1. JDK,JRE,JVM 三者之间的关系，以及 JDK、JRE 包含的主要结构有哪些

2. 为什么要配置 path 环境变量？如何配置？

3.创建如下的类，使得运行的话可以输出：

```
姓名：习大大
性别：男
家庭住址：北京中南海
```

4.编译和运行上述代码的指令

5. 标识符的命名规则有哪些

1. JDK,JRE,JVM 三者之间的关系，以及 JDK、JRE 包含的主要结构有哪些

JDK = JRE + 开发工具

JRE = 常用类库 + JVM

2. 为什么要配置 path 环境变量？如何配置？

path:windows 操作系统执行命名时所要搜寻的路径。

path = %JAVA_HOME%\bin;

3.创建如下的类，使得运行的话可以输出：

```
姓名：习大大

性别：男
家庭住址：北京中南海
```

创建一个 java 文件：Test.java

```
class Test{
    public static void main(String args[]){
        System.out.println("姓名：习大大\\n");
        System.out.println();
        System.out.println("性别：");
        System.out.println("家庭住址：");
    }
```

4.编译和运行上述代码的指令

javac 文件名.java

java 类名

4.Java语言应用的领域：

方向一：企业级应用

方向二：移动Android客户端

5.Java语言的特点

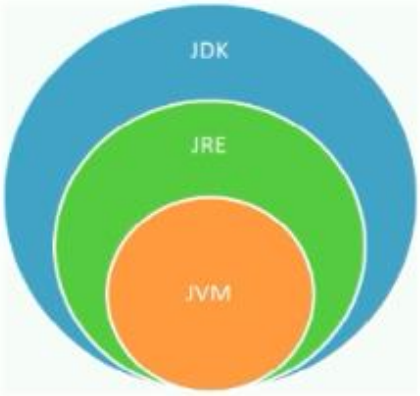
① 面向对象性：两大要素：类、对象；三大特征：封装、继承、多态

② 健壮性：去除了C/C++指针；添加了垃圾的自动回收机制。--> java仍然可能出现内存的溢出和内存的泄漏

③ 跨平台性：write once , run anywhere "一次编译，到处运行" -->JVM实现

不同的操作系统，我们需要安装不同的JVM。

1.3 Java语言的环境搭建：JDK、JRE、JVM关系



- JDK = JRE + 开发工具集（例如Javac编译工具等）
- JRE = JVM + Java SE标准类库

6.2 JDK的下载、安装

下载：www.oracle.com

安装：傻瓜式安装

注意：所有软件安装的路径中，不能包含中文和空格！！！

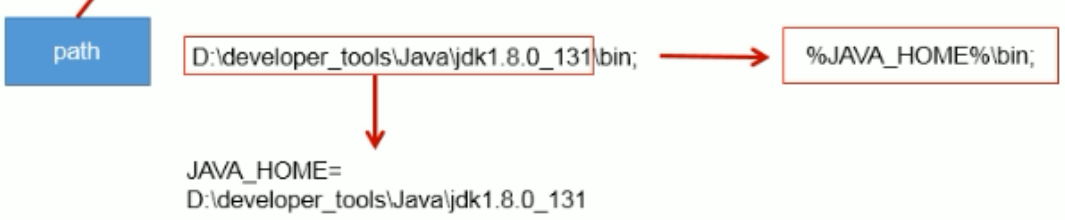
6.3 环境变量的配置

6.3.1 为什么要配置环境变量？

为了保证在任何文件目录下都可以执行java的开发指令，比如：javac.exe java.exe

6.3.2 如何配置环境变量？

path:window操作系统执行命令时，所要搜寻的路径
目的：希望D:\developer_tools\Java\jdk1.8.0_131路径下的命令可以在任何文件路径下执行

