

第三天:流程控制语句

今日内容简介



拓展补充:

键盘录入(**掌握**)

Scanner 中 next()与 nextLine 区别

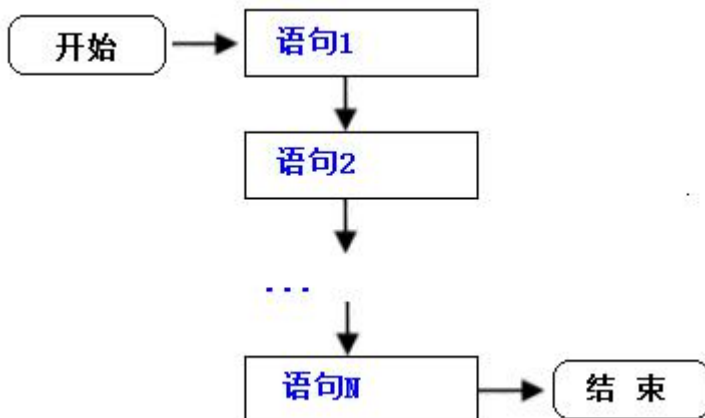
随机函数(**掌握**)

第 1 章 顺序结构

顺序结构:

如果代码里没有流程控制，程序是按照书写的格式从上而下一行一行执行的，

一条语句执行完之后继续执行下一条语句，中间没有判断和跳转，直到程序的结束。



```
1 //顺序结构
2 public class Hello
3 {
4     public static void main(String[] args)
5     {
6         System.out.println("A");
7         System.out.println("B");
8         System.out.println("C");
9         System.out.println("D");
10    }
11 }
```



第 2 章 选择结构

需求:

如果使用程序描述“如果今天是周一，就上班，如果今天是周二就逛街，如果今天是周三就去公园... ..”。显然使用顺序结构是搞不定的。

因为此时程序具有多个条件，需要通过条件判断来决定程序具体做什么，那怎么办呢？

通过判断条件来做选择的语句，我们称为**选择语句或分支语句**。现在，我们一起来学习选择结构，选择结构有两种：分别是 **if** 和 **switch**。

2.1 if 语句

三种格式:

```
1.  if (条件表达式)
    {
        执行语句;
    }
```

```
2.  if (条件表达式)
    {
        执行语句;
    }
    else
    {
        执行语句;
    }
```

```
3.  if (条件表达式)
    {
        执行语句;
    }
    else if (条件表达式)
    {
        执行语句;
    }
    .....
    else
    {
        执行语句;
    }
```



例如:小明跟小强说“如果你追到班上如花同学,我管你叫爸爸”。这句话可以通过下面的一段伪代码来描述。

```
if (小强追到了如花同学) {  
    小明:“ 强爸爸,你好重口味” ;  
}
```

if 语句使用 boolean 表达式或 boolean 值作为选择条件,有三种结构形式:

第一种结构:

if(**boolean** 条件)

{

代码/当 boolean 条件结果为 true 的时候,就执行这里的代码.

}

明天不下雨,我们就去公园.

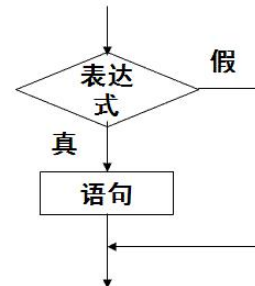
注意:

1):不能写出:if(boolean 条件);{}

2):若 if 的语句只有一句话,其实可以省略{},即是如此,也必须得写{}.

若有多条语句 if 只控制离它最近的语句

3):对于 boolean 类型的表达式,不要使用 **boolean 表达式 == true** 的写法.



2.2 if-else

例如:小明跟小强说“如果你追到班上如花同学,我管你叫**爸爸**,追不到你得管我叫**爷爷**”。

这句话可以通过下面的一段伪代码来描述。

```
if (小强追到班上如花同学) {  
    小明:" 强爸爸你好棒" ;  
}  
else{  
    小强:" 小明爷爷,你赢了." ;  
}
```

思考:通过 if else 语法怎么样来实现判断奇偶数的程序?

