链滴

摸一摸数据结构（绪论）

作者：stillwarter
原文链接：https://ld246.com/article/1630318364242
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

ps: 这里的大纲或许和目录不一样，目录只是记录大致内容。

数据结构的背景

数据结构的发展背景：

自 1946 第一台计算机问世，计算机发展速度远远超出人们对其预期；如今计算机应用不在局限于科计算，计算机加工处理对象由纯粹的数值发展到字符，表格等各种具有一定结构的数据，这就给程序来新的问题。（往往一个数据的更改可能会牵涉到其他数据）

为了编写好一个优良的程序，必须分析待处理对象特性以及各处理对象之间的关系，这就是“数据结”形成与发展的**背景**。

记得那天我在回寝室的路上问学的好的室友“学了数据结构之后有什么不同吗？我咋没什么感觉？”“可以让你写的代码更好。”

他的回答很简单，证明我肯定是没怎么学懂这玩意了。

所以为什么要学数据结构？或许可以让你的代码更好。

数据结构定义与基本概念

什么是数据结构?

一般用计算机解决一个具体过程，大致分为以下几个步骤：首先从具体问题抽象一个适当的数学模型设计为此解决此模型的算法，最后编程实现，进行测试，调整得到最后解答。寻求数学模型的实质试析问题，从中提取操作的对象，并且找到对象之间含有的关系。

简单来说，**数据结构**是一门研究 非数值计算的程序设计问题中操作对象以及它们之间的关系和操作等学科。

到大型程序出现后，软件相对独立，结构程序设计成为主流，领域内的人们越来越重视数据结构，认程序设计的本质是为问题确定一组良好的结构，加上设计一组好的算法。

在书上有详尽的例子，我们将这类非数值计算的数学模型提取成数据结构，用于解决问题和后续的编。所以可以理解为数据结构是一种加工后的产物，其来源是实际问题中我们列出的解决问题的模型或程，然后我们根据数据结构进行后续的编码操作。

比如我想知道我有多少头发，首先我可以测量单位面积（c）下我有多少头发（n），然后用头发覆盖域面积（s）除以单位面积（c）乘单位面积的头发（n）就的到总头发量

```
typedef int units;
typedef struct{
    units c;//单位面积c
    int num;
}numc;
int sum(s,numc){
    return (s/numc.c)*n;
}
```

大概就是这样，在这里看不出来数据结构有什么用，但在大型的程序里良好的结构是必要的。

数据结构基本概念

数据 data

数据是客观事物的符号表示，是信息的载体。在计算机科学中是指所有能输入到计算机中被计算机程处理的符号的总称。

数据元素 data element

数据元素是数据的**基本单位**，在计算机程序中通常作为一个整体进行考虑处理。有时一个数据元素可若干数据项（data item）组成，数据项是数据不可分割的最小单位。

数据对象 data object

数据对象是**性质相同**的数据元素的集合，是一个数据的子集。

数据类型 data type

1. 原子类型　　其值不可再分
2. （固定聚合类型）结构类型　　其值可以再分解为若干成分（分量）的数据类型
3. （可变聚合类型）抽象数据类型　抽象数据组织与之相关操作

数据结构 data structure

数据结构是相互之间存在一种或多种关系的数据元素的集合。（简单解释）

注：数据结构这个概念至今为有一个被公认的定义，不同的人使用这个词所表达的意思可能不同。

随便举个例子

我的头发 sumh=100000；sumh 就是数据 data。

人不仅是有头发的，还有四肢，这个四肢有双手双脚构成；四肢对应人的数据元素 data element，双手双脚就是这个数据元素的数据项 data item。

聊天室水友大抵都是有头发的，hair=（sumh1，sumh2，sumh3……）这个集合就是数据对象 data bject

数据结构三要素

1. 数据的逻辑结构

逻辑结构是数据元素之间的逻辑关系，他与数据存储无关，独立出计算机。

比如线性表，线性表是一种线性结构（逻辑上）；树，图是非线性结构。

分类：

    1. 集合：结构中的数据元素除同属一个结合，并无其他关系
    2. 线性结构：结构中的数据元素之间只有一对一关系，比如 a-b-c
    3. 树形结构：结构中的数据元素之间存在一对多关系，比如 a-b;a-c
    4. 图装结构或网状结构：　多对多关系

2. 数据的存储结构

存储结构是指数据结构在计算机中的表示（又称映像）。数据的存储结构是用计算机实现的逻辑结构为逻辑结构服务），其构成有四种

1. 顺序存储

把逻辑上相邻的元素储存在物理位置上连续的单元中，类似于数组。有点是可实现随机存储，缺点是能产生多的外部碎片

2. 链式存储

不要求在逻辑上相邻的元素在物理上也相邻。借助地址指针来表示元素中的逻辑关系。有点是不会出碎片，缺点是只能指针将会占一定空间

3. 索引存储

在存储信息同时附带索引表，优点是检索快，缺点附件占用空间

4.

5. 散列存储

抽象数据类型

一个抽象数据类型包括定义，表示，实现 3 个部分

ATD{
数据对象:{数据对象定义}
数据关系:{数据关系定义}
基本操作:(基本操作定义)
}adt

实现略过，有点像头文件的准备工作。

抽象数据类型（ADT）描述了逻辑结构和抽象运算，通常用（数据对象，数据关系，基本操作集）这的三元组表示，从而构成一个完整的数据结构定义

算法和算法分析

性质

有穷性，确定性，可行性，输入输出。

算法设计需求

正确性，可读性，健壮性，效率与储量，算法度量（时间复杂度，空间复杂度）

正确性：算法应该能正确的求解问题

可读性：算法便于理解

健壮性：输入非法数据是，算法能适当做出反应进行处理，不会产生莫名的输出结果

效率与低储存：执行时间与占据空间的评定

时间复杂度与空间复杂度

算法的效率由时间复杂度和空间复杂度来描述

时间复杂度：指的是一个语句的频度（主要语句的执行次数）在算法中被执行的次数。

常见的渐进时间复杂度：

O(1)<O(log2^n)<O(n)<O(nlog2^n)<O(n^2)<O(n^3)<O(2^n)

空间复杂度：指的是所消耗的存储空间。

举个例子

```
count=0;
for(i=1;i<=n;i=i*2)
    for(j=1;i<=n;i++)
        count++;
```

这里主要执行的语句 count++ 被执行了 n*log2^n，所以其时间复杂度为 O(n*log2^n)

思考

同一个算法，实现语言的级别越高，执行效率越低。

这是严老师的原话，确实越高级的语言抽象的层次更高，需要二重编译或者其他的方式让计算机读懂码，这样一来二会降低执行效率。

个人感觉严老师似乎不满新语言的快速发展取代旧语言，确实随着这个领域的兴起，商业化与工程化我们不会再去过多追寻这短暂而迅猛的计算机发展史，或许这历史也告诉不了我们什么。