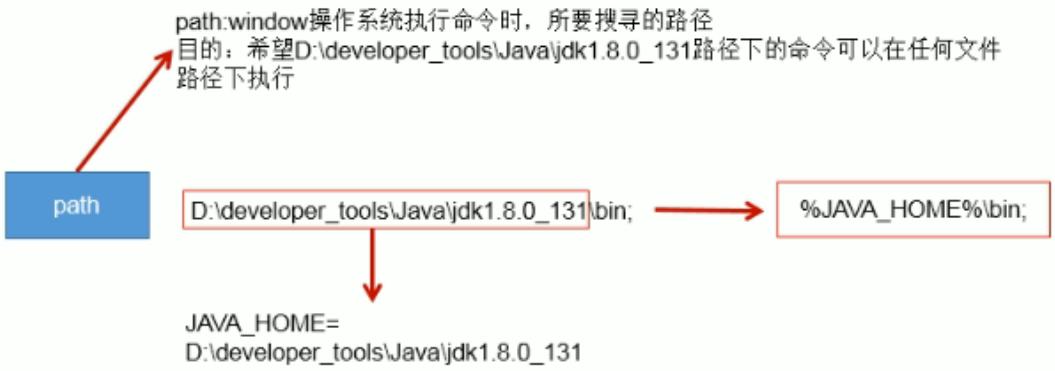


JAVA_HOME=
D:\developer_tools\Java\jdk1.8.0_131

6.3.1 为什么要配置环境变量？

为了保证在任何文件目录下都可以执行java的开发指令。比如：javac.exe java.exe

6.3.2 如何配置环境变量？



补充：何为path环境变量？

path:windows操作系统在执行命名时所要搜寻的路径！

7.开发体验—HelloWorld

创建java文件：HelloWorld.java

```
class HelloChina{

	public static void main(String[] args){

		System.out.println("hello world!你好！ ");

	}

}
```

7.2 编译：

```
javac HelloWorld.java
```

编译完以后，会生成一个或多个.class结尾的字节码文件。这些文件名即为java类名

7.3 运行：

```
java HelloChina
```

1.5 常见问题及解决方法

```
D:\>javac Test1.java
javac: 找不到文件: Test1.java
用法: javac <options> <source files>
-help 用于列出可能的选项
```

- 源文件名不存在或者写错
- 当前路径错误
- 后缀名隐藏问题

```
D:\>java Test1
错误: 找不到或无法加载主类 Test1
```

- 类文件名写错，尤其文件名与类名不一致时，要小心
- 类文件不在当前路径下，或者不在classpath指定路径下

1.5 常见问题及解决方法

```
D:\>javac Test.java
Test.java:1: 错误: 类Test1是公共的, 应在名为 Test1.java 的文件中声明
public class Test1{
       ^
1 个错误
```

- 声明为public的类应与文件名一致，否知编译失败

```
D:\>javac Test.java
Test.java:3: 错误: 需要';'
                System.out.println("hello")
                                          ^
1 个错误
```

- 编译失败，注意错误出现的行数，再到源代码中指定位置改错

① 编写：编写java代码，声明在.java结尾的文件中：源文件

② 编译：使用javac.exe指令编译java程序。格式：javac 源文件名.java

③ 运行：使用java.exe指令解释运行java类。格式：java 类名

2. 一个java源文件中可以声明多个java类

3. 源文件中声明几个java类，就会在编译以后生成几个字节码文件。而且字节码文件名即为java类名

4. main方法是程序的入口。格式是固定的。

5. 输出语句格式是固定的。

```
System.out.println()//在输出数据以后，会换行
System.out.print()
```

6. 在一个java源文件中最多只能有一个类声明为public的。

要求：声明为public的类的类名必须与java源文件名一致。

10.注释(Comment)

分类：单行注释、多行注释、文档注释

作用：

单行注释、多行注释：① 增加程序的可读性 ② 协助调试程序

文档注释：为了javadoc所解析，显示所必须的注释信息

特点：

①单行注释和多行注释，在使用javac.exe命令进行编译时，不会参与编译，进而不会出现在.class字节码文件中

②多个多行注释不能嵌套使用。

③文档注释：注释内容可以被JDK提供的工具 javadoc 所解析，生成一套以网页文件形式体现的该程序的说明文档

11.Java API 文档：

API:application programming interface.

API 文档：关于如何使用API的说明信息。类似于：药的说明书、《新华字典》

1.8 良好的编程风格

- **正确的注释和注释风格**
  - 使用文档注释来注释整个类或整个方法。
  - 如果注释方法中的某一个步骤，使用单行或多行注释。
- **正确的缩进和空白**
  - 使用一次tab操作，实现缩进
  - 运算符两边习惯性各加一个空格。比如：2 + 4 * 5。
- **块的风格**
  - Java API 源代码选择了行尾风格

```
public class Test {
    public static void main(String[] args){
        System.out.println("Block Style!");
    }
}
```