注意: if else 的简写格式: **变量 = (条件表达式)?表达式 1:表达式 2;**

其与三元运算符的对比:

三元可以简化代码,但是三元运算符运算完必须要有结果,而 if else 并不需要

if 语句的第二种结构:if-else 语句.

如果.....,否则.....

```
if(boolean 条件)
```

```
{
```

当 boolean 条件结果为 **true** 的时候,就执行这里的代码.

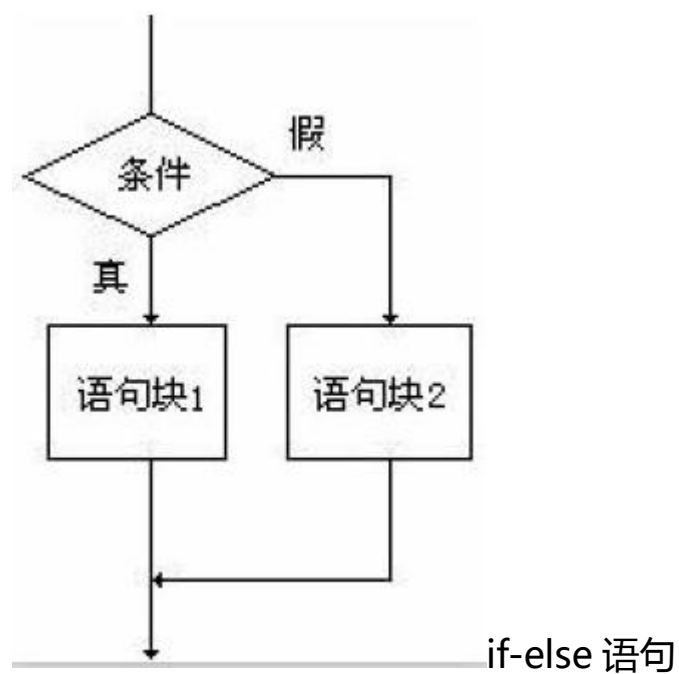
```
}
```

```
else
```

```
{
```

当 boolean 条件结果为 **false** 的时候,就执行这里的代码.

```
}
```



if-else 结构和三元运算符的区别:

- 1):一个是结构,一个运算表达式.
- 2):运算表达式的是必须得有一个结果的.
- 3):从功能上,从语义上是相同的.

不能单独使用 else,必须先有 if,再有 else.

课堂代码:

```
public static void main(String[] args) {  
    //求两个数中的最大的数  
    int x = 120;  
    int y = 50;  
    if(x > y){  
        System.out.println("X是最大");  
    }else{  
        System.out.println("Y是最大");  
    }  
    //分页逻辑  
    //需求:给出一共有46条数据,要求每一页最多10条数据.  
    //计算:一共多少页,  
    int totalCount = 46;  
    int pageSize = 10;  
    int totalPage = totalCount % pageSize == 0 ? totalCount/pageSize : totalCount/pa  
    System.out.println(totalPage);  
  
    if(totalCount % pageSize == 0)  
    {  
        //若为true  
        totalPage = totalCount/pageSize;  
    }else{  
        //若为false  
        totalPage = totalCount/pageSize + 1;  
    }  
  
    System.out.println(totalPage);  
}
```

2.3 if...else if...else 语句

if (小明追到了如花,他们恋爱了一年) {

小强:明爷爷你真牛 B

} else if (恋爱了半年) {

小强:小明哥,你是怎么做到的

}else if (恋爱了一个月)

小强:一开始我还真不相信,你们会坚持一个月

else if (恋爱了一天) {

小强:哈哈,我就知道你嫌她丑.

```
} else {
```

他们连一天都没爱过

```
}
```

需求:考试成绩大于 90 分,就打印优,如果大于 80 分,打印良,如果大于 70 分,打印中,其他情况打印多多努力。对于这个案例,如何解决.

