第一章 概述

刘亮亮

2020 年 3 月 2 日

- 要点：本章主要介绍数据结构的定义，数据结构的三要素，以及抽象数据类型的概念与定义，算法的概念与算法分析。
- 重点：1.数据结构的定义；2.逻辑结构与存储结构;3.时间复杂度
- 难点：1.抽象数据类型；2.时间复杂度

---

1 数据结构是干什么的？

瑞士计算机科学家Pascal（一门早期的程序设计语言）之父Niklaus Wirth提出了著名的公式:数据结构 + 算法 = 程序，正是凭借这句话，让他获得了1984 年的图灵奖，这个公式对计算机科学的影响力类似物理学中爱因斯坦的$E = MC^2$的影响。公式数据结构 + 算法 = 程序 揭示了程序的本质。

因此，要真正学会编程，既需要懂得数据结构，又要懂得算法。那数据结构是干什么的？顾名思义，数据结构应该是关于数据的结构的课程。那《数据结构》这门课程到底干什么的？在回答这个问题之前，首先看几个例子，

例 1 (学生成绩表). 学生成绩表中的每一行除了第一行和最后一行外，每一行的前面都有一行，后面都有一行，我们可以对学生成绩表进行查找、插入、删除、排序等操作，这种结构都有一种特性：“一对一”的线性关系。

例 2 (资源管理器的结构). 大家打开*Windows*操作系统的资源管理器，“我的电脑”下面有多个硬盘的分区（如：*C*，*D*，*E*），每个分区都有多个文件夹和文件，每个文件夹下又有子文件夹和文件....，这种结构存在一种性质，除了最上层的“我的电脑”外，每个文件夹或文件都只有一个上层的节点（一般称为父节点），除最末端的节点，每个结点都可以有一个或多个子文件夹或文件，这种结构是什么样子？大家倒过来看，是不是和窗外的树是类型的。类似于这种结构这种特性是“一对多”的树形结构性质。

例 3 (城市的交通图). 现在大家出门都依靠导航，出门前都会打开导航搜索线路，而城市的交通图就是一种图形结构，从一个地方到另一个地方都有很多条路径，交通图中的点都和多个相邻的点有关联。图中的节点都存在“多对多”的关系。

上面三个例子的这些结构在现实生活中大量出现，正如事物之间的关系一样，数据之间总会存在着一种或多种特定的关系。特别地，我们在计算机中如何去存储和表示这些具有特点关系的数据？并且对这些数据有哪些操作，如何实现这些操作？

并且大家在写程序过程中，一个具体的问题，其解决办法也取决于其采用什么的结果来进行存储，不同的存储可能解决办法也不一样。

因此，数据结构是研究非数值计算的程序设计问题中计算机操作对象（即数据）以及它们之间的关系和操作的学科。数据结构是编译原理、操作系统、程序设计等课程的基础课程，也是介于计算机软件、计算机硬件和数学之间的一门学科。

2 为什么学习数据结构？

随着计算机的发展，大型程序的出现，结构化程序设计成为程序方法设计学的主要方法，人们也越来越重视“数据结构”，认为在编程的过程中实际上是选择一种好的结构来存储和表示数据，并且设计一些好的算法来操作这些数据从而解决具体的问题。因此，数据结构在程序设计中占有重要的地位。

并且大家在实现一个具体的问题的时候，首先都是考虑如何选择一种结构来存储和表示要解决的具体的问题的数据，然后设计相应的算法，而算法的实现也取决于采用什么样的存储结构。因此数据结构是程序设计中首要解决的问题。

此时有些同学会想到，现在的高级编程语言中已经实现了很多的结构，直接可以用，比如Python语言中的list、dictionary等等，那我们为什么还要学习数据结构呢？一方面我们需要了解类似于这种数据结构实现的原理，另一方面，在具体的应用问题中，数据的结构远超过一门语言中定义好的结构，我们需要设计更复杂的结构来存储和表示。因此非常有必要来学习数据结构的知识。

3 基本概念与术语

在定义数据结构的概念之前，首先定义一些概念和术语，最后再引出数据结构的概念。

定义 1 (数据)**.** 在现实生活中，万物皆数据。在计算机领域，一切能输入到计算机内部的符号的总称，都叫数据。它是计算机可操作的对象，能被计算机识别和处理。特别在大数据时代，数据成爆炸性增长，出现了各种各样的数据，如结构化数据和非结构化数据。因此类似于音频、视频、图像、字符、多媒体数据等都是数据。

