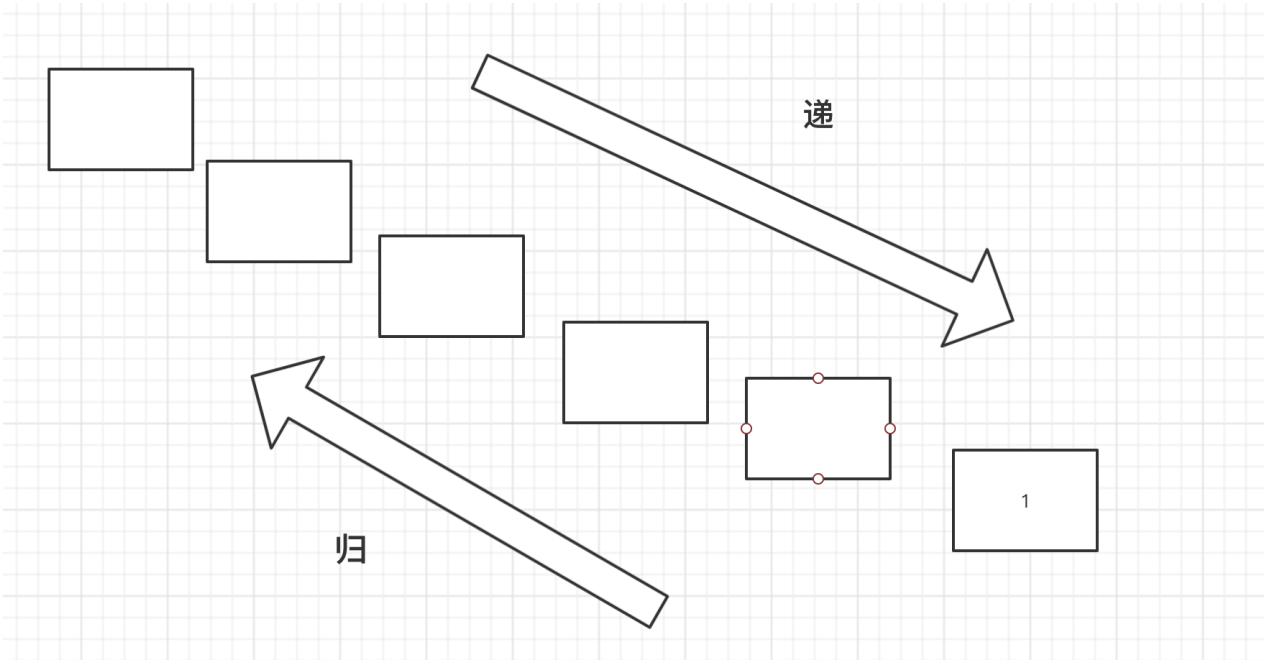
递归

二叉树遍历，深度优先搜索等。

什么是递归?

常规的定义：编程语言中，函数 func 直接或者间接调用函数本身，则该函数称为递归函数。



问前排人是第几排 -> 函数

所有的递归问题都可以用递推公式来表示，所以要用递归解决问题，关键就是先找到递推公式。

```
f(n) = f(n-1)+1
f(1) = 1
```

f(n) 表示你当前是第几排，f(n-1) 前面一排所在的排数，f(1) = 1 表示第一排的人知道自己是第一排。

```
int f(int n){
  if(n == 1) return 1;
  return f(n-1)+1;
}
```

递归需要满足 3 要素：

1、一个父问题可以拆分成若干个子问题，并且若干子问题的结果汇总起来就是父问题的答案。

2、父问题和子问题，解题思路必须完全一致，只是数据规模不同。

3、存在终止条件。

问题在不断拆分的同时，一定要在某个节点终止拆分，得到一个明确的答案。

问题：假设有 n 个台阶，每次可以跨 1 个台阶或者 2 个台阶，请问走完这 n 个台阶一共有多少种走法?