if 语句的第三种结构:**if-else if-else** 语句.

```
if(boolean 条件 1)
```

```
{
```

当条件 1 为 true 的时候,执行这里.

```
}
```

```
else if(boolean 条件 2)
```

```
{
```

当条件 2 为 true 的时候,执行这里

```
}
```

```
else if(boolean 条件 3)
```

```
{
```

当条件 3 为 true 的时候,执行这里

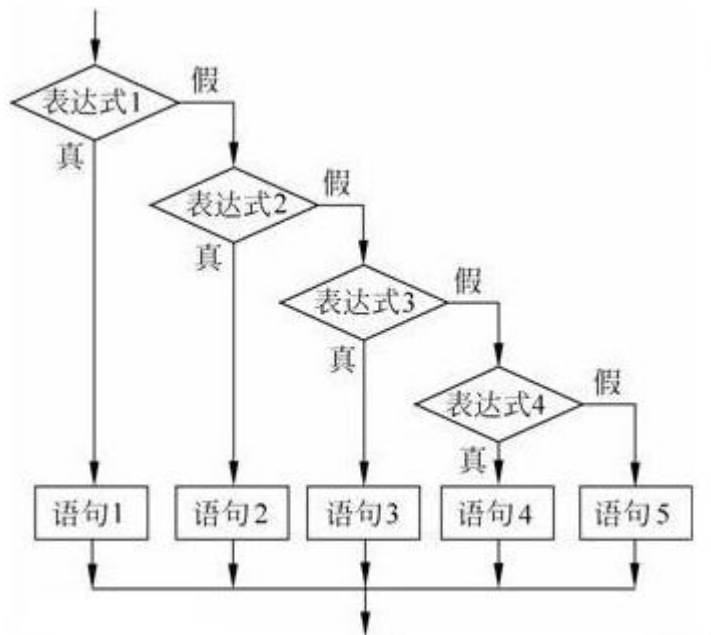
```
}
```

```
else
```

```
{
```

上述所有条件,都不满足,就执行这里.

```
}
```



if-else if-else 语句.

课堂代码 1:

```
int score = 55; //分数
if(score >= 90)
{
    System.out.println("优");
}
else if(score >= 80 && score < 90) // [80,90)
{
    System.out.println("良");
}
else if(score >= 70 && score < 80) // [70,80)
{
    System.out.println("中");
}
else
{
    System.out.println("多多努力");
}
```

课堂代码 2:

```
int today = 15;
if(today == 1)
{
    System.out.println("周1");
}
else if(today == 2)
{
    System.out.println("周2");
}
else if(today == 3)
{
    System.out.println("周3");
}
else if(today == 4)
{
    System.out.println("周4");
}
else if(today == 5)
{
    System.out.println("周5");
}
else if(today == 6)
{
    System.out.println("周6");
}
else if(today == 7)
{
    System.out.println("周末");
}
else
{
    System.out.println("亲,没有这一天");
}
```

2.4 Switch

上述判断 today 变量的例子,从对与错的角度思考,没问题.但是,程序的可读性不好,不太简洁.----->switch 语句(更简单,功能相同)

switch 的语法结构(和 if-elseif-else 的语义相同):

switch(整型表达式)

{

case A 值:

语句 1;

break;//结束

case B 值:

语句 2;

break;

case C 值:

语句 3;

break;

default:

上述所有的结果值都不满足

语句.

}

课堂代码:

```
int today = 13;  
//判断today的不同结果值,输出不同的内容  
switch(today)  
{  
    case 1: System.out.println("周1"); break;  
    case 2: System.out.println("周2"); break;  
    case 3: System.out.println("周3"); break;  
    case 4: System.out.println("周4"); break;  
    case 5: System.out.println("周5"); break;  
    case 6: System.out.println("周6"); break;  
    case 7: System.out.println("周末"); break;  
    default: System.out.println("SB,没这一天");  
}
```

例如：根据考试的名次，给予前 4 名不同的奖品。第一名，奖励笔记本一台；第二名，奖励 IPAD 2 一个；第三名，奖励移动电源一个；最后一名奖励 U 盘一个。

```
int num=1; //保存考试的名次  
switch(num) {  
    case 1:  
        System.out.println("奖励笔记本一台");  
        break;  
    case 2:  
        System.out.println("奖励IPAD 2一个");  
        break;  
    case 3:  
        System.out.println("奖励移动电源一个");  
        break;  
    default:  
        System.out.println("奖励U盘一个");  
}
```

switch 的语法分析以及注意:

1):switch 中的表达式的结果/类型,必须是**整数类型**.

