大学学长亲述的数据结构“绕坑”指南

脑洞大开

数据结构另类攻略

刘隽良　编著
胡　华　王立波　主审

西安电子科技大学出版社
http://www.xduph.com

扫一扫　得攻略

脑洞大开

——数据结构另类攻略

刘隽良　编著

胡　华　王立波　主审

西安电子科技大学出版社

图书在版编目(CIP)数据

脑洞大开：数据结构另类攻略 / 刘隽良编著. —西安：西安电子科技大学出版社，2017.12
ISBN 978-7-5606-4712-8

Ⅰ. ① 脑… Ⅱ. ① 刘… Ⅲ. ①数据结构—研究 Ⅳ. ① TP311.12

中国版本图书馆 CIP 数据核字(2017)第 251588 号

策划编辑 陈 婷 马乐惠
责任编辑 马 静 陈 婷
出版发行 西安电子科技大学出版社(西安市太白南路 2 号)
电 话 (029)8824288588201467 邮 编 710071
网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com
经 销 新华书店
印刷单位 陕西天意印务有限责任公司
版 次 2017 年 12 月第 1 版 2017 年 12 月第 1 次印刷
开 本 787 毫米×1092 毫米 1/16 印 张 21
字 数 421 千字
印 数 1～3000 册
定 价 42.00 元
ISBN 978-7-5606-4712 -8/TP
XDUP 5004001 -1

真相，不只有一个

——前言什么的

嘿嘿，又是前言时间喽。

一、一点点不算感悟的感悟

嗯，表示对于从小就看柯南的孩子而言，“真相，只有一个”这句话貌似已经在脑袋瓜里根深蒂固了。

嘿嘿，不过在数据结构里，这句话又要被毁三观喽。 

因为，在数据结构中，没有绝对的正确或者错误，有的只是合适和更合适。我们要做的就是从这些合适与更合适中找出最赞的一种来构建我们的梦。

是的，是筑梦哦，不是摆代码。 

这也是这本书的初衷，数据结构，不应该仅仅是代码的简单堆叠，而是一种砖头类的东西。学过它之后，想构建什么，就是你的事情喽。

在学的过程中，还是那句话：靠的是自己。所有的书啊、代码啊啥的，依然都是辅助。只有你自己搞明白了这段代码为什么这么写，才是真正有收获，语言的学习永远都是一种感悟的过程。

而且，这回的感悟更有意思。

因为，它的真相不只有一个，你的每次感悟都未必是最合适的解决方案，所以你还有更大的感悟空间，这种看似没有止境的特点也算是数据结构的美吧。 

其他的，我们慢慢来。 

二、写作缘由与经历

这本书的初稿完成于 2014 年 12 月，是我第一次在学校学习数据结构的时候，基本上可以算是学校的数据结构一开课，我就开始写，刚好在考试前完成了全书的初稿。起初的原因是对当时学校所用教材中知识的讲述方法有点“怨念”，觉得知识不应该这样枯燥，应该是很立体且很有趣的，加之自己当时对数据结构也蛮有兴趣，觉得可以通过自己的努力，将这些看似很抽象的数据结构通过另类的方法让它变得具体多彩。于是，这本书的初稿就诞生了。书中的语言风格和行文方式以及内容编排都有自己的特点，这也直接决定了这本书的与众不同。现在的我，对初稿进行了多次完善，使本书能够以更完美的姿态展现在大家面前。

由于写作本书的出发点是不把它作为一本传统的教材，所以全书的框架设计、内容逻辑相对于教材有较大区别。为了能够让大家更容易、更轻松地领悟数据结构，我对本书的知识框架做了较大的调整——我们会从最熟悉的数组结构入手，层层递进，增加大量互动

性问答来降低难度等级，增加闯关趣味性，同时辅以大量图片辅助理解并搭配各种小问题一起研究，较好地摆脱了传统书籍的说教式知识传授过程。此外，在这本书中我们将更加注重细节，对大量不被提及的细节不再人云亦云而是告诉你为什么会这样，让你能够更好地理解和掌握数据结构本身。

