，只要不超过范围。
        byte byteNum7 = 100;

        // 错误的操作
        // int intNum8 = 100;
        // byte byteNum9 = intNum8;
    }
}
```

2.3.2 自动类型提升（掌握）

当一个算术表达式中，包含多个基本数据类型的常量或变量（boolean除外）时，整个算术表达式的结果类型将在出现自动提升，其规则是：

- **所有的byte、short、char类型被自动提升到int类型，再参与运算**
- **整个表达式的最终结果类型，被提升到表达式中类型最高的类型**

```
System.out.println('a' + 1);// 98

byte b = 22;
b = b + 11;// 编译出错，此时结果类型应该是int

double d1 = 123 + 1.1F + 3.14 + 99L ;
```



结论：算数表达式结果的类型就是其中范围最大的数据类型。

```
public class Test08TypeConvert {
    public static void main(String[] args){
        // 思考:
        char c = 98;
        System.out.println(c);

        // 2> 表达式的自动类型提升
        // 要点1:所有的byte、short、char类型被自动提升到int类型
        byte byteNum1 = 10;
        short shortNum2 = 20;
        int r = byteNum1 + shortNum2;

        // 要点2:整个表达式的最终结果类型，被提升到表达式中类型最高的类型
        float floatNum3 = .14f;
        double doubleNum4 = 1.0;
        double r2 = byteNum1 + shortNum2 + floatNum3 + doubleNum4;

        // 思考:
        char c2 = 'a';
        int num = 1;
        int r3 = c2 + num;
        System.out.println(r3);
```

```
    }
}
```

2.3.3 强制类型转换（掌握）

强制类型转换，也称为“显式类型转换”，就是把范围大的数据类型强制转换成范围小的数据类型。

语法格式：

```
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;
```

注意：一般情况下不建议使用强转，因为强转有可能损失精度

```
public class Test09TypeConvert {
    public static void main(String[] args){
        // 3> 强制类型转换
        float floatNum1 = 3.14f;
        int r1 = (int)floatNum1;
        System.out.println(r1);

        // 应用:根据消费金额计算vip积分(规则：一块钱积2分)
        float totalPrice = 998.88f;
        int vipScore = (int)totalPrice * 2;
        System.out.println(vipScore);

        // 思考题
        char c = 97;
        int num = 1;
        char r = (char)(c + num);
        System.out.println(r);
    }
}
```

===

2.4运算符

对常量和变量进行操作的符号称为运算符

常用运算符：

- 算术运算符
- 赋值运算符
- 比较运算符
- 逻辑运算符
- 三元运算符

2.4.1 算术运算符（掌握）

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; b-- / --b	2

用来四则运算的符号，和小学学习的加减乘除无异。

2.4.1.1 加减乘除余（了解）

对于字符串而言，+符号表示连接操作，任何类型的数据和字符串相连接，结果都是字符串。

```
public class Test07Operation1 {
    public static void main(String[] args) {
        // 算术运算符
        // +  -  *
        // /
        // %
        // + 作为算术运算符
        int intNum1 = 10;
        int intNum2 = 20;
        int r1 = intNum1 + intNum2;
        System.out.println(r1);

        // + 作为字符串连接符
        // + 两边的表达式或值只要有一个是字符串，+ 就是字符串连接符
        int intNum3 = 30;
        String str = "intNum3:" + intNum3;
        System.out.println(str);

        // / 整除操作 :
        int r2 = 5 / 2;
        System.out.println("r2 = " + r2);
        int num = 5;
        double r3 = num * 1.0 / 2;
        System.out.println("r3 = " + r3);
        /** 总结
         * 1>如果/两边的表达式都是整形，结果一定是整形
```