整数类型:byte,short,char,int.没有 long.

从 **Java7** 开始,switch 也支持 String 类型.

2):switch 语句,一旦找到入口,就会进入,并执行功能语句,后面的 case 不再判断,除非语句

break,或 return 或则执行完成,才会停下来-----**穿透**.

3):default 表示,当 case 中的值都不成立的时候,才执行,一般的放在最后.

if 和 switch 都是选择语句/条件语句,那到底什么时候选择使用 if,什么时候选择使用 switch.

分析各自的优缺点:

1): 对于所有的条件语句,if 都可以完成.

2): switch 语句,只能正对于**结果为数值**的情况做判断.

3): 当 if 中条件的结果是数值的时候,使用 switch 更简单.

第 3 章 循环结构

代表语句：while ， do while ， for

while 语句格式：

```
while (条件表达式)
```

```
{
```

```
    执行语句;
```

```
}
```

do while 语句格式：

```
do
```

```
{
```

```
    执行语句;
```

```
}while (条件表达式);
```

故事:话说唐僧师徒四人还在 21 世纪继续闯关，一天，师徒四人来到数字王国：

第一关：Boss 叫傻乎乎的唐僧叫 500 声帅哥。可是唐僧平时吃素，体力不好，叫 200 声就晕了，咋办，悟空马上变了个复读机出来，录音，播放。

第二关：Boss 觉得唐僧好欺负再叫他从 1 数到 100。此时复读机用不上来了，咋办？

第三关：求 100 以内正整数和？

于是，唐僧算 1 到 25 的和，悟空算 26 到 50 的和，八戒算 51 到 75 的和，沙僧算 76 到 100 的和。

第四关：求 1000 以内正整数和？

于是，唐僧算 1 到 250 的和，悟空算 251 到 500 的和，八戒算 501 到 750 的和，沙僧算 751 到 1000 的和。

3.1 While

三大循环之 while 循环:

语法:

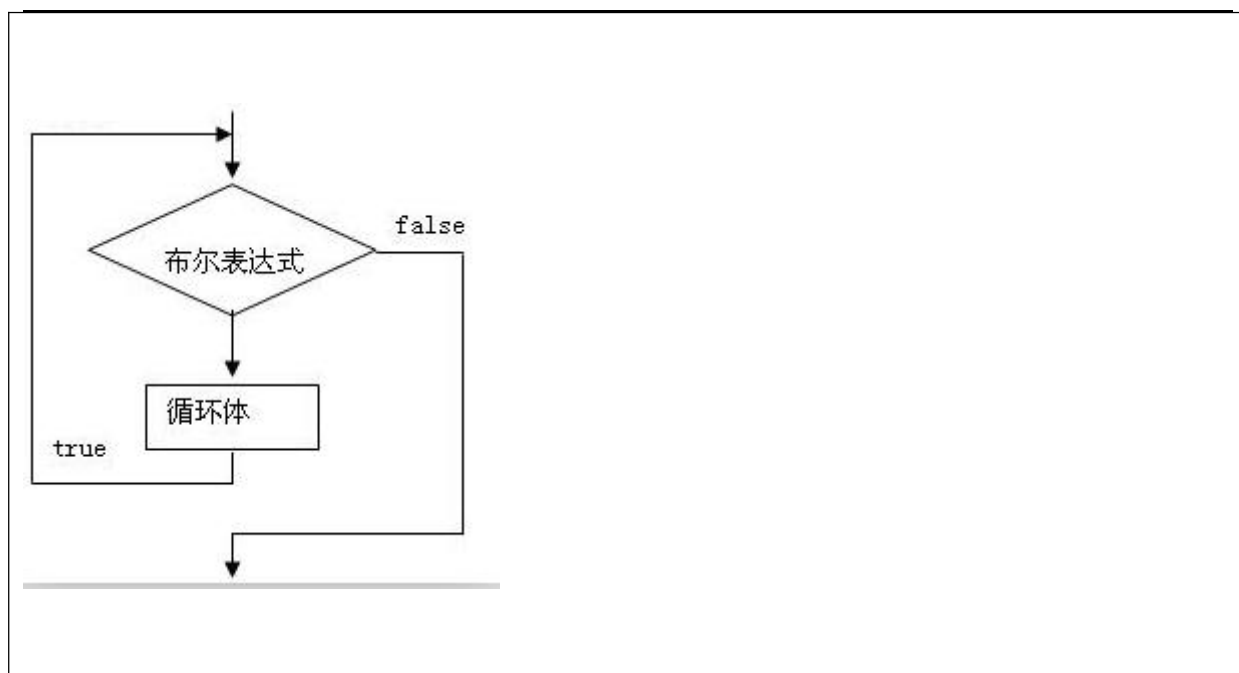
while(boolean 表达式)

