控制结构

2019年11月4日　　11:36

分类：顺序结构　分支结构　循环结构

一、顺序结构

默认控制结构就是顺序结构

从上到下 从左到右依次执行

最常用的一种结构

二、分支结构

1.if语句

if:

格式一：

if(判断条件){

Code;

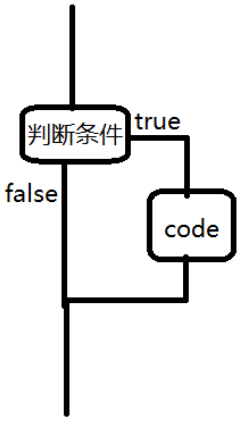
}

判断条件：表达式 boolean值； 结果必须为boolean类型 true false

Code:代码块 一行或者多行代码

执行流程：

先执行判断条件，如果值为true,则执行Code，if语句结束；如果值为false，Code不执行，if语句结束；



案例：

1、获取用户从键盘输入的一个整数，判断该数是否大于0，如果大于0则输出；

格式二：if else

if(判断条件){

Code1;

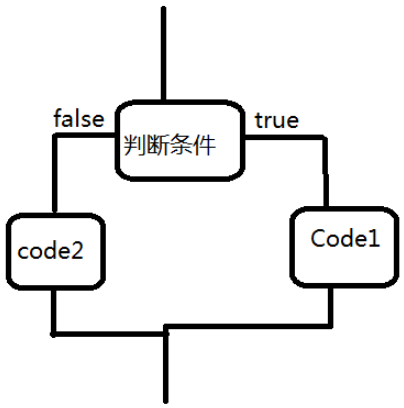
}else{

Code2;

}

执行流程：

先执行判断条件，如果值为true，则执行Code1；如果值为false，则执行Code2；



注意：

else需要紧跟if使用，不能单独存在；

案例：

1、获取用户从键盘输入的数据，并判断该数的奇偶性

2、判断一个数是否是9的倍数，如果是9的倍数 输出“是”如果不是9的倍数 输出“否”

格式三：if else if else

```
if(判断条件1){
    Code1;
}else if(判断条件2){
    Code2;
}else if(判断条件3){
    Code3;
}...
else{
    Code;
}
```

执行流程：

1、先执行判断条件1，

true:执行Code1; if语句结束了

false：执行判断条件2；

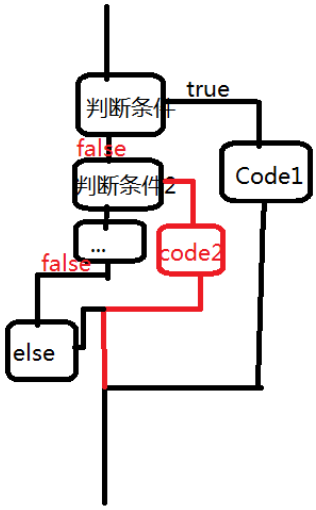
true 执行Code2; if语句结束了

false：执行判断条件3；

true：....

false:....

如果所有的判断条件都为false，则执行else的代码块



案例：

1、获取用户从键盘输入的成绩，输出学生对应的等级

低于60分　不合格

60-70　　合格

70-80　　中等

80-90　　良好

90-100　　优秀

2、判断该数是否大于0的偶数

注意：

1、如果代码块只有一行，可以省略{}

如果没有{}时，只会把紧跟在后边的第一行代码作为代码块

2、判断条件：可以直接给定true或者false，程序编译运行不会有问题，但是没有意义

2.switch..case语句

switch()的值类型：byte/short/char/int，从JDK1.5开始允许使用枚举，从JDK1.7开始允许使用String

格式：

```
switch(值){
    case 值1:
        Code1;
        break;
    case 值2:
        Code2;
        break;
    case ...:
        Code n;
        break;
```

default:
Code;
break;
}

执行流程：

值与case之后的值进行一一匹配，如果和某个case后的值匹配成功，则执行该case相应的代码块；如果和case后的所有值匹配都不成功，则执行defualt的代码块；

case穿透：

当值和case后的某个值匹配成功，先执行该case对应的代码块，之后所有case的代码块都会相应的执行，包括default的代码块也会执行；

注意：

1、default可以放在switch中任意一个位置，但是执行的流程仍然为所有case匹配均不成功的情况下，才会匹配到default;如果default位置不在最后，代码块中也相应的需要添加break； 建议default放在最后

2、break:关键字 结束switch case语句

3、switch支持值的数据类型： int byte short char,String(JDK1.7之后支持)