行尾风格

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Style!");
    }
}
```

次行风格

标题 关键字与标识符

1.java关键字的使用

定义：**被Java语言赋予了特殊含义，用做专门用途的字符串（单词）**

特点：java中的关键字都是小写的。

2.保留字：

现Java版本尚未使用，但以后版本可能会作为关键字使用。自己命名标识符时要避免使用这些保留字 :goto 、const

3.标识符的使用

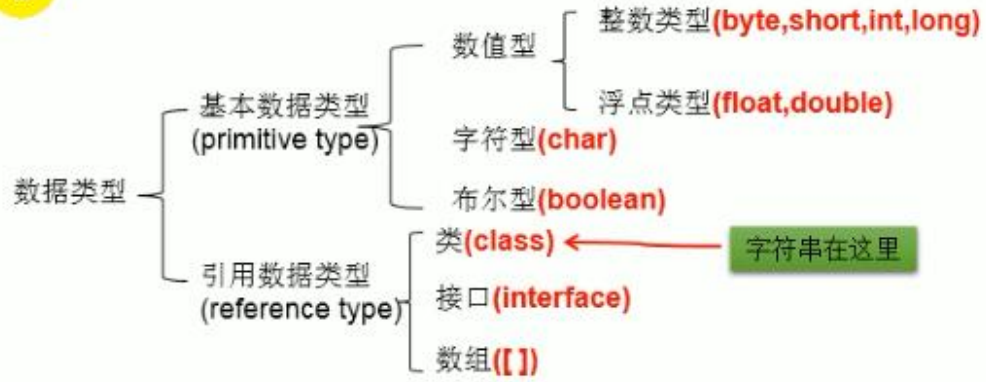
定义：**凡是自己可以起名字的地方都叫标识符。**

涉及到的结构：

类名、接口名、方法名、变量名、常量名、包名、....

规则：(必须要遵守，否则，编译不通过)

1.1 按数据类型分类



1.2 按声明的位置分类(了解)

- 在方法体外，类体内声明的变量称为成员变量。
- 在方法体内部声明的变量称为局部变量。

2.定义变量的格式：

数据类型　变量名 = 初始化值;

数据类型 变量名;

变量名 = 初始化值;

3.变量使用的注意点：

- Java中每个变量必须先声明，后使用
- 使用变量名来访问这块区域的数据
- 变量的作用域：其定义所在的一对{ }内
- 变量只在其作用域内才效
- 同一个作用域内，不能定义重名的变量

```
/*
基本数据类型变量间的运算：
1. 自动类型提升：当容量小的数据类型变量与容量大的数据类型变量做运算时，结果为容量大的数据类型。
说明：
① 此时的运算只包含基本数据类型中的7种，唯独不包括boolean类型！
② byte 、short、char ---> int ---> long ---> float ----> double


2. 强制类型转换：是自动类型提升运算的逆运算。


注意：此时的容量的大小，不是指占用内存的空间的大小，而是指表数范围的大小。
      long(8个字节) 相较于 float(4个字节) 算容量小的。

*/
```

2.3 变 量

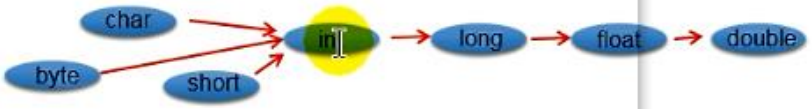
尚硅

强制类型转换

- 自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型。使用时要加上强制转换符：()，但可能造成精度降低或溢出,格外要注意。
- 通常，字符串不能直接转换为基本类型，但通过基本类型对应的包装类则可以实现把字符串转换成基本类型。
  - 如： String a = “43”; int i = Integer.parseInt(a);
  - boolean类型不可以转换为其它的数据类型。

基本数据类型转换

- **自动类型转换**：容量小的类型自动转换为容量大的数据类型。数据类型按容量大小排序为：



- 有多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后再进行计算。
- byte,short,char之间不会相互转换，他们三者在计算时首先转换为int类型。
- boolean类型不能与其它数据类型运算。
- 当把任何基本数据类型的值和字符串(String)进行连接运算时(+)，基本数据类型的值将自动转化为字符串(String)类型。

```
/*
基本数据类型变量间的运算：
1. 自动类型提升：当容量小的数据类型变量与容量大的数据类型变量做运算时，结果为容量大的数据类型。
说明：
① 此时的运算只包含基本数据类型中的7种，唯独不包括boolean类型！
② byte、short、char ---> int ---> long ---> float ----> double
 特别的，byte、short、char之间做运算，结果为int型。


