1、环境变量

2、第一个程序；

3、系统输出语句；

4、**变量**

为什么要有变量；-- 计算 数据（0~9）、字符|字符串、布尔值、

数据类型：《重点》

人脑：计算区域、存储区域

电脑：CPU、内存

存储计算后的结果数据（中间数据）

有效范围、有效期：作用域；

输入的数据，也是映射到大脑皮层；

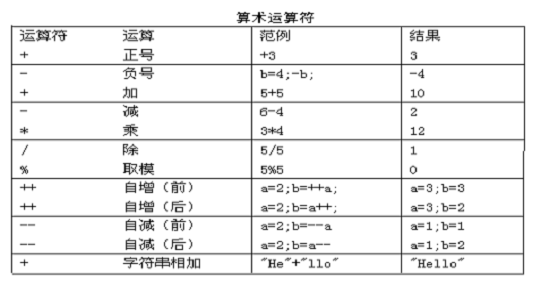
变量：在内存中开僻出的一个空间（区域），用来 存储、暂时的；该区域可以重复存放不同数据；

数据类型 变量名 ；//申明变量；

变量名 = 数据 ；//赋值语句；赋值符号；

**变量命名规范**

1. 运算符

算术运算符： + - \* / %（取余|取模） ；

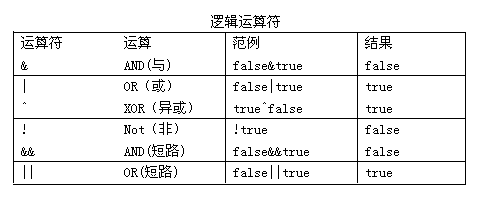
赋值运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 运算 | 范例 | 结果 |
| = | 赋值 | a=3,b=2 | a=3,b=2 |
| += | 加等于 | a=3,b=3;a+=b; | a=5,b=2; |
| -= | 减等于 | a=3,b=2,a-=b; | a=1,b=2; |
| \*= | 乘等于 | a=3,b=2,a\*=b; | a=6,b=2 |
| /= | 除等于 | a=3,b=2,a/=b; | a=1,b=2; |
| %= | 模等于 | a=3,b=2,a%=b; | a=1,b=2 |

比较运算符|关系运算符： > 、<、 ==、>=、 <=、 !=；

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 运算 | 例子 | 结果 |
| == | 相等于 | 4= =3 | false |
| != | 不等于 | 4!= 3 | true |
| < | 小于 | 4 < 3 | flase |
| > | 大于 | 4>3 | true |
| <= | 小于等于 | 4<=3 | false |
| >= | 大于等于 | 4>=3 | true |
| Instanceof | 检查是否是类的对象 | "hello"instanceof String | true |

逻辑运算符： &&(and) 、||(or)、 !(not)；



**优先级：**



1. 条件控制语句

如果 （条件满足） {

做什么事件//执行语句；代码块

}

否则{

做其它事件

}

Int i =scanner.nextInt();

If(i < 5){

}else{

// i>=5

}

* 多重条件控制语句：

If (条件表达式1){

}else if(条件表达式2){

}else{

}

**public** **static** **void** main(String[] args) {

**int** x = 8;

**if** (x == 1) {

System.*out*.println("星期一");

} **else** **if** (x == 2) {

System.*out*.println("星期二");

} **else** **if** (x == 3) {

System.*out*.println("星期三");

} **else** **if** (x == 4) {

System.*out*.println("星期四");

} **else** **if** (x == 5) {

System.*out*.println("星期五");

} **else** **if** (x == 6) {

System.*out*.println("星期六");

} **else** **if** (x == 7) {

System.*out*.println("星期日");

} **else** {

System.*out*.println("请输入数字1-7");

}

~~If (条件表达式1){~~

~~}~~

~~if(！条件表达式1 && 条件表达式2){~~

~~}~~

这种方式：代码复杂；计算运行效率低；

* 嵌套语句

条件表达式的代码块中（大括号中），再放一个条件表达式；

If(条件表达式1){

If(条件表达式2){

}

}

* Switch语句

Switch(待比较的表达式)

只能做等值比较；

Case 关键字；

Case 比较值 **：{**

**}**

Break 关键字；

中断、跳出当前switch语句；

Default 关键字；

|  |
| --- |
| switch(表达式)  {  case 取值1:  执行语句；  break；  case 取值2:  执行语句；  break；  …...  default:  执行语句；  break；  } |

1. 循环语句；

* While循环语句
* 初始化语句、比较语句、变量叠代；

While(比较语句){

//循环体；需要重复执行的代码；

//添加退出机制（步长）

//或者用break退出;

}

|  |
| --- |
| public static void main(String[] args){  int x=1;  while(x<100){    if(x%2!=0){  System.out.print(x);  }  x++;  }  System.out.println();  } |

* Do While

//先执行一次，再判断是否继续循环

// while: 先判断是否继续循环，满足条件才执行

Dowhile最后的分号“；”不要漏掉；

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** x = 0, y = 0;  **do** {  System.*out*.println(x);  x++;  } **while** (x < 0);  // do while do会先执行一次,不管是否满足循环条件。  **while** (y < 0) {  System.*out*.println(y);  y++;  }  } |

**循环语句，也可以嵌套；**

* For循环

把数据初始化、比较语句、叠代语句，放在一起；

For(数据初始化 **;** 比较语句 **;** 叠代语句){

//循环的内容

}

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 1-100之间 7的倍数的个数,并打印。  **int** count = 0;  **for** (**int** x = 0; x <= 100; x++) {  **if** (x % 7 == 0) {  System.*out*.println(x);  count++;  }  }  System.*out*.println(count);  } |

Break:中断或者退出循环体;

Continue ： 本次循环中，关键字后面的不执行，直接转到下次循环;

**Break 和continue，只在当前循环中有效；**

* **数组**

方便管理大量数据（变量）

数据类型 [] 数组名；//申明；

两种初始化

**基本初始化**

* New 数据类型 [数组长度]

数组中，没有任何数据值，只默认值 ；

带数据的初始化

* New 数据类型[]{数据1，数据2}；

此种情况，不用指定长度，使用后面数据个数作为长度；

给数组赋值

**数组名[索引|下标] = 数据**

/\*

冒泡排序。

比较方式：相邻两个元素进行比较。如果满足条件就进行位置置换。

原理：内循环结束一次，最值出现在尾角标位置。

\*/

**public** **static** **void** bubbleSort(**int**[] arr)

{

**for**(**int** x=0; x<arr.length-1; x++)

{

**for**(**int** y=0; y<arr.length-x-1; y++)//-x:让每次参与比较的元减。

//-1:避免角标越界。

{

**if**(arr[y]>arr[y+1])

{

**int** temp = arr[y];

arr[y] = arr[y+1];

arr[y+1] = temp;

}

}

}

}

使用数组：

**数组名[索引|下标]**

***二维数组***

条件

人与人交流：自然的语言

人与机交流：程序语言；

计算机内部：计算机语言；01