定义 2 (数据元素)**.** 又称为记录，是组成数据的、有一定意义的基本单位，在计算机中一般作为一个整体来处理。

比如说，整型这种数据中的数据元素就是整数，人类中的人就是数据元素。

定义 3 (数据项)**.** 一个数据元素是由若干个数据项来构成的。例如一个具体的人，他由姓名、性别、体重、身高等多个数据项构成。

数据项是数据不可分割的最小单位。

定义 4 (数据对象)**.** 性质相同的数据元素的集合，它是数据的子集。

何为性质相同？是指数据元素具有相同数量和类型的数据项。比如说人都姓名、性别、体重等相同的数据项，而其他类型的数据就不一定有这些数据项。

在一个特定的应用问题中，往往处理的数据元素都具有相同的性质，一般我们泛称为数据对象为数据。

有了这些概念和术语，可以定义数据结构的定义。

定义 5 (数据结构)**.** 具有一种或多种特定关系的数据元素的集合。

计算机中，数据元素往往并不是孤立的，杂乱无章的，而是存在着一种或多种特定关系，这种关系也称为组织方式，换句话说用什么方式来组织这些数据。

因此，在编写程序的时候，首先要分析数据之间存在什么样的关系，才能设计一种好的组织方式来表示数据。这也就是我们研究数据结构的意义。

问题是：数据存在一种或多种特定关系？是什么样的关系呢？

4 逻辑结构与存储结构

定义 6 (逻辑结构)*. 指数据对象中的数据之间的相互关系*。

这个关系是指现实中的数据关系，因此称为逻辑结构。具体的问题的数据有具体的逻辑结构，一般现实中的逻辑结构包括以下几种：

- 集合结构：数据元素只有“同属于”一个集合的关系，没有其它的关系。这种集合结构类似于数学中的集合。
- 线性结构：数据元素之间存在“一对一”的关系。比如说某一条地铁线中的各站之间的关系。
- 树形结构：数据元素之间存在“一对多”的关系。比如说：资源管理器的结构、家谱、行政关系图等；
- 图形结构：数据元素之间存在“多对多”的关系。比如说：地铁图、交通图、社交网络图等等。

定义 7 (存储结构)*. 又称为物理结构，是指数据在计算机的存储形式，用物理结构来反映数据的逻辑结构。*

如何用物理结构来反映数据的逻辑结构，是一个重点和难点。

两种存储结构：顺序存储结构和链式存储结构。

- 顺序存储结构：把数据元素存在在地址连续的存储单元里面，数据的逻辑关系和物理关系是一致的。
- 链式存储结构：把数据元素存放在任意的存储单元里面，“任意”的意思存储单元可以连续，也可以不连续。

顺序存储结构是逻辑上相邻，物理上也相邻。而链式存储结构则不是，它通过指针来表示这种相邻关系。也就是说一个元素存储的时候，同时存放了它后继的元素的地址（指针），通过这个地址来指向它的相邻的元素，这种存储结构比较灵活，只需要指针就可以找到对应的元素，这种指针构成了无形的“链”。

具体采用什么样的存储结构，需要看具体的应用，后面的内容会详细来介绍每种数据结构的存储方式，并且会介绍在某种存储结构下的算法的实现是如何的。因此，算法的实现依赖于存储结构。

5 抽象数据类型

抽象是指从具体事物抽出、概括出它们共同的方面、本质属性与关系等，而将个别的、非本质的方面、属性与关系舍弃的思维过程。抽象是一种非常重要的能力，是隐藏复杂的细节，只保留实现目标所必须的信息。在程序设计中，特别是面向对象程序设计中，抽象是一个重要的概念，通过类来定义抽象数据类型。因此，要学会抽象的能力。

定义 8 (抽象数据类型(Abstract Data Type，ADT))**.** 是指一个数学模型以及定义在该模型的一组操作。

抽象数据类型只是定义一种数据类型的逻辑特性，和其在计算机里面的实现和表示没有关系。换句话说，就是定义这种数据类型的包含什么样的数据、存在什么样的关系以及需要具备哪些操作。具体计算机如何存储和实现，不需要在抽象数据类型定义体现。

比如一个游戏里面的魔法师，这种数据类型，他具备一些数据特性，比如造型、名字、身穿什么装备等，他还具备一些操作，比如说可以发魔法等。

抽象数据类型的定义用关键字ADT来进行定义，示例如下，

定义 9 (抽象数据类型定义格式)**.**

ADT 抽象数据类型名称

{

数据对象：*D*=数据对象的定义

数据关系：*R*=*R1*

基本操作：

操作名称*1*(参数列表)

初始条件：

操作结果：

操作名称n(参数列表)

初始条件：

操作结果：

}抽象数据类型名称.