2. 强制类型转换：是自动类型提升运算的逆运算。


注意：此时的容量的大小，不是指占用内存的空间的大小，而是指表数范围的大小。
      long(8个字节) 相较于 float(4个字节) 算容量小的。

*/
```

```
/*
基本数据类型变量间的运算：
强制类型转换：将容量大的数据类型的变量转换为容量小的数据类型的变量。
需要使用强转符：()

注意：使用强转符，可能会损失精度


*/
```

字符串类型：String

- String不是基本数据类型，属于引用数据类型
- 使用方式与基本数据类型一致。例如：String str = “abcd”;
- 一个字符串可以串接另一个字符串，也可以直接串接其他类型的数据。例如：

```
str = str + “xyz” ;
int n = 100;
str = str + n;
```

```
/*
String:字符串
1.String属于引用数据类型中的类，不是基本数据类型
2.使用一对""来声明
3.String可以与8种基本数据类型的变量做运算。运算的结果为：String型

*/
```


练习1

```
String str1 = 4;        //判断对错：no
String str2 = 3.5f + “”;          //判断str2对错：yes
System.out.println(str2);       //输出：3.5
System.out .println(3+4+“Hello!”);    //输出：7Hello!
System.out.println(“Hello!”+3+4);     //输出：Hello!34
System.out.println(‘a’+1+“Hello!”);   //输出：98Hello!
System.out.println(“Hello”+‘a’+1);    //输出：Helloa1
```

练习2

判断是否能通过编译

