关于全排列 (C++)

code:

```
#include<iostream>
#include<algorithm>
using namespace std;
    /**
    测试c++ next_permutation(arr,arr.size()) 函数:
    a. 函数模板: next_permutation(arr, arr+size);
    b. 参数说明:
            arr:  数组名;
            size: 数组元素个数;
    c. 函数功能:  返回值为bool类型, 当当前序列不存在下一个排列时, 函数返回false, 否则返回true, 排列好的数在数组中存储;
    d. 注意: 在使用前需要对欲排列数组按升序排序, 否则只能找出该序列之后的全排列数。比如, 如果数组num初始化为2,3,1, 那么输出就变为了: {2 3 1} {3 1 2} {3 2 1}.
    e. 头文件: #include<algorithm>
    **/

    //对应的有: prev_permutation(arr,arr.size())函数。
int main(){
    int arr[] = {3, 2, 1};
    cout<<"prev_permutation() 对arr进行全排列: "<<endl;
    do{
        cout << arr[0] << ' ' << arr[1] << ' ' << arr[2] <<'\n';
    }while(prev_permutation(arr, arr+3));

    int arr1[] = {1, 2, 3};
    cout<<"next_permutation() 对arr1进行全排列: "<<endl;
    do{
        cout << arr1[0] << ' ' << arr1[1] << ' ' << arr1[2] <<'\n';
    }while(next_permutation(arr1, arr1+3));
    return 0;
}
```

输出:

```
prev_permutation() 对arr进行全排列:
3 2 1
3 1 2
2 3 1
2 1 3
1 3 2
1 2 3
next_permutation() 对arr1进行全排列:
1 2 3
1 3 2
2 1 3

2 3 1
```

```
3 1 2
3 2 1
```

对C(5,12)的组合

```java
package test16;

//排列组合
public class q7 {
    public static void main(String[] args) {
        int count = 0;
        int a[] = new int[5];
        for (a[0] = 0; a[0] < 12; a[0]++) {
            for (a[1] = a[0] + 1; a[1] < 12; a[1]++) {
                for (a[2] = a[1] +1 ; a[2] < 12; a[2]++) {
                    for (a[3] = a[2]+1; a[3] < 12; a[3]++) {
                        for (a[4] = a[3]+1; a[4] < 12; a[4]++) {
                            System.out.print(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+"
"+a[4]+"\n");
                            count++;
                        }
                    }
                }
            }
        }
        System.out.print(count);
    }
}
```

Calender

```java
package lanqiao;

import java.util.Calendar;
//世纪末的星期天
public class test13_q1 {
    /**
     * TODO: 那一年就是那一年
     * TODO: 从0开始, 即: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
     * TODO: 周日在前, 即[周日, 周一, 周二, 周三, 周四, 周五, 周六]
     *
     * System.out.print(calendar.getTime() + " | ");
     * System.out.print(calendar.get(Calendar.YEAR) + "-"); // 那一年就是那一年
     * System.out.print(calendar.get(Calendar.MONTH) + "-"); // 从0开始, 即: [0, 1, 2, 3,
4, 5, 6, 7, 8, 9, 10, 11]
     * System.out.print(calendar.get(Calendar.DAY_OF_MONTH) + " ");
     * System.out.print(" week"+calendar.get(Calendar.DAY_OF_WEEK)+" "); // 周日在前, 即
[周日, 周一, 周二, 周三, 周四, 周五, 周六]
     * System.out.print(calendar.get(Calendar.HOUR_OF_DAY) + ":");
     * System.out.print(calendar.get(Calendar.MINUTE) + ":");

     * System.out.println(calendar.get(Calendar.SECOND));
     **/
    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.YEAR, 1999);
        calendar.set(Calendar.MONTH, 11);
        calendar.set(Calendar.DAY_OF_MONTH, 31);
        System.out.println(calendar.getTime());
        while (calendar.get(Calendar.DAY_OF_WEEK) != Calendar.SUNDAY) {
            calendar.add(Calendar.YEAR, 100);
        }
        System.out.println(calendar.getTime());
    }
}
```

输出:

```
Fri Dec 31 17:32:18 CST 1999
Sun Dec 31 17:32:18 CST 2299
```

BigInteger/BigDecimal

```
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Arrays;

public class test {
    public static void main(String[] args) {
        BigInteger integer1 = new BigInteger("1");
        BigInteger integer2 = new BigInteger("3");
        BigDecimal bigDecimal;
        System.out.println(integer1.divide(integer2));

//          BigDecimal bigDecimal1=new BigDecimal(2);
//          BigDecimal bigDecimal2=new BigDecimal(2.3);
//          BigDecimal bigDecimal3=new BigDecimal("2.3");
//          System.out.println(bigDecimal1);
//          System.out.println(bigDecimal2);
//          System.out.println(bigDecimal3);

        BigDecimal bigDecimal4 = new BigDecimal("1.0");
        BigDecimal bigDecimal5 = new BigDecimal("3.0");
        System.out.println(bigDecimal4.divide(bigDecimal5, 100,
BigDecimal.ROUND_HALF_UP));////获得小数点后100位, 方式四舍五入 a.divide(int b, 保留位数, 舍入方
式)
    }
}
```

输出:

```
0
0.3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
33333333333333
```