{

 若 boolean 表达式结果为 true,则执行这里.循环体

}

注意:先判断 boolean 表达式.



课堂代码:

```
//案例：叫500声帅哥  
int count = 1;//计数器,记录叫了多次帅哥  
  
while(count <=500)  
{  
    //System.out.println("帅哥"+count);  
    count++;  
}
```

```
//案例：从1数到100  
int num = 1;  
while(num <= 100)  
{  
    //System.out.println(num);  
    num++; //每次递增1  
}
```

```
//案例：帮大圣解决问题，计算100以内的正整数和  
//先求出每一个加数,再把每一个加数给叠加起来。  
int x = 1;  
int sum=0;//表示前N个数的和  
while(x <= 100)  
{  
    //sum = sum + x;等价于  
    sum += x;  
    x++;  
}  
System.out.println(sum);
```

3.2 Do-while

三大循环之 do while 循环:

语法:

do

{

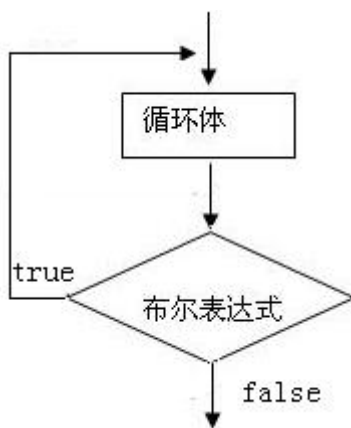
 循环体

}while(boolean 表达式);

注意: do while,不管 boolean 表达式对与错,先执行一次循环体.

 即使 boolean 表达式为 false,do while 也会执行一次.

推论: 无论条件如何,dowhile 至少会执行一次.



课堂代码:

//案例：叫500声帅哥

```
int count = 1;
do
{
    //System.out.println("帅哥"+count);
    count++;
}while(count <= 500);
```

//案例：从1数到100

```
int num = 1;
do
{
    //System.out.println(num);
    num++;
}while(num <= 100);
```

//案例：帮大奎解决问题，计算100以内的正整数和
//先求出每一个加数，再把每一个加数给叠加起来。

```
int x = 1;
int sum = 0; //表示前N个数之和
do
{
    sum += x;
    x++;
}while(x <= 100);
System.out.println(sum);
```

3.3 For

三大循环之 for 循环:

语法:

for(初始化语句;boolean 表达式;循环之后的操作语句)

```
{  
    循环体  
}
```

初始化语句:定义变量,赋初始值;

boolean 表达式:判断条件,若条件为 true,则进入循环体,若为 false,则跳过循环.

循环之后的操作语句: 变量的自增/自减操作.

执行顺序:

1):初始化语句.

2):boolean 表达式:

 true: GOTO 3

 false:跳过循环,不执行循环体

3):执行循环体.

4):循环之后的操作语句, GOTO 2.