```
1）short  s = 5;
    s = s-2;                    //判断：no
2）  byte b = 3;
     b = b + 4;                 //判断：no
     b = (byte)(b+4);           //判断：yes
3）char c = 'a';
    int  i = 5;
    float d = .314F;
    double result = c+i+d;      //判断：yes
4）  byte b = 5;
     short s = 3;
     short t = s + b;           //判断：no
```


关于进制

- 所有数字在计算机底层都以二进制形式存在。
- 对于整数，有四种表示方式：
  - 二进制(binary)：0,1 ，满2进1.以0b或0B开头。
  - 十进制(decimal)：0-9 ，满10进1.
  - 八进制(octal)：0-7 ，满8进1. 以数字0开头表示。
  - 十六进制(hex)：0-9及A-F，满16进1. 以0x或0X开头表示。此处的A-F不区分大小写。
    如：0x21AF +1= 0X21B0

十进制	十六进制	八进制	二进制
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000

十进制	十六进制	八进制	二进制
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111
16	10	20	10000
17	11	21	10001

二进制

- Java整数常量默认是int类型，当用二进制定义整数时，其第32位是符号位；当是long类型时，二进制默认占64位，第64位是符号位
- 二进制的整数有如下三种形式：
  - 原码：直接将一个数值换成二进制数。最高位是符号位
  - 负数的反码：是对原码按位取反，只是最高位（符号位）确定为1。
  - 负数的补码：其反码加1。

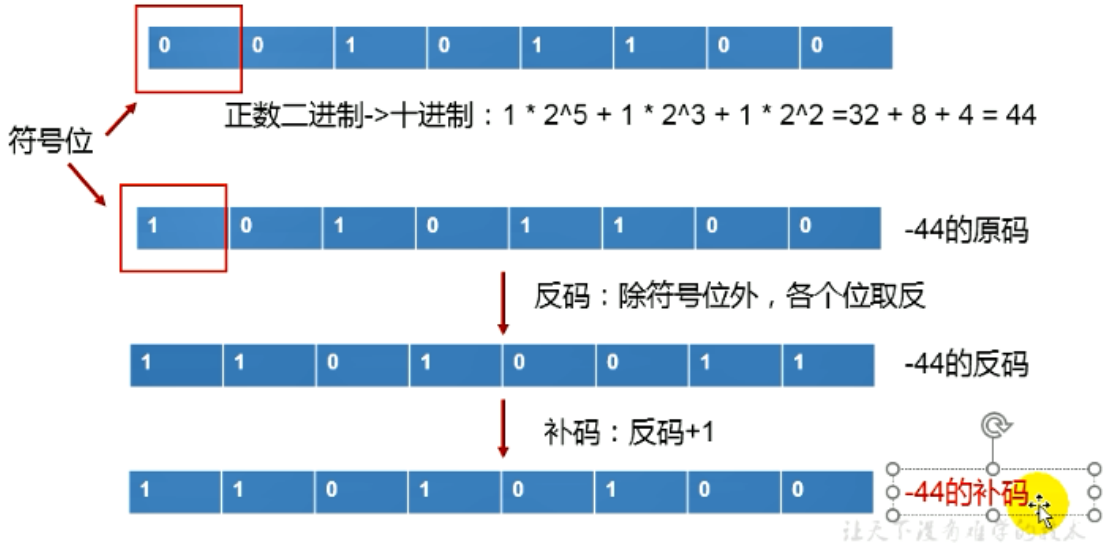
- 计算机以二进制补码的形式保存所有的整数。
  - 正数的原码、反码、补码都相同
  - 负数的补码是其反码+1

2.3 变量之进制

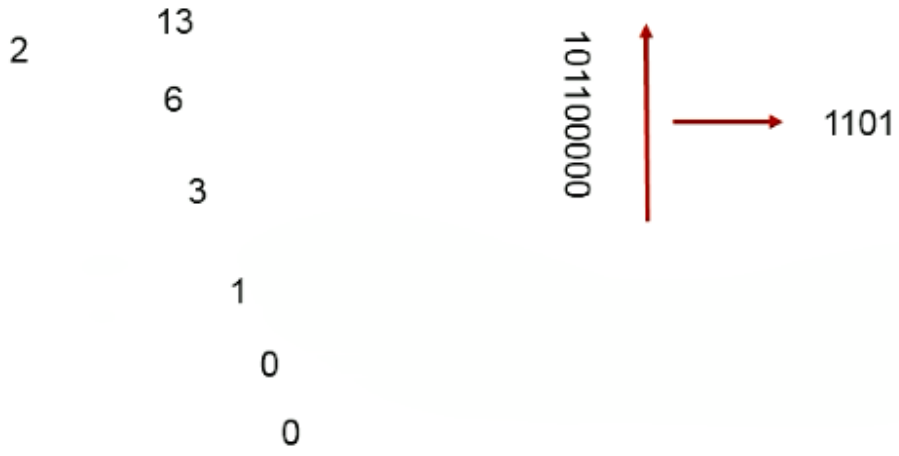
●为什么要使用原码、反码、补码表示形式呢？

计算机辨别“符号位”显然会让计算机的基础电路设计变得十分复杂! 于是人们想出了将符号位也参与运算的方法. 我们知道, 根据运算法则减去一个正数等于加上一个负数, 即: 1-1 = 1 + (-1) = 0 , 所以机器可以只有加法而没有减法, 这样计算机运算的设计就更简单了。

$1-1 = 1 + (-1) = [0000\ 0001]_{原} + [1000\ 0001]_{原} = [0000\ 0001]_{补} + [1111\ 1111]_{补} = [0000\ 0000]_{补} = [0000\ 0000]_{原}$



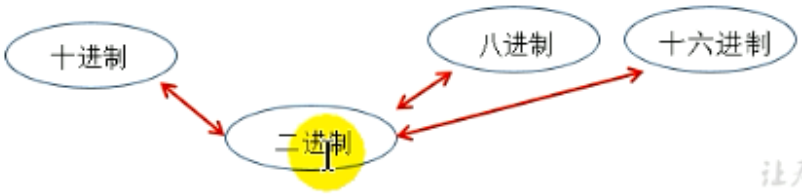
十进制转二进制：除2取余的逆



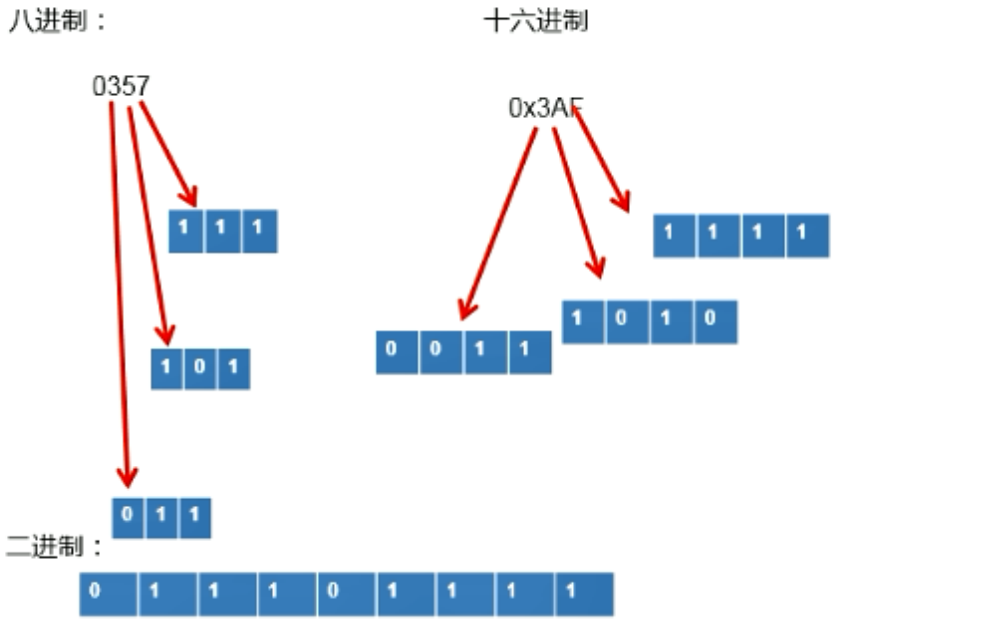
2.3 变量之进制间的转换

进制间转化

- 进制的基本转换
  - 十进制 二进制互转
    - ✓ 二进制转成十进制 乘以2的幂数
    - ✓ 十进制转成二进制 除以2取余数
  - 二进制 八进制互转
  - 二进制 十六进制互转
  - 十进制 八进制互转
  - 十进制 十六进制互转



2.3 变量之进制



2.4 运算符

运算符是一种特殊的符号，用以表示数据的运算、赋值和比较等。

- 算术运算符
- 赋值运算符
- 比较运算符（关系运算符）
- 逻辑运算符
- 位运算符
- 三元运算符

2.4.1 运算符：算术运算符

运算符	运算	范例	结果
+	正号	+3	3
-	负号	b=4; -b	-4
+	加	5+5	10
-	减	6-4	2
*	乘	3*4	12
/	除	5/5	1
%	取模(取余)	7%5	2
++ ++	自增（前）：先运算后取值 自增（后）：先取值后运算	a=2;b=++a; a=2;b=a++;	a=3;b=3 a=3;b=2
- - - -	自减（前）：先运算后取值 自减（后）：先取值后运算	a=2;b=- -a a=2;b=a- -	a=1;b=1 a=1;b=2
+	字符串相加	“He”+”llo”	“Hello”

2.4.1 运算符：算术运算符

练习1：算术运算符：自加、自减