希望这样的设计能给大家带来更好的学习体验。

三、本书结构

本书主要分成了 9 章：

第 1～2 章是一个开头总结和引导，分别介绍了数据结构和算法，告诉大家一切并没有想象得那么难。

第 3～7 章每章都介绍了一种或两种数据结构，分别为数组和串、链表、栈与队列、树及图。

第 8～9 章是对排序和查找算法的趣味研究。 

中间你可能会碰到很多“坑爹”的问题，别怕，很正常，坚持读下去，并配合以实践，相信我，不久你便会豁然开朗。 

四、致谢

首先依然还是要感谢父母，他们一如既往地支持我做自己感兴趣的事情。在本书的成书过程中，杭州电子科技大学胡华副校长和王立波老师对书稿进行过多次审核，提出了很多很有价值的修改意见，非常感谢他们的付出，使得这本书能够以更加完善的姿态展现在读者面前；同时要感谢西安电子科技大学出版社的支持，尤其感谢编辑陈婷老师和编审马乐惠老师在本书出版过程中提供的诸多帮助。

感谢母校杭州电子科技大学对本书出版的全额经费支持。

五、求“勾搭”

当然，毕竟金无足赤，人无完人，更何况我自己也还远远达不到真正的高手水平……所以书中一定还会有所不足以及这样那样的问题，所以大家如果发现了什么瑕疵或者对这本书有更好的建议，随时欢迎沟通交流指(gou)教(da)。问题肯定还有很多，所以依然需要你辩证地看它喽。

联系邮箱：ddizxt@126.com。

最后，希望这本书能对你有所启发哦。 

刘隽良

2017/4/9

于杭州电子科技大学

目　录

第 1 章　哪有那么难.....1

1.1　什么是数据结构？.....1

1.2　到底都学些啥？.....2

1.3　什么是抽象数据类型(ADT)？.....4

1.4　什么是逻辑结构？.....6

1.5　什么是物理结构？.....7

1.6　为什么会有这么多数据结构咧？.....8

第 2 章　哎呀　算法.....10

2.1　什么是算法？.....10

2.2　算法效率的度量方法.....13

2.3　算法的时间复杂度和空间复杂度.....14

第 3 章　从数组和串说起.....17

3.1　数组内存的静态分配和动态分配.....17

3.2　一维数组的访问.....18

3.3　一维数组的遍历.....21

3.4　一维数组元素的插入和删除.....26

3.5　二维数组以及假如没有二维数组.....39

3.6　有一种矩阵叫稀疏矩阵.....43

3.7　什么是串？.....45

3.8 字符串的基本处理 48
3.9 字符串略微高级点的处理 62
第 4 章 另一个重要的东西：链表 66
4.1 什么是链表？ 66
4.2 单向链表 68
4.3 单向循环链表 78
4.4 双向链表 96
4.5 链表的遍历和连接 100
4.6 链表结点的插入和删除 106
4.7 链表的反转以及静态链表 123
第 5 章 学以致用——栈与队列 129
5.1 什么是栈？什么是队列？ 129
5.2 栈和队列的实现 131
5.3 栈与队列实现的细节技巧 138
5.4 栈的应用之一：递归？(大雾) 153
5.5 栈的应用之二：回溯算法 159
5.6 栈的应用之三：简易文字处理器 164
5.7 队列应用：好长的代码 167
第 6 章 画棵树吧 170
6.1 什么是树 170
6.2 树的存储结构 173
6.3 什么是二叉树？它是树吗？ 180

6.4 二叉树的存储结构 184
6.5 二叉树的遍历 186
6.6 二叉树的构建 190
6.7 二叉树的查找 203
6.9 二叉树的复制 209
6.10 线索二叉树 213
6.11 树、森林和二叉树的转换 222
6.12 哈夫曼树和哈夫曼编码 225
第 7 章 无图 无真相 229
7.1 什么是图 229
7.2 图的表示法 232
7.3 图的遍历 241
7.4 最短路径计算 262
7.5 最小生成树 264
7.6 有向图的拓扑排序 268
第 8 章 查找的基础：排序 271
8.1 经典的回顾：冒泡排序法 271
8.2 又是老朋友——选择排序法 273
8.3 插入排序法 274
8.4 希尔排序法 277
8.5 快速排序法 280
8.6 二叉查找树排序法 286

