# “循环”也疯狂--深入理解循环和数组

传一科技 高洪涛 2018-04-03

学习Java都从基础知识（变量、数组、循环）开始，一般人员普遍感觉是易学难精，本文将会介绍关于数组、循环的各种深入用法，通过这些高级写法锻炼编程思维，达到精通及随心所欲地步。

本文包含以下部分：

1简单的一重循环

2简单的二重循环

3简单的累加计算

4简单的三重循环计算

5 打印HELLO

6 递归问题

## 1 简单的一重循环

1.1 循环打印1 2 3 4 5 6 7 8 9

这道题目但凡学过循环的都会做，很简单的一重循环就搞定了,只是个开胃小菜。

for(int i=0; i<=9;i++){

System.out.print(i+” ”);

}

1.2 循环打印 1 2 3

4 5 6

7 8 9

这道题是在上一道题的改进，观察数字就发现规律是逢369换行，可以采用i%3判断是否是3的倍数。

for(int i=0; i<=9;i++){

System.out.print(i+” ”);

if(i%3==0)

System.out.println();

}

## 2 简单的二重循环

2.1 打印九九乘法表

打印乘法表是个老掉牙的题目了，能很好的表现二重循环的控制能力，一般写法如下：

for(int i=1;i<=9;i++) {

for(int j=1;j<=i; j++){

System.out.print(j+"\*"+i+"="+j\*i+"\t");

}

System.out.println("");

}

2.2 使用一重循环打印乘法表

现在要求更进一步，只能使用一重循环完成乘法表打印，这就是高级货了，常常见于笔试题中。如何把行列的变化用一重循环实现呢？

for(int i=1, j=1; i<9; j++){ //i控制行，j控制列

System.out.print(i+ “\*” +j+ “=”+i\*j+ “ ”);

if(j==i){

j=0;

i++;

System.out.println();

}

}

上述代码的核心思想是用j驱动i的递增，每当j==i的时候就i++，然后j复位为0。是不是很巧妙呢？

## 3 简单的累加计算

3.1 计算计算1+2+3+4+…+100

这也是老掉牙的题目了，学习循环的入门题目，一口气就写完了。

int sum=0;

for(int i=1; i<=100; i++){

sum += i;

}

System.out.println(“sum=”+sum);

* 1. 计算1+1/1+1/2+1/3+1/4+…+1/100

这道题目的真实含义不是考察循环累加，而是考察是否认识并正确的使用类型强制转换。

一般人的写法是

sum += (double)(1/i);

更巧妙的写法是

sum += 1.0/i;

怎么样，感觉精简很多吧？使用到了不同类型数据混合运算自动提升类型的特点。

## 4 简单的三重循环计算

4.1题目： x+2y+5z=100， 求有多少组整数解

这个题目很简单，但凡学过三重循环的人员都能随手写出程序：

int count=0;

for（int x=0; x<100;x++）

for（int y=0; y<100;y++）

for（int z=0; z<100;z++）{

if(x+2\*y+5\*z==100){

count++;

}

}

要注意的这个三重循环写法中有很多无效循环次数，总循环次数是100\*100\*100=100万次。

4.2 要求使用二重循环求解上述问题

观察公式x+2y+5z=100，思考后就能发现x可有可无，只要2y+5z<=100，就总能找到一个x使得x+2y+5z=100成立。这样可以省略x循环，写的代码如下：

int count=0;

for（int y=0; y<=50;y++）

for（int z=0; z<=20;z++）{

if(2\*y+5\*z<=100){

count++;

}

}

这次循环次数=50\*20 = 1000次 大大减少了。

4.3 继续提高要求，使用一重循环解法

这个要求可就厉害了，出自华为笔试题，一般人能写出二重循环就算学得好的了，写的出一重循环的简直是凤毛麟角。简洁的分析过程如下:

先省略x，观察2y+5z<=100，变形成 y<=（100-5z）/2， 依次取z=0/1/2/3…计算y的值。

Z=0， y<=50, 表示当z=0时， y的最大值是50，取值是0/1/2/3/4…50, 总计51个值，也就代表z=0时有51组解。

Z=1， y<=47, 表示当z=1时， y的最大值是47，取值是0/1/2/3/4…47, 总计48个值，也就代表z=0时有48组解。

Z=2， y<=45, 表示当z=2时， y的最大值是45，取值是0/1/2/3/4…45, 总计46个值，也就代表z=0时有46组解。

Z=3， y<=42, 表示当z=2时， y的最大值是45，取值是0/1/2/3/4…42, 总计43个值，也就代表z=0时有46组解。

依次类推。。。

Z=18， y<=5, 表示当z=18时， y的最大值是5，取值是0/1/2/3/4/5 总计6个值，也就代表z=0时有6组解。

Z=19， y<=2, 表示当z=19时， y的最大值是2，取值是0/1/2, 总计3个值，也就代表z=0时有3组解。

Z=20， y<=0, 也就代表z=20时有1组解。

现在只要对z进行循环，程序也就呼之欲出了：

int count=0;

for（int z=0; z<=20;z++）{

count += (100-5\*z)/2+ 1;

}

现在循环计算的次数是 21次，算法改进后大大简化了循环次数，这就是算法的威力。

4.4 终极问题：还能不能继续减少循环次数呢？

很多人看到这肯定在说不可能了，再也简化不了了。事实的真相就在大家的观察角度。

仔细看看上述循环中的序列：

51 48 46 43 41 38 36 33 31 28 26 23 21 18 16 13 11 8 6 3 1

每两个数字叠加的和是99 89 79 69 59 49 39 29 19 9 1 ，这么有规律的数字一个循环搞定：

int count=1;

for（int i=9; i<=99; i+=10）{

count += i;

}

现在只用了10次循环就搞定了，从100万次到10次，算法的威力真是让人咂舌啊。

## 5 打印HELLO

题目： 使用数组打印输出下列图形“HELLO”



5.1 用一维数组打印

这很简单，把每一行当做一个String字符串，总共5行就是一个5个元素的String[], 一个循环就打印输出了，代码省略。

5.2 用二维数组打印

观察图案，可以把每个字母当做一个二维字符数组，分解为H、E、L、O,这样用二维循环打印输出。

public class StringPrint {

public static char[][] strH = {

{ '\*', ' ', ' ', '\*' },

{ '\*', ' ', ' ', '\*' },

{ '\*', '\*', '\*', '\*' },

{ '\*', ' ', ' ', '\*' },

{ '\*', ' ', ' ', '\*' } };

public static char [][] strE = {

{ '\*', '\*', '\*', '\*' },

{ '\*', ' ', ' ', ' ' },

{ '\*', '\*', '\*', '\*' },

{ '\*', ' ', ' ', ' ' },

{ '\*', '\*', '\*', '\*' } };

public static char [][] strL = {

{ '\*', ' ', ' ', ' ' },

{ '\*', ' ', ' ', ' ' },

{ '\*', ' ', ' ', ' ' },

{ '\*', ' ', ' ', ' ' },

{ '\*', '\*', '\*', '\*' } };

public static char [][] strO = {

{ ' ', '\*', '\*', ' ' },

{ '\*', ' ', ' ', '\*' },

{ '\*', ' ', ' ', '\*' },

{ '\*', ' ', ' ', '\*' },

{ ' ', '\*', '\*', ' ' } };

public static void main(String[] args) {

// TODO Auto-generated method stub

printString("HELLO");

}

public static void printString(String str) {

// 分解字符串的每一个字母，分别打印

char[] array = str.toCharArray();

for (char ch : array) {

//System.out.print(ch);

printCharacter(ch);

}

}

public static void printCharacter(char ch) {

char[][] charray = null;

switch (ch) {

case 'H':

charray = strH;

break;

case 'E':

charray = strE;

break;

case 'L':

charray = strL;

break;

case 'O':

charray = strO;

break;

default:

break;