下面通过复数的定义这种抽象数据类型来介绍抽象数据类型的定义。复数是由两个部分构成的，一部分是实数部分c_1，一部分是虚数部分c_2。这两个部分可以定义一种序的关系，利用关系$R =< c_1, c_2 >$，其中c_1是实数部分，c_2是虚数部分。可以根据两个实数定义构造一个复数，以及定义复数的加法，取复数的实数部分，取复数的虚数部分等操作，定义如下。

例 4 (复数的定义).

ADT ComplexData{

数据对象$D = c_1, c_2 | c_1, c_2$

数据关系$R = < c_1, c_2 > | c_1 c_2 J$

基本操作：

$Init(\&c, c1, c2)$:

操作结果：根据两个实数构造一个复数c

$GetReal(\&c, \&c1)$:

初始条件：复数c存在c

操作结果：用$c1$返回复数c的实数部分

$GetImaginary(\&c, \&c2)$:

初始条件：复数c存在c

操作结果：用$c2$返回复数c的虚数部分

$Add(\&c, \&c1, \&c2)$:

初始条件：复数$c1$,$c2$存在

操作结果：用c返回两个复数$c1$，$c2$的和

....

}*End ComplexData.*

6 算法与算法分析

算法是什么，算法与程序是什么关系，可能很多初学者都比较迷惑，对于这种概念，写过的程序是不是一个算法。其实前面已经给出了算法与程序的关系：数据结构 + 算法 = 程序。下面我们首先给出算法的定义。

6.1 什么是算法？

在给出算法定义前，我们首先看两个例子。

例 5. 如何做“西红柿炒鸡蛋”？

Step1 取2-3个西红柿洗干净

Step2 用开水烫一下把西红柿，去皮

Step3 将西红柿切成块

Step4 打两个鸡蛋，搅拌好

.....

看第二个例子，大家看过赵本山和宋丹丹的小品里面的一个笑话：如果将大象放入冰箱？

例 6. 如何将大象放冰箱？

Step1 把冰箱门打开

Step2 把大象推进去

Step3 把冰箱门关上

当然第二个例子只是一个笑话，不能实现的。但是这两个例子都有一个共性，就是对解决具体问题的步骤。当然对于某个特定问题有很多种方法来求解，比如说“西红柿炒鸡蛋”也有很多种做法，有些人先炒鸡蛋后炒西红柿，有些人先炒西红柿后炒鸡蛋，有些认还放糖等等。但是不论什么方法，它都是一系列的步骤集合。当然大家从这两个例子可以看到，第一个例子是可行的，第二个例子是不可行的。并且不论什么方法，它肯定在有限步骤能完成。因此，算法的定义如下：

定义 10 (算法(Algorithm)). 是对特定问题求解步骤的描述，它是指令的有限序列。

对特定问题求解的描述是算法的定义，并且解决问题的方法也不唯一。但是算法首先是正确的，否则这个算法也不能解决问题。另外是指令的有

限序列，因此算法是在执行有穷步就可以结束。并且大家看“西红柿炒鸡蛋”这个问题，需要做这道菜，需要有“西红柿”、“鸡蛋”等材料，这些称作为输入，做好以后，会得到“西红柿炒鸡蛋”这道菜，这个称为输出。因此根据分析，算法必须具有如下性质：

- 有穷性：一个算法在执行有穷步后结束，并且每一步都会在有限的时间内执行完。
- 确定性：每一个步骤都有确定的含义，不会出现模棱两可的意思，存在二义性。并且在步骤执行的过程中，相同的输入一定是相同的输出，执行的路径也是一样的。
- 可行性：一个算法首先必须是可行的，每一步都可以通过实现的基本操作或步骤来执行。
- 输入：一个算法可以有一个或多个输入，也可以没有输入。
- 输出：一个算法必须有一个或多个输出。

除此之外，一个算法首先必须是正确的，还有就是一个算法必须具备健壮性，可读性要好等特点。

既然一个特定问题有多种方法实现，那么哪个算法好呢？也就是说，如何评价一个算法的好坏？

6.2 如何评价算法？

先看一个例子，如计算“1...100”的和，大家可能会写出如下算法：

算法1：

$int\ sum = 0;$

$for(i = 1; i <= 100; i++)$

{

$\quad sum += i;$

}

我们评价一个算法，一般能想到就是它执行效率高不高？那上面的算法效率高？不一定吧，看看少年高斯的解法。

算法2：

$int\ sum = (1 + 100) * 100/2;$

很显然，少年高斯的算法更优秀，计算机只需要运行一次，而循环的方法，需要运行100次才能进行求和。当然对于这个问题来说，计算机执行1次和执行100次没有明显的区别，但对于一个数据量比较大又更为复杂的问题来说，差异就会体现。