案例：

1、10086

1-查询话费余额 2-查询套餐内剩余流量 3-查询积分 0-人工服务

2、简易计算器程序的实现 + - * / %

输入两个整型数字 1 2

输入运算符 +

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    //1.获取用户从键盘输入的成绩，输出学生对应的等级
    //低于六十不合格  60-70 合格  70-80 中等  80-90 良好 90-100 优秀
    //格式三  if  else if  else
    Scanner sc=new Scanner(System.in);
    System.out.println("请输入第一个数");
    int num1=sc.nextInt();
    System.out.println("请输入第二个数");
    int num2=sc.nextInt();
    System.out.println("请输入符号");
    String str=sc.next();
    switch(str){
    case "+":
    System.out.println(num1+"+"+num2+"="+(num1+num2));
```

```
            break;
        case "-":
            System.out.println(num1+"-"+num2+"="+(num1-num2));
            break;
        case "*":
            System.out.println(num1+"*"+num2+"="+(num1*num2));
            break;
        case "/":
            System.out.println(num1+"/"+num2+"="+(num1/num2));
            break;
        case "%":
            System.out.println(num1+"%"+num2+"="+(num1%num2));
            break;
        default :
            System.out.println("暂不支持该运算。");
            break;
        }
    }
```

if语句和switch case:

If 更适合范围的判断

switch case适合在有限的值内进行一一匹配

三、循环结构

while do.while for

循环四要素:

初始化条件 循环条件 循环体 改变循环条件的语句

1.while

格式:

初始化条件语句;

while(循环条件){

循环体;

改变循环条件的语句;

}

循环条件:结果为boolean类型 true false

循环体:一行或者多行语句

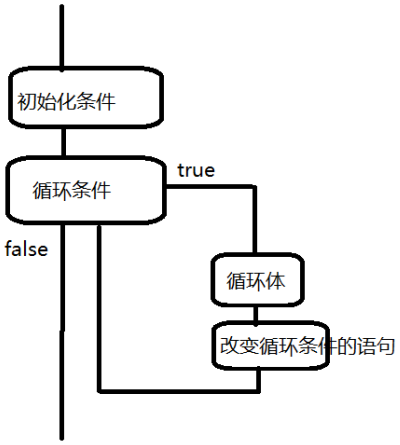
执行流程:

1、初始化条件语句;

2、执行循环条件,

true:循环体；改变循环条件的语句；重复步骤2

false:循环结束，继续执行的后续代码；



注意：

1、循环条件直接为true，则成为死循环，且循环之后的代码无法执行；此情况编译会报错；

2、如果循环条件在运行中恒成立，也为死循环；

3、循环条件直接为false，此情况下循环体永远执行不到，编译会报错；

4、while(num<=200);{} 此时分号表示空语句，该空语句成为while循环体，死循环不断执行，程序卡住；

案例:

1、循环打印10遍Hello World

2、求1-10的和

```
public static void main(String[] args) {
    int sum=0;
    int i=1;
    while(i<=10){
        sum+=i;
        i++;
    }
    System.out.println(sum);
    }
```

3、打印2-200以内的所有偶数

```
public static void main(String[] args) {
    int num=2;
    while(num<=200){
        if(num%2==0){
            System.out.println(num);
        }
        num++;
```

```
    }
}
```

2. do.while循环

格式：

初始化条件语句;

do{

循环体;

改变循环条件的语句;

}while(循环条件);//必须要有分号

执行流程：

1、初始化条件语句;

2、循环体;

3、改变循环条件的语句;

4、循环条件：

true:重复步骤2

false:循环结束

do.while和while区别：

do.while的循环体无论如何都会执行一次；

while循环体有可能一次都不会执行；

3.for

格式：for(初始化条件语句;循环条件;改变循环条件的语句){

循环体；

}

执行流程：

1、初始化条件语句;

2、执行循环条件，

true:循环体；改变循环条件的语句；重复步骤2

false:循环结束，继续执行的后续代码；

案例：

1、for 求1-10和

2、获取用户从键盘输入的整数，求该数每一位上的数字之和

123 1+2+3=6

While循环和for循环的区别：

如果清楚循环的次数，适合使用for循环；

如果不清楚循环的次数，适合使用while循环。

4.循环嵌套

可以将内层循环看作外层循环的循环体；

先内层再外层循环

内层代表列，外层代表行。

案例：

打印99乘法表

四、break和continue

break:

1、switch case语句中，结束switch case语句

2、结束循环语句

continue:

结束当次循环，继续下一次循环

如果循环嵌套，break、continue默认作用的是它所属的循环

java支持给循环编号：

outer:

break 编号;