1、假设有 1 个台阶，一共有（1） 种走法

2、假设有 2 个台阶，一共有 2 种走法 【1，1】【2】

3、假设有 3 个台阶，一共有（）种走法? 【1，1，1】【1，2】【2，1】

……

可以根据第一步的走法进行分类

第一类是第一步走了 1 个台阶

第二类是第一步走了 2 个台阶

所以 n 个台阶的走法就等于先走 1 个台阶后，n-1 个台阶的走法+先走 2 个台阶后，n-2 个台阶的走法。

```
f(n) = f(n-1)+f(n-2)
```

f(1) = 1，能否作为终止条件?

n = 2，f(2) = f(1)+f(0)，如果终止条件只有一个 f(1) = 1，f(2) 就无法求解， 因为 f(0) 的值无法确定，

把 f(2) = 2 作为一个终止条件

终止条件有两个：

f(1) = 1;

f(2) = 2;

n = 3，f(3) = f(2)+f(1) = 3

n = 4，f(4) = f(3)+f(2) = 3 + 2 = 5

递推公式

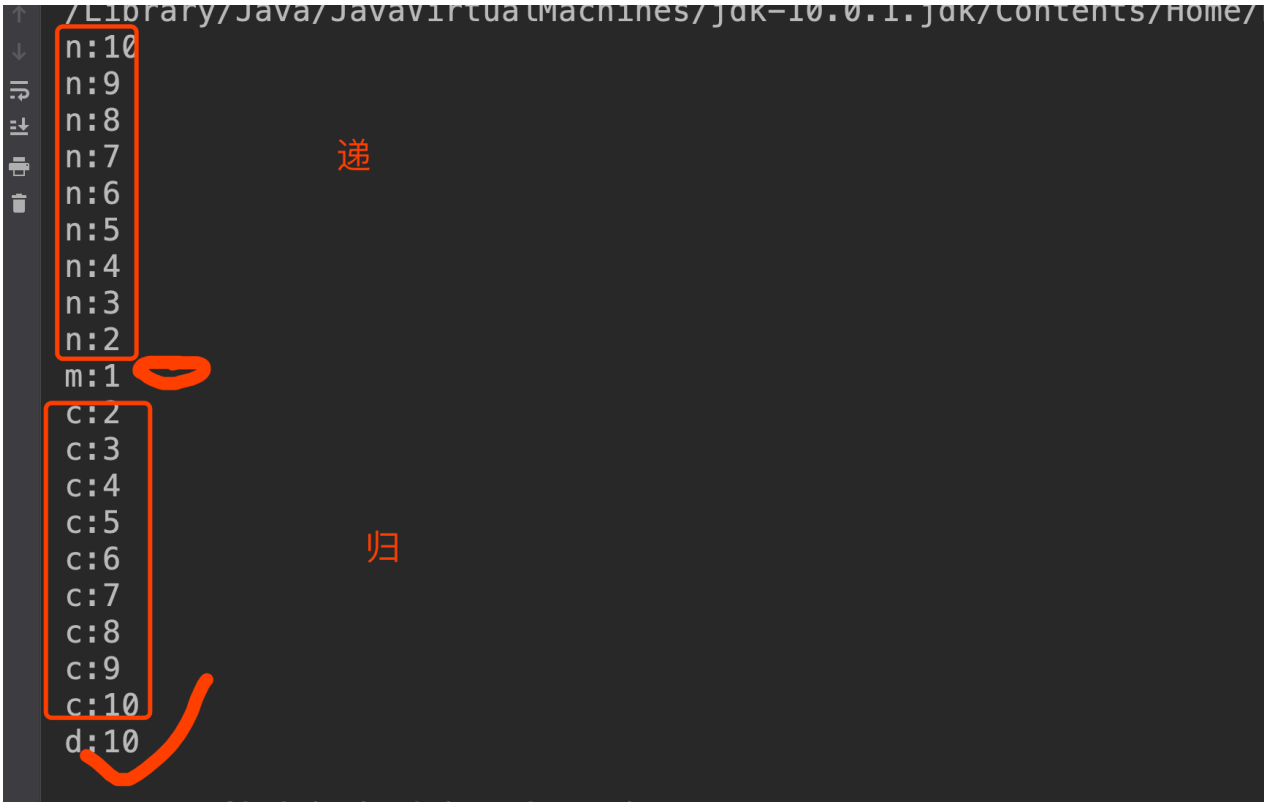
```
f(1) = 1;
f(2) = 2;
f(n) = f(n-1)+f(n-2);
```

推导出递归代码

```
int f(int n){
  if(n == 1) return 1;
  if(n == 2) return 2;
  return f(n-1) + f(n-2);
}
```

```java
package com.southwind.demo;

public class Test {
    public static void main(String[] args) {
        for (int i = 1; i <= 30; i++) {
            System.out.println(i+"个台阶共有"+f(i)+"种走法");
        }
    }

    public static int f(int n){
        if(n == 1) return 1;
        if(n == 2) return 2;
        return f(n-1) + f(n-2);
    }

}
```

```java
package com.southwind.demo;

public class Test2 {
    public static void main(String[] args) {
        int d = f(10);
        System.out.println("d:"+d);
    }

    public static int f(int n){
        if(n == 1){
            System.out.println("m:"+n);
            return 1;
        }else{
            System.out.println("n:"+n);
            //n=4
            int c = f(n-1)+1;
            System.out.println("c:"+c);
            return c;
        }
    }
}
```



集合框架

为什么要使用集合框架?