8.7 顺带一提的堆排序 289
第 9 章 最后，该查找啦 293
9.1 顺序查找 293
9.2 二分查找 295
9.3 索引查找 299
9.4 二叉查找树查找 300
9.5 平衡二叉树(AVL 树) 315
9.6 B-树和 B+树 317
9.7 了解一下哈希查找 324
会是终结吗？嘿嘿 当然不会 326
参考文献 327

第 1 章　哪有那么难

数据结构这个内容，貌似被好多人神话了，如果大家初次接触数据结构时，看到的是使用了伪代码和 ADT 描述的偏于理论化而相对缺乏应用代码和过程细致图解的书籍，心里就会更加“没底”，从而产生了畏惧抵触心理。其实，数据结构真的没有那么难，它也可以很立体、很生动、很有趣。换句话说，帮大家摆脱对数据结构的“神话式设定”，便是本书的目标。

哦，当然，我没有鄙视的意思，那类书从应试上讲是经典的。虽然你可以靠背那类书的知识点通过考试，却可能永远不能领悟到编程这门艺术的真谛。所以在本书里，我会从另一种不同的角度带你进入数据结构的世界。那么，现在就开始吧！

1.1 什么是数据结构?

说了这么多，到底啥是数据结构咧？其实这个概念本身就十分难定义……简单说，数据结构是计算机存储、组织数据的方式，但我相信大家都有自己的理解。嘿嘿，这也就是说每个人从经验中领悟出来的东西不同啦，所以要想真心掌握，还是需要自己去领悟啦。

我对数据结构的理解是：它是数据存储的形式和信息的组织方式，对于不同的问题，通过使用合适的数据结构，并在数据结构中使用恰当的算法，就可以达到高效率的目标。也就是说，要想解决一个问题，合适的数据结构和恰当的算法都是不可或缺的(说到算法，一想到它要用到数学，顿时全是眼泪啊)。比方说解决“从屏幕读取多个整数并按其倒序输出”这个问题，我们需要的是一个用来存放数据的整型数组和一个逆向输出的 for 循环语句，那么其实这个数组就是这个问题中用到的数据结构，而 for 循环就算是和它搭配的算法。

所以数据结构不是什么高大上的东东，它就是一种在解决问题时用到的存储数据的方式。从各数据之间的联系(即逻辑)上它分为线性结构(数组、链表、串、栈和队列等)和非线

性结构(二维以上数组、广义表、树和图等)。再细点分的话，非线性结构包括集合结构(集合)、树形结构(二叉树)和图形结构(图)。从数据的物理存储方式上它分为顺序存储方法(数组等连续存储的结构)、链接存储方法(链表等线性表)、索引存储方法(二叉树等)和散列存储方法(图等)，对于这些结构的详情现在不知道没关系，后面都会详细解说。

也就是说，其实从你开始了解 C 语言的那一刻起，数据结构就已经悄悄存在于你编写代码的字里行间啦。你解决的每个问题中都曾或多或少用到过不同的数据结构哦，只不过这次咱们要更加深入一点，学习一些更方便的结构而已。所以，没必要担心学不会哦。

1.2 到底都学些啥?

其实数据结构有好多种，咱们在这里只会讲一些最常用也是最实用的。在这里先把要讲的东东做个总括和简介，你会发现，它们真的没有那么难。

1. 线性结构

咱们先从线性结构讲起，主要会讲链表、栈、队列、串和数组五种结构。

链表说白了它就是一种各结点在物理地址上不连续或无规律、通过指针彼此连接而成的线性结构。每一个存储点叫结点，其中包含存储内容的空间和其他结点地址的空间。它可以是单向的，也可以是双向的。单向的链表每个结点就只有一个指针，指向下一个结点；双向的有两个指针，一个指向下一个结点，另一个指向上一个结点。如果把头结点的地址赋值给尾结点指针的话，就可以形成环状结构，就是循环链表啦。

