# 第一章、基础

1. 基础编程模型
   1. JAVA程序的基本结构

一个类包括类的定义、类的方法、并且必须包含一个被系统调用的main（）函数

* 1. 原始数据类型和表达式

整型

浮点型

布尔型

字符型

* 1. 语句

声明语句 创建某种类型的变量并用标识符来命名

赋值语句 把一个表达式的值赋予一个变量

条件语句 根据条件来执行两个代码段中的一个

循环语句 只要条件为真就重复执行某一段代码

调用和返回语句

* 1. 静态方法
     1. 静态方法的组成

由签名（）和函数体组成

调用静态方法通过方法名然后在括号内写参数

* 1. 递归

编写递归查找最重要的原则

递归的结果总有一个最简单的情况——方法的第一条语句总是一个包含条件判断的return语句

递归调用总是去尝试解决一个规模更小的问题，这样才能收敛到一个最简单的情况来结束递归

递归调用的父问题和尝试解决的子问题之间不应该有交集 // todo

* 1. 单元测试

每一个静态方法库都有一个main（）方法，借此我们可以调用该库中所有的静态方法，以此来调试

* 1. 基础编程模型

java开发的基本模式是编写一个静态方法库（包含一个main（）方法）来完成一个任务

* + 1. 命令行参数

在命令行执行某个文件的时候可以输入参数，main（）函数可以接收到这个参数

* + 1. 格式化输出

格式化输出函数的第一个参数描述了输出的格式，后面的参数是

* + 1. 重定向和管道

重定向：程序的输入和输出不再通过终端而是通过一个文件，例如将输出结果写入到文件中或者从文件中读取数据作为参数

管道：一个程序的输出作为另一个程序的输入参数

* 1. 二分法查找

用于查找一个整数 key是否在一个有序数列中，他通过定义两个整数 lo 和 ho作为查找范围，每次将key与[lo, ho] 的中间键作比较如果相等则返回否则[lo, ho]所表示的范围将会减小，直到查找范围为空则查找完毕

* 1. 数据抽象

“抽象让我们忽略细节，在不同层次上处理细节，而封装则填补了抽象留下来的空白”

抽象让我们忽略不同对象实体的不同点，而抽离出他们的共同点，而封装则是隐藏了操作对象的具体实现。

1. 背包、队列和栈
   1. 泛型

泛型是指在实现抽象数据结构的时候不指定集合中元素的具体数据类型，而使用Item变量进行占位，用例在创建该数据类型的时候再指定具体的数据类型，给变量Item赋予某种数据类型，然后通过Item实现具体的数据类型

* 1. 迭代

迭代是指用例可以直接遍历集合中的元素，而不必关心数据的具体实现

实现方式是，通过该迭代类的iterator()方法返回一个继承了Itrator接口的内部类的实例化引用，通过该内部类的hasNext()、next()方法来实现对集合中元素的遍历

* 1. 背包

背包是一种不支持删除的抽象数据类型，它的目的是帮助用例收集元素并迭代遍历所有元素，也可以用来判断是否为空以及元素数量，它的应用场景是用于统计分析数据，它其中的数据只有在同一个集合的关系，元素的顺序也不重要

* 1. 队列

队列是一种满足先入先出策略的集合类型，就好像排队购票，先排队的先处理先出队，该类型除了保存元素的值还保存了元素的相对顺序，即出队的顺序和入队的顺序相同

* 1. 栈

栈使用的是后入先出的策略，就好像是处理信件，每到一封信就把这封信放到信堆的上面，而处理信件的时候会从最顶上先处理，他的作用的保存元素的顺序并在遍历使用这些元素的时候颠倒他们的相对顺序

1. 链表
   1. 定义

链表是一种递归的数据结构，它或者为null，或者指向一个结点的引用，该结点包含一个泛型的元素和一个指向另一个链表的引用。

* 1. 下压堆栈（链表实现）

需要保存一个栈顶结点，和栈元素数量，插入元素时将新元素指向旧的链表，删除元素时，将栈顶元素指向下一个结点即可

* 1. 先进先出队列（链表实现）

跟栈不同的是，队列是从队尾插入，从队首取出，所以需要保存队首结点和队尾结点，

插入元素时如果时插入第一个元素那么队首和队尾保存为同一个元素，否则将原本的队尾指向新队尾的引用，将新队尾保存为队列的队尾

取出元素时将队首修改为下一个元素的引用即可

1. 算法分析

算法分析的目的是分析出某个算法运行所需要的时间消耗或者内存消耗

* 1. 对增长数量级的猜想

对于算法分析来说增长数量级即为程序运行时间相对于问题规模的比值，而程序运行时间又与计算机的性能有关，但是对于某个具体型号的计算机来说，这个增长数量级是固定的，

* 1. 成本模型

成本模型即为算法时间成本数量级最大的那些操作。

* 1. 总结

对于大多数程序来说，我们得到其运行时间的数学模型的步骤如下：

确定输入模型，定义问题的规模

识别内循环

根据内循环的操作确定成本模型

对于给定的输入，判断这些操作的频率，

* 1. 2-sum

对于在一个数组中查找和为0的整数对的问题，可以借助于二分查找，即对于某一个元素a[i]，通过二分查找到值为-a[i]的元素就累加数量，通过判断找到的下标是否大于i来防止重复查找

* 1. 3-sum

对于在一个数组中查找和为0的三个整数的问题，可以借助于二分查找，即对于某两个元素a[i]和a[j]，通过二分查找到值为-a[i]-a[j]的元素就累加数量，通过判断找到的下标是否大于j来防止重复查找

1. 案例研究：union-find算法
   1. 1
   2. 1
   3. 1