注：

A ,for 里面的连个表达式运行的顺序，初始化表达式只读一次，判断循环条件，为真就执行循环体，然后再执行循环后的操作表达式，接着继续判断循环条件，重复找个过程，直到条件不满足为止。

B ,while 与 **for** 可以互换，区别在于 **for** 为了循环而定义的变量在 **for** 循环结束就是在内存中释放。而 **while** 循环使用的变量在循环结束后还可以继续使用。

C, 最简单无限循环格式：while(true), for(;;),无限循环存在的原因是并不知道循环多少次，而是根据某些条件，来控制循环。

执行顺序示例:

```
1
2 5 8 4 7
for (int x = 0; x < 3; x++)
{
    System.out.println("x=" + x); 3 6
}
```

初始化表达式只执行一次,条件不满足循环就结束

```
//案例：叫500声帅哥
for(int i = 1;i<=500;i++)
{
    //System.out.println("帅哥"+i);
}
//案例：从1数到100
for(int i = 1;i<=100;i++)
{
    //System.out.println(i);
}
//案例：帮大圣解决问题，计算100以内的正整
int sum = 0;
for(int i = 1;i<=100;i++)
{
    sum += i;
}
System.out.println(sum);
```

课堂代码

```
/**
```

```
* 测试For循环语句
```

```
*  
*/  
  
public class TestFor {  
  
    public static void main(String[] args) {  
  
        int a = 1;    //初始化  
  
        while(a<=100){ //条件判断  
  
            System.out.println(a);    //循环体  
  
            a+=2; //迭代  
  
        }  
  
        System.out.println("while循环结束！");  
  
  
        for(int i = 1;i<=100;i++){ //初始化//条件判断 //迭代  
  
            System.out.println(i);    //循环体  
  
        }  
  
        System.out.println("for循环结束！");  
  
    }  
}
```


循环综合练习 01 :

```
public class TestWhileFor {  
  
    public static void main(String[] args) {  
  
        //分别求出0到100之间的奇数和偶数的和  
  
        int oddSum = 0; //用来保存奇数的和  
  
        int evenSum = 0; //用来存放偶数的和  
  
        for(int i=0;i<=100;i++){  
  
            if(i%2!=0){  
  
                oddSum += i;  
  
            }else{  
  
                evenSum += i;  
  
            }  
  
        }  
  
        System.out.println("奇数的和："+oddSum);  
  
        System.out.println("偶数的和："+evenSum);  
  
        System.out.println("#####");  
  
        //用while和for循环输出1-1000之间能被5整除的数，并且每行输出3个  
  
        for(int j = 1;j<=1000;j++){
```

```
        if(j%5==0){  
            System.out.print(j+"\t");  
        }  
  
        if(j%(5*3)==0){  
            System.out.println();  
        }  
    }  
  
}
```

3.4 死循环和循环语句对比

一般的,循环都会有一个出口(跳出循环),在开发中有时候咱们就得使用**死循环**.

<pre>while(true){ 循环体 }</pre>	<pre>do { 循环体 }while(true);</pre>	<pre>for(;;) { 循环体 }</pre>
---	---	--

三大循环语句都一样,仅仅只是语法结构上不一样而已.

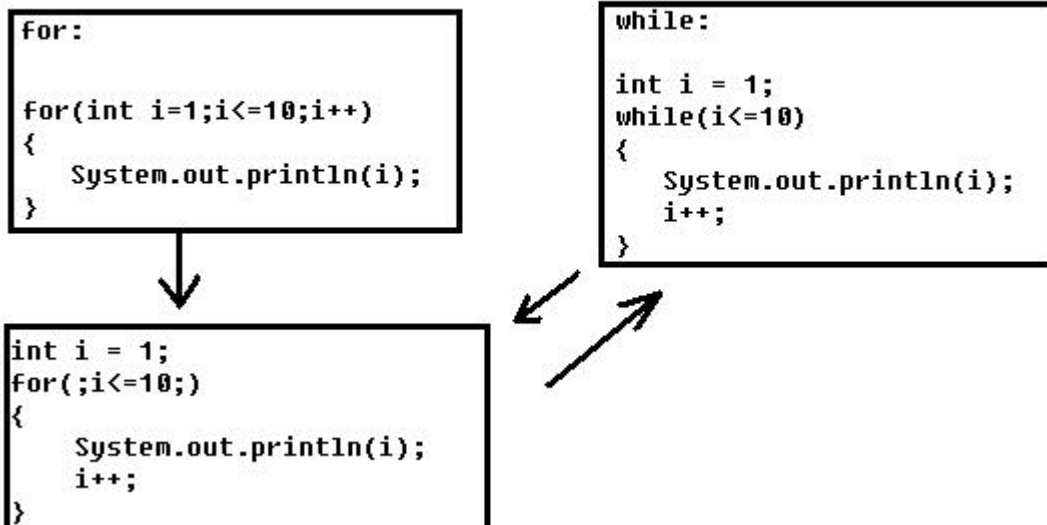
把 while 循环和 for 循环转换.

