前置教程

JDK的安装目录

目录名称 说明

1. bin 该路径下存放了JDK的各种工具命令。**javac**和**jav**就放在这个目录
2. conf 该路径下存放了JDK的相关配置
3. include 该路径下存放了一些平台特定的头文件
4. jmods 该路径存放了JDK各种模块
5. legal 该路径存放了JDK各种模块的授权文档
6. lib 该路径存放了JDK工具的一些补充JAR包

基础语法

注释

```
//单行注释
/* 多行
            注释
 */
```

关键字

关键字的字母全部小写

常量

1. 字符串常量：用双引号的内容
2. 整数常量
3. 小数常量
4. 字符常量：用单引号的内容
5. 布尔值和空值null

数据类型

最小信息单元：位(bit)，最小存储单元：字节(Byte|B)

1. 基本数据类型
    1. 数值型
        1. 整数(byte 128,short 32768,long 2^63)
        2. 浮点数(float E38,double E308)
        3. 字符(char 65535)
    2. 非数值型
        1. 布尔(boolean)
2. 引用数据类型
    1. 类(class)
    2. 接口(interface)
    3. 数组([])

变量

变量是内存中的一小块区域 强类型语音：变量名、数据类型、变量值 变量使用事项：

1. 一个方法内一个变量名最多只能定义一次
2. 定义变量时需要赋值，且在long和float赋值时要加L和F，因为整形默认值为int，浮点型默认值为double。

标识符

概念

1. 由数字、字母、下划线(_)及美元符($)组成

2. 不能以数字开头
3. 不能是关键字
4. 区分大小写 约定 小驼峰命名法：方法，变量
5. 标识符是一个单词的时候，首字母小写
6. 标识符由多个单词组成时，第一个单词首字母小写，其他单词首字母大写 大驼峰命名法：类
7. 标识符是一个单词的时候，首字母大写
8. 标识符由多个单词组成的时候，每个单词首字母大写

类型转换

自动类型转换

double d = 10;

强制类型转换

把一个表示数据范围大的数值或变量赋值给另一个表示数据范围小的变量，一般不使用 int k = (int)88.88;

运算符

算数运算符

(+ - * /(商,准确商，小数与除数与被除数中最多小数位的数据类型一致) %(余数))

字符"+"操作

算术表达式中包含多个基本数据类型的时候，整个算术表达式的类型会自动进行提升。等级顺序为byte,short,char->int->long->float->double char i,c;正确相加结果为：int j = i + c

字符串"+"操作

"it " + "is" = "it is" 666 + "it" = "666it" "黑马" + 66 + 4 = "黑马664" 66 + 4 + "黑马" = "70黑马" 字符串中含有数字时，显示和运算从左到右进行，可用括号提升优先级

赋值运算符

"+="等运算符隐含了强制类型转换 "-="、"*="、"/="、"%="

```
int i = 10;
i += 10;//+=的操作包含了强制类型转换
short s = 10;
s = short(s + 10);//需要变换成short类型
```

自增、自减运算符

i++/++i等价于i = i + 1; i-- 参与操作使用： int j = i++; ++放在变量后面，则首先赋值，再执行++，反之则先执行++再赋值

关系运算符

==,!=,>,<,<=,>= int和float可以进行比较

逻辑运算符

与：& 或：| 异或：^ 非!

短路逻辑运算符

&& || 区别：

```
int i = 10;
int j = 20;
(i++ > 100) && (j++ > 100);
//此时因为i++>100为假，当使用短路逻辑与运算符时便不再继续运行，i和j最终值为11，20，及j++不再执行
```

短路逻辑运算遵从从左到右运算，当达到终结条件时右边的关系式不再进一步运算

三元运算符

格式：关系表达式？表达式1：表达式2 a>b?a:b; 计算关系表达式的值，如果为真，则表达式1就是运算结果，否则表达式2为运算结果

案例

三个数最大值

```
int a = 150;
int b = 210;
int c = 165;
int max = (a>b?a:b)>c?(a>b?a:b):c;;
```

数据输入

```
import java.util.Scanner
Scanner sc = new Scanner(System.in);
int x = sc.nextInt();
```

```
int y = sc.nextInt();
System.out.println("x:" + x);
```

流程控制

顺序语句、分支语句、循环语句

if-else语句

```
if(关系表达式){
    语句体1;
}else{
    语句体2;
}
```

if-elseif-else语句

```
if(关系表达式1){
    yuju1;
}else if{
    yuju2;
}else{
    yuju3'
}
```

switch 语句

注意switch中每个case都要求有个break;否则会出现case穿透，导致顺序运行case直到遇到break时停止

```
switch(表达式){
    case 值1:
        yujuti1;
        break;
    case 值2:
        yujti2;
        break;
    case 值3:
        yujuti3;
        break;
    default:
        yujuti4;//因为default在最后，可以不要break
}
//简化版
swicth(表达式){
    case 1:case 2:case 3:
```

```
            yujuti1;
            break;
        case 4:case
    }
```

for循环语句

for循环语句格式

```
for (初始化条件;条件判断句;条件控制句){
    循环体语句;
}
```