}

for(int i=0;i<charray.length;i++){

for(int j=0;j<charray[i].length;j++){

System.out.print(charray[i][j]);

}

System.out.println();

}

}

}

5.3 用三维数组打印

上述的二维数组有多个，可以把这几个二维数组集中起来单独定义一个三维数组：

char[][][] str\_hello = {

strH,

strE,

strL,

strL,

strO

};

然后直接打印输出：

for(int i=0; i<str\_hello[0].length;i++){ //先按行循环

for(int j=0; j<str\_hello.length;j++){ //后按字母循环

for(int k=0; k<str\_hello[j][i].length;k++){

System.out.print(str\_hello[j][i][k]);

}

System.out.print(" ");

}

System.out.println();

}

5.4 能否用任意的字符打印”HELLO”呢？例如：



要实现这样的任意字符打印输出，就得要改造数组和打印函数，一个思想是模仿字库和字模，把每个字母的二维数组定义成1001之类的数据，当为1时输出指定的打印字符，当为0时输出空格。

public class StringPrint {

public static int[][] strH = {

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 } };

public static int[][] strE = {

{ 1, 1, 1, 1 },

{ 1, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 0 },

{ 1, 1, 1, 1 } };

public static int[][] strL = {

{ 1, 0, 0, 0 },

{ 1, 0, 0, 0 },

{ 1, 0, 0, 0 },

{ 1, 0, 0, 0 },

{ 1, 1, 1, 1 } };

public static int[][] strO = {

{ 0, 1, 1, 0 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 0, 1, 1, 0 } };

public static void main(String[] args) {

// TODO Auto-generated method stub

printString("HELLO");

}

public static void printString(String str) {

// 分解字符串的每一个字母，分别打印

char[] array = str.toCharArray();

for (char ch : array) {

//System.out.print(ch);

printCharacter(ch);

}

}

public static void printCharacter(char ch, char print) {

int[][] charray = null;

switch (ch) {

case 'H':

charray = strH;

break;

case 'E':

charray = strE;

break;

case 'L':

charray = strL;

break;

case 'O':

charray = strO;

break;

default:

break;

}

for(int i=0;i<charray.length;i++){

for(int j=0;j<charray[i].length;j++){

if(charray[i][j]==0){

System.out.print(" ");

}else{

System.out.print(print);

}

}

System.out.println();

}

}



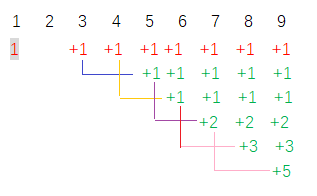
}

## 6 递归问题

题目： 数兔子

兔子在出生两个月后，就有繁殖能力，一对兔子每个月能生出一对小兔子来。如果所有兔都不死，那么一年以后可以繁殖多少对兔子？

分析： 写出每个月的新增兔子对数，注意新出生的兔子2个月后就会生小兔子，老兔子每个月都会生小兔子。



推理计算就得到每个月的兔子数量是： 1 1 2 3 5 8 13 21 34 ，观察到的规律是 前两个数相加等于第三个数，这个数列就是大名鼎鼎的菲波那切数列。用数学公式表达：

F(n)=F(n-1)+F(n-2)

如果对 F(n-1)/ F(n) 求极限，可得到0.618， 这就是黄金分割数。

现在如何编写程序计算出一年后第12个月的兔子数呢？

6.1 用循环实现计算，注意迭代赋值

int f1=1;

int f2=1;

int f3=0;

for(int i=3;i<=12;i++){

f3 = f2+f1;

f1 = f2;

f2 = f3;

}

System.out.println("f3="+f3);

6.2 用递归实现计算， 简洁明了。

public static int f(int n){

if(n==1 || n==2){

return 1;

}

return f(n-1)+f(n-2);

}

------------------ 全文完 --------------------