提示：

三大循环，是可以互换的，一般情况下，要是指定次数的循环，选用 for 循环要方便点。

一般的，习惯上选择 i、j、k 作为循环变量。

从1输出到10. while循环和for循环转换



3.5 嵌套循环

需求:

现在 Boss 要让师徒四人都从 1 数到 100，此时怎么办？

使用已学知识解决：

循环:当一个操作需要重复 N 次.

嵌套循环: 当一个重复的操作,需要执行 N 次.

若外循环,循环次数是 M,内循环的循环次数为 N,则 内循环的循环体需要 $M * N$ 次.

```
for(int count=1; count<=4; count++)  
{  
    for(int i =1;i<=100;i++)  
    {  
        System.out.println(i);  
    }  
}
```

嵌套循环注意:

- 1.使用循环嵌套时，内层循环和外层循环的循环控制变量不能相同。
- 2.循环嵌套结构的书写，最好采用“右缩进”格式，以体现循环层次的关系。
- 3.尽量避免太多和太深的循环嵌套结构。

3.6 嵌套循环性能优化案例

案例描述:请对以下的代码进行优化:

```
for (int i = 0; i < 1000; i++) {  
  
    for (int j = 0; j < 100; j++){  
  
        for (int k = 0; k < 10; k++){
```

循环体代码

```
    }  
  
    }  
  
}
```

从给出的代码可知，不论如何优化，循环体代码执行的次数都是相同的，该部分不存在优化的可能。那么，代码的优化只能从循环变量 i、j、k 的实例化、初始化、比较、自增次数等方面的耗时上进行分析。

```
for (int i = 0; i < 10; i++) {  
  
    for (int j = 0; j < 100; j++){  
  
        for (int k = 0; k < 1000; k++){  
  
            循环体代码  
  
        }  
  
    }  
  
}
```

```
}  
  
-----  
  
int i, j, k;  
  
for (i = 0; i < 10; i++){  
    for (j = 0; j < 100; j++){  
        for (k = 0; k < 1000; k++){  
            循环体代码  
        }  
    }  
}
```

3.7 嵌套循环案例

输出矩形图案:

```
*****
```

```
//第1行
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.println();
//第2行
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.println();
//第3行
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
System.out.print("*");
```

```
//第1行
for(int i=1;i<=5;i++)
{
    System.out.print("*");
}
System.out.println();
//第2行
for(int i=1;i<=5;i++)
{
    System.out.print("*");
}
System.out.println();
//第3行
for(int i=1;i<=5;i++)
{
    System.out.print("*");
}
```

```
//第line行
for(int line=1;line<=3;line++)
{
    for(int i=1;i<=5;i++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

输出三角形图案：

```
*
**
***
****
*****
```

```
System.out.println("*")
System.out.println("**");
System.out.println("***");
System.out.println("****");
System.out.println("*****");
```

第1行：

```
for(int i=1;i <=1 ;i++){
    System.out.print("*");
}
```

```
System.out.println();
```

第2行：

```
for(int i=1;i <=2 ;i++){
    System.out.print("*");
}
```

```
System.out.println();
```

第3行：

```
for(int i=1;i <=3 ;i++){
    System.out.print("*");
}
```

```
System.out.println();
```

.....

第N行

```
for(int line = 1; line<=5 ;line++)
{
```

```
    for(int i=1;i <= line ;i++){
        System.out.print("*");
    }
```

```
}
```

```
当 line = 1: i <= 1;
```

```
当 line = 2: i <= 2;
```

```
当 line = 3: i <= 3;
```

输出九九乘法表：

1x1=1

1x2=2 2x2=4

1x3=3 2x3=6 3x3=9

1x4=4 2x4=8 3x4=12 4x4=16

1x5=5 2x5=10 3x5=15 4x5=20 5x5=25

1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36

1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49

1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64

1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81