```
         * 2> 0不能作为除数
         */

        // % : 取模,求余数
        System.out.println(5 % 2);
        System.out.println(1 % 5);
        // 需求: 给定47天，问47天中有__月(30)__天
        int days = 47;
        int month,day;
        month = days / 30;
        // day = days - month * 30;
        day = days % 30;
        System.out.println(month);
        System.out.println(day);
        /**
         * 总结
         * 1> 模谁结果不会超过谁
         */
    }
}
```

2.4.1.2 自增和自减（掌握）

自增：++，递增操作符，使变量值增加1，有前置和后置之分，只能操作变量。

自减：-- ，递减操作符，使变量值减去1，有前置和后置之分，只能操作变量。

自增和自减具体操作是一样的，仅仅是一个是加1，一个是减1而已，现在单讲++。

代码 result ++和 ++result，结果都是result变量的值加1。

唯一的区别是：

- 前置（++result）：表示对result加1之后的结果进行运算
- 后置（result++）：表示对result变量加1之前的值（原始值）进行运算。

如果仅仅执行简单的递增操作（只写result++或++result），那么选用任意一个都可以。

```
public class Test07Operation2 {
    public static void main(String[] args) {
        // 自加
        // i++ / i-- : 遇到i++/i--,i先参与运算，运算后自增1/自减1
        // 情况1（必须掌握）:
        /*int i = 10;
        i++;
        System.out.println("i = " + i);*/

        // 情况2（必须掌握）:
        /*int i = 10;
        int j;
        j = i++;
        System.out.println("i = " + i);
        System.out.println("j = " + j);*/
    }
}
```

比较权威的解释：

- ++a表示取a的地址，增加它的内容，然后把值放在寄存器中；
- a++表示取a的地址，把它的值装入寄存器，然后增加内存中的a的值；

如果不理解什么是寄存器，简单记住，都可以表示当前变量自身加1，区别是：

- 前置++：先增加后使用
- 后置++：先使用后增加

2.4.2 赋值运算符（掌握）

```
变量 = 表达式的值或者常量值
```

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

```
public class AssigningOperatorDemo {
    public static void main(String[] args) {
        // 把常量17赋值给int类型的变量a
        int a = 17;
        System.out.println("a=" + a);
        // += 把左边和右边的数据进行运算，最后赋值左边变量
        a += 10;// 相当于a = a + 10
        System.out.println("a=" + a);

        short s = 5;
        s += 2; //底层相当于 s = (short) (s + 2)
        System.out.println("s=" + s);
    }
}
```

2.4.3 比较运算符（掌握）

用于比较变量或常量、表达式之间的大小关系，其结果是boolean类型（要么为true，要么为false）。

其操作格式为：

```
boolean  result  = 表达式A  比较运算符  表达式B;
```

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

注意：>=符号，表示大于或者等于。

```
public class ComparisonOperatorDemo {
    public static void main(String[] args) {
        //直接操作常量
        System.out.println(10 > 5);//true
        System.out.println(10 >= 5);//true
        System.out.println(10 >= 10);//true
        System.out.println(10 < 5);//false
        System.out.println(10 <= 5);//false
        System.out.println(10 <= 10);//true
        System.out.println(10 == 10);//true
        System.out.println(10 != 10);//false

        //使用变量操作
        int a = 10;
        int b = 5;
        boolean result = a > b;
        System.out.println(result);//true
    }
}
```

2.4.4 三元运算符（掌握）

三元运算符，表示有三个元素参与的表达式，所以又称为三目运算符，其语义表示if-else（如果什么情况就做什么，否则做什么）。 如果...那么...否则...

语法格式：

```
数据类型 变量 = boolean表达式 ?  结果A : 结果B;
```

- 如果boolean表达式结果：
  - 为true，则三元运算符的结果是结果A；
  - 为false，则三元运算符的结果是结果B；

注：三元运算符**必须定义变量接受**运算的结果，否则报错

三元运算符结果的类型由结果A和结果B来决定的，结果A和结果B的类型是相同的。

需求1：判断一个数99是不是偶数

```
public class TernaryOperatorDemo1{
    public static void main(String[] args) {
        int a = 99;
        String result = a % 2 == 0 ? "偶数" : "奇数";
        System.out.println(result);
    }
}
```

需求2：求99和20两个数中的最大值

```
public class TernaryOperatorDemo2{
    public static void main(String[] args) {
        int a = 99;
        int b = 20;
        int result = a > b ? a : b;
        System.out.println("最大值："+result);
    }
}
```

需求3：一共55条数据，每页10条数据，一共分多少页


