# 一、数据结构

数据结构是一门研究非数值计算的程序设计问题中的操作对象，以及它们之间的关系和操作等相关问题的学科。

**程序设计 = 数据结构 + 算法**

1. **基本概念和术语**
   1. **数据**

数据：是描述客观事物的符号，是计算机中可以操作的对象，是能被计算机识别，并输入给计算机处理的符号集合。

* 1. **数据元素**

数据元素：数组成数据的、具有一定意义的基本单位，在计算机中通常作为整体处理。也被称为记录。

例如：在人类中，数据元素是人；畜类中，数据元素就是牛马羊鸡猪狗等等。

* 1. **数据项**

数据项：一个数据元素可以由若干个数据项组成。数据项是数据不可分割的最小单位。

例如：人这个数据元素，可以有眼、鼻、嘴、手、脚这些数据项，也可以有姓名、年龄、性别等等。

* 1. **数据对象**

数据对象：是性质相同的数据元素的集合，是数据的子集。

* 1. **数据结构**

不同数据元素之间不是独立的，而是存在特定的关系，我们将这些关系称为结构。

数据结构：是相互之间存在一种或者多种特定关系的数据元素的集合。

1. **逻辑结构与物理结构**
   1. **逻辑结构**

逻辑结构：是指数据对象中元素之间的相互关系。

逻辑结构包括：集合结构、线性结构、树形结构、图形结构。

* 集合结构

集合结构中的数据元素除了同属于同一个集合外，它们之间没有其他关系。

* 线性结构

线性结构中的数据元素之间是一对一的关系。

* 树形结构

树形结构中的数据元素之间存在一种一对多的层次关系。

* 图形结构

图形结构的数据元素是多对多的关系。

* 1. **物理结构（存储结构）**

物理结构：是指数据的逻辑结构在计算机中的存储形式。

物理结构包括：顺序存储结构、链式存储结构

* 顺序存储结构

顺序存储结构是把数据元素放在地址连续的存储单元里，其数据间的逻辑关系和物理关系是一致的。说白了就是排队占位，数组就是这样的顺序存储结构。

* 链式存储结构

链式存储结构是把数据元素存放在任意的存储单元里，这组存储单元可以是连续的，也可以是不连续的。

1. **抽象数据类型**
   1. **数据类型**

数据类型是指一组性质相同的值的集合及定义在此集合上的一些操作的总称。

* 1. **抽象数据类型**

抽象数据类型（Abstract Data Type,ADT）：是指一个数学模型及定义在该模型上的一组操作。

“抽象”的意义在于数据的类型的数学抽象特性。抽象数据类型体现了程序设计中问题分解、抽象和信息隐藏的特性。

抽象数据类型的标准格式：

ADT 抽象数据类型名

Data

数据元素之间逻辑关系的定义

Operation

操作1

初始条件

操作结果描述

操作2

……

操作n

……

endADT

# 算法

算法：是解决特定问题求解步骤的描述，在计算机中表现为指令的有限序列，并且每条指令表示一个或多个操作。

1. **算法的特性**

算法具有5个特征：输入（0个或者多个输入）、输出（至少一个或者多个输出）、有穷性、确定性、可行性。

* 有穷性：指算法在执行有限的步骤之后，自动结束而不会出现无限循环，并且每一个步骤在可接受的时间内完成。
* 确定性：算法的每一步骤都具有确定的含义，不会出现二义性。
* 可行性：算法的每一步都必须是可行的，也就是说，每一步都能通过执行有限次数完成。

1. **算法的设计要求：正确性、可读性、健壮性、时间效率高和存储量低。**
   1. 正确性：算法的正确性是指算法至少应该具有输入、输出和加工处理无歧义性、能正确反映问题的需求、能够得到问题的正确答案。
   2. 可读性：算法设计的另一目的是为了便于阅读、理解和交流。
   3. 健壮性：当输入数据不合法时，算法也能做出相关处理，而不是产生异常或莫名其妙的结果。
   4. 时间效率高和低存储量：设计算法应该尽量满足时间效率高和存储量低的需求。
2. **算法效率的度量方法**
   1. **事后统计方法（不科学、不准确）**

事后统计方法：这种方法主要是通过设计好的测试程序和数据，利用计算机计时器对不同算法编制的程序的运行时间进行比较，从而确定算法效率的高低。

* 1. **事前分析估算方法**

事前分析估算法：在计算机程序编制前，依据统计方法对算法进行估算。

一个程序的运行时间，依赖于算法的好坏和问题的输入规模，所谓问题的输入规模就是指输入量的多少。在分析程序的运行时间时，最重要的是把程序看成是独立于程序设计语言的算法或一系列步骤。

1. **函数的渐近增长**

函数的渐近增长：给定两个函数f（n）和g（n），如果存在一个整数N，使得对于所有的n>N，f（n）总是比g（n）大，那么，我们说f（n）的增长渐近快于g（n）。

判断一个算法的效率时，函数中的常数和其他次要项常常可以忽略，而更应该关注主项（最高阶项）的阶数。

1. **算法时间复杂度**

**算法时间复杂度定义：在进行算法分析时，语句的执行次数T（n）是关于问题规模n的函数，进而分析T（n）随n的变化情况并确定T（n）的数量级。算法的时间复杂度，也就是算法的时间度量，记作：T（n）=O（f（n））。它表示随问题规模n的增大，算法执行时间的增长率和f（n）的增长率相同，称作算法的渐近时间复杂度，简称为时间复杂度。其中，f（n）是问题规模n的某个函数。**

* 1. **推导大O阶的方法**

1. **用常数1取代运行时间中的所有加法常数。**
2. **在修改后的运行次数函数中，只保留最高阶项。**
3. **如果最高阶项存在且不是1，则去除与这个项相乘的常数。得到的结果就是大O阶。**
4. **常见的时间复杂度**
5. 常数阶：O（1），注意没有O（3）之类的的
6. 线性阶：O（n）
7. 对数阶：O（logn）
8. nlogn阶：O（nlogn）
9. 平方阶：O（n2）
10. 立方阶：O（n3）
11. 指数阶：O（nn）

常用的时间复杂度所耗费时间从小到大依次是：

O(1) < O(logn) < O(n) < O(nlogn) < O(n2) < O(n3) < O(2n) < O(n!) < O(nn)

1. **最坏情况与平均情况**

最坏情况运行时间是一种保证，那就是运行时间将不会再坏了。在应用中，这是一种最重要的需求，通常，除非特别指定，我们提到的运行时间都是最坏情况的运行时间。

平均运行时间是所有情况中最有意义的，因为它是期望的运行时间。

1. **算法空间复杂度**

**算法的空间复杂度通过计算算法所需的存储空间实现，算法空间复杂度的计算公式记作：S（n）=O（f（n）），其中n为问题的规模，f（n）为语句关于n所占存储空间的函数。**