> 信息学院 计科 22920212204066 邓语苏
>
> 2023.4.12

庆祝厦门大学校庆

题目要求：

在厦门大学建校 102 周年庆典期间，信息学院学子用工科生的浪漫借代码向厦大告白——用 C 语言绘出以下图形：

用 Java 也绘制以上图形：

- 使用数学方法指定打印规则
- 不能使用逐行打印原始文本

核心代码如下：

```
package com.test1;

public class heart {
    public static void main(String[] args) {
        System.out.println("厦门大学102岁生日快乐！");

        float x, y;
        int flag=0;
        for (y = 1.3f; y > -1.1f; y -= 0.15f) {
```

```
            if(y>-0.9f&&y<0.9f) {
                System.out.print("    *********                  ");
            }else{
                System.out.print("*****************             ");
            }
            for (x = -1.2f; x <= 1.2f; x += 0.05f) {//爱心部分

                float temp = x * x + y * y - 1;
                //用心形函数实现爱心
                if ((Math.pow(temp, 3) - (x * x * Math.pow(y, 3))) <= 0.0f) {
                    if(Math.pow(temp, 3) - (x * x * Math.pow(y, 3)) <= 0.0f)
                    System.out.print("*");

                }else{
                    System.out.print(" ");
                }
            }


            if(y>-0.7f) {//U部分
                System.out.print("          ****           ****");
            }else{
                System.out.print("          ******************");
            }
            //换行
            System.out.println();
        }
    }
}
```

算法思想：

- 一层层打印，从上到下，从左到右
- 爱心部分用数学中的心形函数来实现
- 用Math.pow函数来简化幂运算的书写

实现效果：

```
E:\Java\jdk-17\bin\java.exe "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\li
厦门大学102岁生日快乐!
*****************                                                  ****         ****
*****************                  *************      *************            ****         ****
*****************              ****************************************         ****         ****
   *********                 ********************************************       ****         ****
   *********               ************************************************     ****         ****
   *********               ************************************************     ****         ****
   *********               ************************************************     ****         ****
   *********                ***********************************************     ****         ****
   *********                 *********************************************      ****         ****
   *********                  ********************************************      ****         ****
   *********                    ******************************************      ****         ****
   *********                     ****************************************       ****         ****
   *********                      **************************************        ****         ****
   *********                        ************************                    ****         ****
   *********                           ****************                         ****         ****
   *********                               *********                            ******************
*****************                              *                                ******************
*****************                                                               ******************
```

找出所有水仙花数

题目要求：

水仙花数是一个三位数，其各位上的数字的立方之和等于这个三位数。
例如：$371 = 3^3 + 7^3 + 1^3$,因此371是一个水仙花数。

- 请找出并打印所有的水仙花数
- 并统计水仙花数个数。
- 尽可能尝试多种解法（算法），并分析每种解法的性能。

核心代码：

```
package com.test1;

public class flowers {
    public static void main(String[] args) {
        flower1();
        flower2();
    }
    public static void flower1(){//方法一
        int n,i,j,k;
        int sum=0;
        for(n=100;n<1000;n++){
            i=n/100;//最高位
            j=(n/10)%10;
            k=(n%10)%10;
            if(n==Math.pow(i,3)+Math.pow(j,3)+Math.pow(k,3)){//用pow函数
                System.out.println(n);
                sum++;
            }
        }
        System.out.println("100~1000内，水仙花数有："+ sum +"个");
    }
    public static void flower2(){//方法2
        int sum=0;
        for(int i=1;i<10;i++){
            for(int j=0;j<10;j++){
                for(int k=0;k<10;k++){
                    int n=i*100+j*10+k;
                    if(n==Math.pow(i,3)+Math.pow(j,3)+Math.pow(k,3)){
                        System.out.println(n);
                        sum++;
                    }
                }
            }
        }
        System.out.println("100~1000内，水仙花数有："+ sum +"个");
    }

}
```

算法思想：

- 方法一：暴力求解，遍历100~999，得到满足条件的数
- 方法二：用三重循环嵌套求解

实现效果：

```
153
370
371
407
100~1000内，水仙花数有：4个
153
370
371
407
100~1000内，水仙花数有：4个
```