1、数组的长度是固定

2、数组无法同时存储多个不同的数据类型

集合简单理解就是一个长度可以改变，可以保持任意数据类型的动态数组。

集合本身是数据结果的基本概念之一，我们这里说的集合是 Java 语言对这种数据结果的具体实现。

Java 中的集合不是由一个类来完成的，而是由一组接口和类构成了一个框架体系。大致可分为 3 层，最上层是一组接口，继而是接口的实现类。

接口

Collection：集合框架最基础的接口，最顶层的接口。

List：Collection 的子接口，存储有序、不唯一（元素可重复）的对象，最常用的接口。

Set：Collection 的子接口，存储无序、唯一（元素不可重复）的对象。

Map：独立于 Collection 的另外一个接口，最顶层的接口，存储一组键值对象，提供键到值的映射。

Iterator：输出集合元素的接口，一般适用于无序集合，从前往后输出。

ListIterator：Iterator 子接口，可以双向输出集合中的元素。

Enumeration：传统的输出接口，已经被 Iterator 取代。

SortedSet：Set 的子接口，可以对集合中的元素进行排序。

SortedMap：Map 的子接口，可以对集合中的元素进行排序。

Queue：队列接口。

Map.Entry：Map 的内部接口，描述 Map 中存储的一组键值对元素。

Collection 接口

Collection 是集合框架中最基础的父接口，可以存储一组无序，不唯一的对象。

```
 * @since 1.2
 */

public interface Collection<E> extends Iterable<E> {
    // Query Operations
```

Collection 接口可以存储一组无序，不唯一（可重复）的对象，一般不直接使用该接口，也不能被实例化，只是用来提供规范。

Collection 是 Iterable 接口的子接口。

int size() 获取集合长度

boolean isEmpty() 判断集合是否为空

boolean contains(Object o) 判断集合中是否存在某个对象

Iterator iterator() 实例化 Iterator 接口，遍历集合

Object[] toArray() 将集合转换为一个 Object 数组

T[] toArray(T[] a) 将集合转换为一个指定数据类型的数组

boolean add(E e) 向集合中添加元素

boolean remove(Object o) 从集合中删除元素

boolean containsAll(Collection c) 判断集合中是否存在另一个集合的所有元素

boolean addAll(Collection c) 向集合中添加某个集合的所有元素

boolean removeAll(Collection c) 从集合中删除某个集合的所有元素

void clear() 清除集合中的所有元素

boolean equals(Collection c) 判断两个集合是否相等

int hashCode() 返回集合的哈希值

Collection 子接口

- List：存放有序、不唯一的元素
- Set：存放无序、唯一的元素
- Queue：队列接口

List 接口

```
 * @since 1.2
 */

public interface List<E> extends Collection<E> {
    // Query Operations
```

List 常用的扩展方法

T get(int index) 通过下标返回集合中对应位置的元素

T set(int index,T element) 在集合中的指定位置存入对象

int indexOf(Object o) 从前向后查找某个对象在集合中的位置

int lastIndexOf(Object o) 从后向前查找某个对象在集合中的位置

ListIterator listIterator() 实例化 ListIterator 接口，用来遍历 List 集合

List subList(int fromIndex,int toIndex) 通过下标截取 List 集合

List 接口的实现类

ArrayList 是开发中使用频率最高的 List 实现类，实现了长度可变的数组，在内存中分配连续空间，所以读取快，增删满。

```
package com.southwind.demo2;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add("Hello");
        list.add("World");
        list.add("JavaSE");
        list.add("JavaME");
        list.add("JavaEE");
        System.out.println("list:"+list);
        System.out.println("list长度:"+list.size());
        System.out.println("list是否包含Java:"+list.contains("Java"));
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
        Iterator iterator = list.iterator();
        while(iterator.hasNext()){
            System.out.println(iterator.next());
        }
        list.remove("Hello");
```

```
        list.remove(0);
        System.out.println("******************");
        System.out.println(list);
        list.add(1,"Spring");
        System.out.println(list);
        list.add(1,"Spring Boot");
        System.out.println(list);
        list.set(1,"Spring Cloud");
        System.out.println(list);
        System.out.println("*************");
        System.out.println(list.indexOf("Spring"));
        System.out.println(list.subList(1,3));
    }
}
```