```
//第1行
System.out.print("1*1=1");
System.out.println();//空行
//第2行
System.out.print("1*2=2");
System.out.print("2*2=4");
System.out.println();//空行
//第3行
System.out.print("1*3=3");
System.out.print("2*3=6");
System.out.print("3*3=9");
System.out.println();//空行
```



```
//第1行
for(int i=1;i<=1;i++){
    System.out.print(i + "*1=1");
}
System.out.println();//空行
//第2行
for(int i=1;i<=2;i++){
    System.out.print(i+"*2="+i*2);
}
System.out.println();//空行
//第3行
for(int i=1;i<=3;i++){
    System.out.print(i+"*3="+i*3);
}
System.out.println();//空行
```



```
for(int line = 1;line <=9; line ++){
    for(int i=1;i<=line;i++){
        System.out.print(i+"*"+line+"="+i*line +"\t");
    }
    System.out.println();//空行
}
```

第 4 章 控制循环结构语句

break 语句:

结束当前所在的循环.

注意:break 之后的语句,执行不到,若编写了,则语法报错.

案例 1:从 1 数到 10,到数到 5 的时候,就停止.

```
for(int i=1;i<=10;i++)
{
    System.out.println("i="+i);
    if(i == 5)
    {
        break;
    }
}
```

案例 2 : 打印出 100 以内前 5 个 3 的倍数的数字 ;

分析问题: 1):找出哪些是 3 的倍数.

2):定义一个计数器,每次出现 3 的倍数,计数器递增 1 .

3):当计数器递增到 5 的时候,就停止循环.

```
int count = 0;
for(int i=1;i<=100;i++)
{
    if(i % 3 == 0)
    {
        System.out.println(i);
        count++;
    }
    if(count == 5)
    {
        break;
    }
}
```


continue 语句:

表示跳过的这一次当前循环,继而进行下一次循环.

注意:contine 之后的语句,执行不到,若编写了,则语法报错.

案例 1:从 1 数到 10,当数到 4 的时候,跳过.

```
//案例1:从1数到10,当数到4的时候,跳过.
for(int i=1;i<=10;i++)
{
    if(i == 4)
    {
        continue; //跳过当前这一次循环
    }
    System.out.println(i);
}
```



案例 2:输出 100 到 200 之间不能被 3 整除的数。

```
for(int i=100;i<=200;i++)
{
    if(i % 3 == 0)
    {
        continue;
    }
    System.out.println(i);
}
```

return 语句:

表示结束循环所在的方法,方法都结束了,循环结构自然也就结束了。

注意:

break、continue、return 后面的语句永远没有机会执行,所以不能再跟任何语句,否则编译失败。

```
public static void main(String[] args)
{
    for(int i=1;i<=10;i++)
    {
        System.out.println("i="+i);
        if(i == 5)
        {
            break;
        }
    }
    System.out.println("...ending...");//能执行到
    for(int i=1;i<=10;i++)
    {
        System.out.println("i="+i);
        if(i == 5)
        {
            return;
        }
    }
    System.out.println("...ending...");//不能执行
}
```

控制外层循环:

此时就得使用标签了，标签就是给某个循环起的别名，不过该别名得满足标识符的规范。

若要控制外层循环,就在 break 或 continue 后面跟上循环的别名就 OK 了

如: break <标签名>;

```
public static void main(String[] args)
{
    outer: for(int i=1;i<=5;i++)
    {
        for(int j=1;j<=4;j++)
        {
            //当i等于3的时候,终止循环
            if(i == 3)
            {
                break outer;
            }
            System.out.println("i="+i+",j="+j);
        }
    }
}
```

