数据类型转换：

隐示转换：

自动转换，把所占字节少的数据类型转换成所占字节大的数据类型。安全的转换，

由系统自动执行

显示转换：

强制转换，把所占字节多的数据类型转换成所占字节少的数据类型，是不安全的转换

由开发人员转换

运算符：

关系运算符：

> < == <= >= !=

逻辑运算符：

&&：逻辑与 并且 规则：两侧同时为真，才为真 ，否则为假

短路现象：左为假，右侧不参与运算

| |：逻辑或 或者 规则：两侧同时为假，才为假，否则为真

短路现象：左为真，右侧不参与运算

！ ： 逻辑非 规则：真变假，假变真

条件运算符：

条件表达式？ 表达式1：表达式2

程序的三大结构：

顺序结构：

从main方法开始，从上向下从左向右依次执行。

分支结构：

程序执行到某个位置，进行条件判断。根据不同的结果执行不同的语句

if语句：

```
1. if (条件表达式){
       语句
   }
```

执行顺序：先判断条件表达式的结果，如果为ture，执行语句，false则跳过

2. if(条件表达式){
    语句1
}else{
    语句2
}

执行顺序：先计算条件表达式的结果，如果为ture，执行语句1，false执行语句2

3. if(条件表达式){
    语句1
}else if{
    语句2
}else{
    语句3
}

执行顺序：先计算条件表达式1的结果，如果为ture，执行语句1，false，再计算表达式2的结果为ture，执行语句2，如果false执行语句3

switch语句：

```
switch(整型表达式/字符表达式/字符串表达式){
    case 值1:
        语句;
        break;
    case 值2:
            语句;
        break;
    default:
        语句 ;
```

```
        break;
    }
```

执行顺序：先计算结果，然后拿case的值进行匹配，相同。则执行case中的语句；如果没匹配到，执行defalut 中语句。遇到break时，跳出switch语句。

switch语句效率高于if语句

条件运算符

执行顺序：判断表达式的结果，结果为true 返回表达式1的值 否则，返回表达式2

循环结构：

程序反复执行某些代码，直到条件不满足

for循环：

```
for(循环变量初始化；循环条件；循环变量改变){
    循环体
}
```

执行顺序：第一次执行先执行循环变量初始化，再执行循环体，然后循环变量改变

以后执行：循环条件，循环体，循环变量。当循环条件不满足时循环结束

循环嵌套时：外层循环控制行数，内层循环控制列数。

执行次数=外层循环*内层循环次数

外层循环执行一次，内层循环执行一遍，循环嵌套不能超过三遍

while循环：

```
while(条件表达式){
    语句
}
```

执行顺序：先计算表达式的结果，如果ture，执行语句，false 循环结束

do...while循环：

```
do{
    语句
```

}while(条件表达式);

执行顺序：先执行语句，在计算条件表达式，如果ture 执行语句如果为false 循环结束

循环的选择：

1.知道循环的次数，用for循环

2.知道循环的条件，使用while循环和do...while循环

3.do...while循环至少执行一次循环体

结束循环：

break：跳出本层循环

continue： 跳出本次循环