```
public class TernaryOperatorDemo3{
    public static void main(String[] args) {
        int totalCount = 54;
        int pageSize = 10;
        int totalPage = totalCount % pageSize == 0
                    ? totalCount / pageSize
                    : totalCount / pageSize + 1;
        System.out.println(totalPage);
    }
}
```

2.4.5 逻辑运算符（掌握）

逻辑运算符用于连接两个boolean表达式，结果也是boolean类型的。

语法格式为：

```
boolean  result = boolean表达式A  逻辑运算符  boolean表达式B;
```

运算规则如下：

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	false
&&	短路与	false&&true	false
||	短路或	false||true	true

规律：

- 非：取反，！true则false,！false则true
- 与：有false则false
- 或：有true则true
- 异或：^ 相同则false,不同则true

```
& 与运算，可以理解为 "并，并且"
true  & true  => true
true  & false => false
false & true  => false
false & false => false
总结 ： & 运算，只要两边的表达式有一个为false，结果就为false

&& 短路与
&& 运算，只要两边的表达式有一个为false，结果就为false，如果第一个表达式为false，后续表达式不再运算；

| 或运算，可以理解为 "或，或者"
true  | true  => true
true  | false => true
false | true  => true
false | false => false
总结 ： | 运算，只要两边的表达式有一个为true，结果就为true

|| 短路或
|| 短路或运算，只要两边的表达式有一个为true，结果就为true，如果第一个表达式为true，后续表达式不再运算；

! 非运算，可以理解为 取反
!true = false
!false = true
```

2.4.5.1 基本使用（掌握）

```
public class LogicalOperatorDemo1 {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int c = 30;
        //与操作
        System.out.println((a > b) & (a > c));// false  &   false
        System.out.println((a > b) & (a < c)); // false &   true
        System.out.println((a < b) & (a > c)); // true  &   false
        System.out.println((a < b) & (a < c)); // true  &   true

        //或操作
        System.out.println((a > b) | (a > c));// false  |   false
        System.out.println((a > b) | (a < c)); // false |   true
        System.out.println((a < b) | (a > c)); // true  |   false
        System.out.println((a < b) | (a < c)); // true  |   true

        //相反操作
        System.out.println((a > b)); // false
        System.out.println(!(a > b)); // !false
        System.out.println(!!(a > b)); // !!false
    }
}
```

2.4.5.2 &和&&的区别（掌握）

&： &左边表达式无论真假，&右边表达式都进行运算；

&& ：如果&&左边表达式为真，&&右边表达式参与运算，否则&&右边表达式不参与运算，故称短路与。

| 和 || 的区别同理，||，左边为真，右边不参与运算。

```
public class Test05LogicOperator2 {
    public static void main(String[] args) {
        // 逻辑运算符
        // 短路运算 && ||
        int m = 10;
        int n = 20;
        // boolean r1 = false && false;
        boolean r1 = (m > n) && (++m > 1);
        System.out.println("r1 = " + r1);
        System.out.println("m = " + m);


        boolean r2 = (m > n) | (m < 1);
        System.out.println("r2 = " + r2);
    }
}
```

上述代码，一行一行的测试，测试完，注释该行代码。

2.4.6 运算优先级

表达式的运算都是有优先级的，基本上和数学中的优先级类似，这里需要注意的是，赋值符号。

注意：赋值符号最后运算的，并且是从右向左运算的。

优先级	运算符	类	结合性
1	()	括号运算符	由左至右
1	[]	方括号运算符	由左至右
2	!、+（正号）、-（负号）	一元运算符	由右至左
2	~	位逻辑运算符	由右至左
2	++、--	递增与递减运算符	由右至左
3	*、/、%	算术运算符	由左至右
4	+、-	算术运算符	由左至右
5	<<、>>	位左移、右移运算符	由左至右
6	>、>=、<、<=	关系运算符	由左至右
7	==、!=	关系运算符	由左至右
8	&（位运算符 AND）	位逻辑运算符	由左至右
9	^（位运算符号 XOR）	位逻辑运算符	由左至右
10	|（位运算符号 OR）	位逻辑运算符	由左至右
11	&&	逻辑运算符	由左至右
12	||	逻辑运算符	由左至右
13	?:	条件运算符	由右至左
14	=	赋值运算符	由右至左

（）的优先级最高，赋值运算符优先级最低

赋值运算符的运算方向从右向左