```
public class TestSign{
    public static void main(String[] args){
        int i1 = 10;int i2 = 20;
        int i = i1++;
        System.out.print("i="+i);
        System.out.println("i1="+i1);
        i = ++i1;
        System.out.print("i="+i);
        System.out.println("i1="+i1);
        i = i2--;
        System.out.print("i="+i);
        System.out.println("i2="+i2);
        i = --i2;
        System.out.print("i="+i);
        System.out.println("i2="+i2);
    }
}
```

输出：

i= i1=
i= i1=
i= i2=
i= i2=

2.4.1 运算符：算术运算符

练习2

随意给出一个整数，打印显示它的个位数，十位数，百位数的值。
格式如下：
数字xxx的情况如下：
个位数：
十位数：
百位数：

例如：
数字153的情况如下：
个位数：3
十位数：5
百位数：1

2.4.2 运算符：赋值运算符

●符号：=

- 当“=”两侧数据类型不一致时，可以使用自动类型转换或使用强制类型转换原则进行处理。
- 支持连续赋值。

●扩展赋值运算符： +=, -=, *=, /=, %=

2.4.5 运算符：位运算符

位运算符的细节	
<<	空位补0，被移除的高位丢弃，空缺位补0。
>>	被移位的二进制最高位是0，右移后，空缺位补0； 最高位是1，空缺位补1。
>>>	被移位二进制最高位无论是0或者是1，空缺位都用0补。
&	二进制位进行&运算，只有1&1时结果是1，否则是0;
|	二进制位进行 | 运算，只有0 | 0时结果是0，否则是1;
^	相同二进制位进行 ^ 运算，结果是0；1^1=0 , 0^0=0 不相同二进制位 ^ 运算结果是1。1^0=1 , 0^1=1
~	正数取反，各二进制码按补码各位取反 负数取反，各二进制码按补码各位取反