设计一个简化的取棍游戏

游戏规则：

1. 一定初始数量的木棍摆在桌上
2. 两名玩家交替取棍
3. 玩家一次可取1,2或3根木棍

取最后一根木棍的输
实验要求：

- 实验初始时需要向人类玩家询问并用Scanner设定：
  - 初始木棍数量，并判断输入木棍数量数据合法性：大于等于5根，小于等于30根
- 程序中电脑和人类玩家交替取棍，直到木棍个数为0
  - 双方取棍数量为1,2或3，需要对人类玩家的输入数据进行合法性检查
- 人类玩家取棍时需要再次从键盘上输入取棍个数
- 根据哪位玩家取走最后一根木棍，宣布最后赢家
- 用System.out中的打印方法进行游戏过程的展示

核心代码：

```
package com.test1;
import java.util.Random;
import java.util.Scanner;
public class game01 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();//木棍总数
        Random random=new Random();

        while(n<5||n>30) {
            System.out.println("输入无效");
            n = sc.nextInt();
        }
            while (n > 0) {
                int x=random.nextInt(2)+1;
```

```
                n -= x;
                System.out.println("电脑取走了"+ x +"个木棍"+"，现在剩余"+n+"个木棍");
                int y = sc.nextInt();
                while (y > 3 || y < 1) {
                    System.out.println("输入无效");
                    y = sc.nextInt();
                }
                n-=y;
                System.out.println("人类取走了"+ y +"个木棍"+"，现在剩余"+n+"个木棍");
                if(n==0)
                    System.out.println("人类玩家获胜");
                if(n<0)
                    System.out.println("电脑获胜");

            }
    }
}
```

算法思想：

- 用while循环判断取棍是否有效，不断重新输入，直至有效后退出循环
  - 是否$\in [1,3]$
  - 所取木棍是否小于当前木棍数
- 用java自带的random函数生成1~3的随机数
- 若电脑取完，木棍为0，电脑获胜，退出循环；否则，若人类取完，木棍为0，人类获胜，退出循环

发现的规律：

设先手、后手分别为A、B；

若先手取完，剩4个木棍给B时，A必胜。

同理往前推，当A取完，剩8个木棍给B时，A必胜

即先手只要抢占到，取完剩4k的位置，先手必胜

实现效果：

```
10
电脑取走了1个木棍，现在剩余9个木棍
5
木棍不足！ or 输入无效
3
人类取走了3个木棍，现在剩余6个木棍
电脑取走了1个木棍，现在剩余5个木棍
3
人类取走了3个木棍，现在剩余2个木棍
木棍不足！
电脑取走了1个木棍，现在剩余1个木棍
1
人类取走了1个木棍，现在剩余0个木棍
人类获胜
```

创建一个多行的数字阵列

实验内容：
程序需要响应用户输入的整数，代表行的长度n,创建长度为的int数组，数组中元素被随机的赋值，同时打印出该数组。在m次输入后，将前面所有创建行都存储到一个二维数组中，并打印出这个二维数组。
实验要求：

1. 请设计以下函数来辅助完成这个程序：创建随机数组，打印数组。
2. 用户每次输入的行的长度不一定相等。
3. m的值是确定的，可以预先赋值，也可以通过询问用户获得

核心代码：

```
package com.test1;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

public class array {
     public static void main(String[] args){
        new_Array();
    }
    public static void new_Array(){
        Scanner sc=new Scanner(System.in);
        Random random=new Random();
        int n=sc.nextInt();
        int [] arr=new int[n];

        ArrayList al=new ArrayList();

        for(int i=0;i<n;i++) {
            int m = sc.nextInt();
            if(i>=1)
                arr[i] = m+arr[i-1];
            else
                arr[i]=m;
            while (m > 0) {
                int x = random.nextInt(10);
                al.add(x);
                m--;
            }
        }
        int row=0;
        for(int i=0;i<al.size();i++){
            if(i==arr[row]){
                System.out.println("\n");
                row++;
            }
            System.out.print(al.get(i)+" ");

        }
    }
}
```

算法思想:

- 用java中的动态数组ArrayList来存储输入的数组
- 用数组Arr来存储每行的维数
- m的值通过询问用户获取
- 用randow来创建随机数组

实现效果:

```
3
6
3
4
9 1 2 0 1 2
7 7 6
7 2 8 1
```

滚动阵列

实验内容:
请设计一个交互式的程序:
程序可以响应用户输入的方向键a,s,d,w(左，下，右，上)，对随机生成的数字阵列按照输入的方向进行滚动，并且打印出移动后的数字阵列。

实验要求:

1. 随机生成的数组阵列中的总行数和每行的长度都应是随机的
2. 调用实验4中的函数来完成随机生成数字阵列和打印数组的功能。
3. 响应用户输入一次，滚动距离应为一个元素的距离，数组末尾的元素会滚动到开头。