那如何评价算法的优劣呢？————采用**时间复杂度**和**空间复杂度**来度量。

6.3 时间复杂度与空间复杂度

一般衡量一个算法的好坏除了从正确性、可读性等来衡量外，还主要从两个角度进行分析，一个是算法运行的时间，一个是算法所需要的存储空间。当然，我们可以写完算法后，运行算法来计算算法的运行时间，但是这是事后统计。我们需要对算法进行事前统计，这个统计并不一定精确，而是估计一个数量级。那如何估计算法的运行时间呢？首先看几个例子，

例 7. *x++;*

例 8.
```
x = 0;
while(x < n){
    x++;
}
```

例 9.
```
x = 0;
for(i=0;i < n;i++)
{
    for(j=0;j < n;j++)
    {
        x++;
    }
}
```

分析：在例7中，语句“x++”执行1次，例8中第一行执行1次，”x++”执行n次，$x < n$执行n次，总共执行$2n + 1$；在例9中，第一行代码执行1次，第一重循环中“i++”执行n次，内循环执行“j++”执行$n * n$次，$x++$执行$n * n$次，总共大约是$2n^2 + n + 1$次。

可以看到，运行次数和问题的规模n是有关系的，我们在估算的时候，用问题规模的同阶函数来估算时间复杂度。 关于“O”符号的定义，

$f(n) = O(g(n))$表示存在一个正常的常数c和$n0$，使得对所有的$n >= n0$，有$f(n) <= c * g(n)$。 大写O符号给出了函数f的一个上限。

定义 11 (时间复杂度). 算法中基本操作的重复执行的次数是问题规模n的某个函数$f(n)$,算法的时间量度记作 $T(n) = O(f(n))$。 随着问题规模n的增大，时间增长率和$f(n)$增长率相同。

在例7中,$f(n) = 1$,时间复杂度为$O(1)$，例8中,$f(n) = 2n + 1$,时间复杂度为$O(n)$，例9中,$f(n) = 2n * 2 + n + 1$,时间复杂度为$O(n^2)$。

时间复杂度$T(n)$按数量级递增顺序：$O(1) < O(log_2 n) < O(n) < O(nlog_2 n) < O(n^2) < O(n^3) < ... < O(n^k) < O(2^n)$

一般而言，尽可能减少时间复杂度。

定义 12 (空间复杂度). 空间复杂度算法所需存储空间的度量记作$S(n) = O(f(n))$,n是问题的规模。

算法的存量包括：输入数据所占空间、程序本身所占空间、辅助变量所占空间。

时间复杂度与空间复杂度一般而言是互斥的，一般低的时间复杂度会比较高的空间复杂度，低的空间复杂度会比较高的时间复杂度。一般在改进算法的时候，通常会利用空间来换时间或者使用时间来换空间。

6.4 小结

本章主要介绍了数据结构的概念，并且数据结构的三要素进行了介绍，最后介绍了算法及算法分析。

- 数据结构：具有一种或多种特定关系的数据元素的集合。
- 数据结构的三要素：逻辑结构、存储结构与操作
- 逻辑结构与存储结构
  - 逻辑结构：集合（同属于一个集合）、线性结构（一对一的关系）、树形结构（一对多的关系）、图形结构（多对多的关系）
  - 存储结构：顺序存储结构（一组地址连续的存储单元）与链式存储结构（地址可以连续，可以不连续）
- 抽象数据类型：一个数学模型以及一组操作的总称。

- 算法：对特定问题求解步骤的描述，指令的有限序列。五大特性：确定性、有穷性、可行性、输入、输出。
- 算法分析：时间复杂度与空间复杂度

7 练习题

1. 数据结构的三要素是_____、_____、_____。

2. 从逻辑结构分，可以分为集合、_____、_____、_____等四种结构。

3.地址连续来表示逻辑相邻的存储结构是_____,通过指针来表示逻辑相邻的存储结构是_____。

4.算法是_____，其五大特性是_____、_____、_____、_____、_____。

5.算法必须要有输入，这个观点是_____，算法可以有一个或多个输出是_____。

6.for(i=0;$i < n/2$;i++){x++;}的时间复杂度是_____。

7.以下代码的时间复杂度是：（　　）

$for(i = 0; i < n; i++)$

{

　　$for(j = i; j < n; j++)$

　　$x++;$

}

A.$O(n)$　　B.$O(n*(n-i))$　　C.$O(n^2)$　　D. $O(nlog_2n)$

8. 定义三元组(e_1, e_2, e_3)的抽象数据类型定义。

9. 思考题：想想现实生活中哪些数据的结构是线性结构、树形结构及图形结构？

8 Next

下一章介绍第一类结构————线性结构，并且介绍一种简单灵活的线性结构：线性表。具体内容如下：

- 线性结构的特点
- 什么是线性表
- 线性表的抽象数据类型定义
- 线性表的顺序存储结构及其实现
- 线性表的链式存储结构及其实现。