核心代码:

```
package com.test1;

import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

public class run {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        Random random=new Random();
        int n=sc.nextInt();
        int [] arr=new int[n];
        ArrayList al=new ArrayList();
        int max=-1;
        for(int i=0;i<n;i++) {
            int m = sc.nextInt();
            if(m>max)
                max=m;
                arr[i]=m;
            while (m > 0) {
                int x = random.nextInt(10);
                al.add(x);
                m--;
```

```
            }
        }


//        int sum=0;
//        for(int i=0,j=0;i<al.size();i++){
//            System.out.print(al.get(i)+" ");
//            sum++;
//            if(sum==arr[j]){
//                System.out.print("\n");
//                j++;
//                sum=0;
//            }
//        }
        int [][] Arr=new int[n][max];
        int row=0;
        for(int i=0,j=0;i<al.size();i++){
            if(j==arr[row]){
                row++;
                j=0;
                Arr[row][j++]=(int)al.get(i);
            }
            else Arr[row][j++]=(int)al.get(i);
        }

        for(int i=0;i<n;i++){
            for(int j=0;j<max;j++){
                if(Arr[i][j]!=0)
                System.out.print(" "+Arr[i][j]);
            }
            System.out.println("\n");
        }
        //移动操作。
        char order;
        order=sc.next().charAt(0);
        while(order!='0'){//输入0时退出

            switch (order){
                case 'a':{
                    for(int i=0;i<n;i++){
                        int temp=Arr[i][0];
                        for(int j=0;j+1<max;j++){
                            Arr[i][j]=Arr[i][j+1];
                        }
                        Arr[i][max-1]=temp;
                    }
                    for(int i=0;i<n;i++){
                        for(int j=0;j<max;j++){
                            if(Arr[i][j]!=0)
                            System.out.print(" "+Arr[i][j]);
                        }
                        System.out.println("\n");
                    }

                    break;
```

```
                }
                case 'd':{
                    for(int i=0;i<n;i++){
                        int temp=Arr[i][max-1];
                        for(int j=max-1;j>0;j--){
                            Arr[i][j]=Arr[i][j-1];
                        }
                        Arr[i][0]=temp;
                    }
                    for(int i=0;i<n;i++){
                        for(int j=0;j<max;j++){
                            if(Arr[i][j]!=0)
                            System.out.print(" "+Arr[i][j]);
                        }
                        System.out.println("\n");
                    }
                    break;
                }
                case 'w':{
                    for(int j=0;j<max;j++){
                        int temp=Arr[0][j];
                        for(int i=0;i+1<n;i++){
                            Arr[i][j]=Arr[i+1][j];
                        }
                        Arr[n-1][j]=temp;
                    }
                    for(int i=0;i<n;i++){
                        for(int j=0;j<max;j++){
                            if(Arr[i][j]!=0)
                            System.out.print(" "+Arr[i][j]);
                        }
                        System.out.println("\n");
                    }
                    break;
                }
                case 's':{
                    for(int j=0;j<max;j++){
                        int temp=Arr[n-1][j];
                        for(int i=n-1;i>0;i--){
                            Arr[i][j]=Arr[i-1][j];
                        }
                        Arr[0][j]=temp;
                    }
                    for(int i=0;i<n;i++){
                        for(int j=0;j<max;j++){
                            if(Arr[i][j]!=0)
                            System.out.print(" "+Arr[i][j]);
                        }
                        System.out.println("\n");
                    }
                    break;
                }
                default:{
                    System.out.println("请输入(a,s,w,d)来滚动阵列：");
                    break;
```

```
                }
            }
            order=sc.next().charAt(0);
        }

    }
}
```

算法思想：

- 先用java中的动态数组ArrayList来完成不定长数组的输入
- 通过ArrayList来创建数组
- 用while循环+switch选择来完成不同指令的操作

实现效果：

```
3
3
4
6
 6 2 6

 7 8 1 2

 3 7 9 1 3

a
 2 6 6

 8 1 2 7

 3 7 9 1 3

w
 8 1 2 7

 3 7 9 1 3

 2 6 6

s
 2 6 6

 8 1 2 7

 3 7 9 1 3

d
 6 2 6

 7 8 1 2

 3 7 9 1 3
```

```
j
请输入(a,s,w,d)来滚动阵列:
0

Process finished with exit code 0
```