三种链表的示意图如图 1-1 所示。

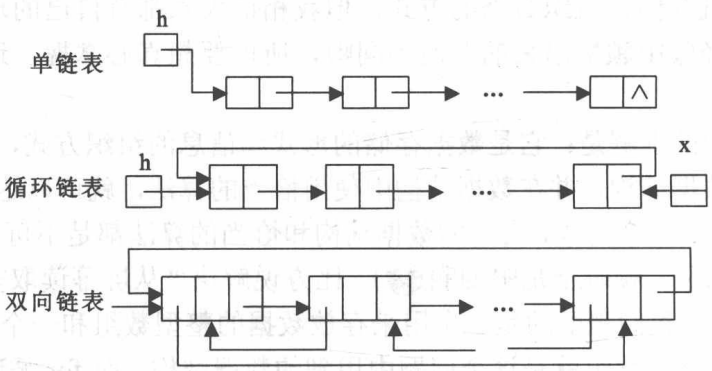

图 1-1

栈这个东东听起来是不是有点耳熟咧？嘿嘿，对啦，在讲 C 语言的时候咱们讲过 C 的内存分配，里面就有一个栈空间。但是此栈并非彼栈哦，C 语言中的栈是一种内存空间的

名字，而数据结构里的栈结构是一种拥有和栈空间类似工作原理的数据结构——“后进先出”，如图 1-2 所示。

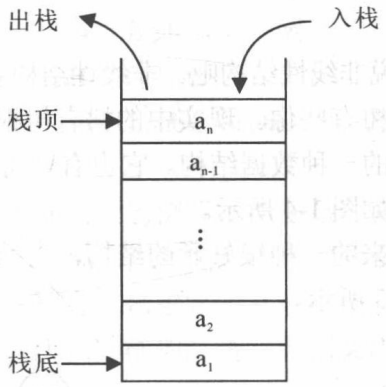

图 1-2

所谓的后进先出，指的是越早入栈的东东离栈底越近，反之越晚放的东东离栈顶越近；而出栈是从栈顶开始的，所以越早进栈的东东越晚出栈，就像咱们讲函数嵌套的时候，嵌套在越里面的函数进栈越晚，出栈越早。栈结构就是一种符合栈空间这种规则、将存储的数据后进先出的结构。它既可以通过线性结构的链表实现，也可以通过线性结构的数组实现，后面两种都会讲哦。

说到队列，它跟栈刚好相反。栈是后进先出结构，队列则是先进先出结构，如图 1-3 所示。顾名思义，它就是一种类似排队的结构。从队头开始进入队列结构，又从队头开始走出队列结构，银行的排号系统就是典型的队列。

队列也是一种既可以通过线性结构的链表实现也可以通过线性结构的数组实现的结构。当然，这两种也都会讲的。

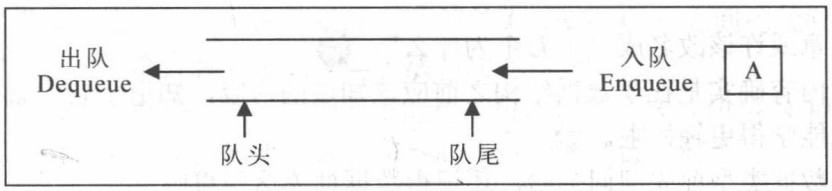

图 1-3

串这个名字听起来十分高大上，但是！注意重点是“但是”，这家伙的“小名”你知道是啥吗？字符串!! 而数组的话，也是老朋友啦。

嘿嘿，所以，你懂的。

链表和数组是后面所有数据结构的基础，所以一定会详细介绍。

跟注重学习原理和操作的数组和链表比起来，等咱们讲栈和队列的时候将会更着重于

应用。嘿嘿，因为数组和链表讲究学会的操作和原理多，以便于用它们构建其他数据结构时更方便，而栈跟队列更讲究应用时的技巧哦。

2. 非线性结构

说完了线性结构，再来说说非线性结构吧。非线性结构主要讲树和图两种结构。

至于树嘛，它跟现实中的树有些像。现实中的树有树根、树干和树叶，而数据结构中的树就是模仿了现实中树结构的一种数据结构。它也有“树根”、“树干”和“树叶”，不过名字有些改变。树结构示意图如图 1-4 所示。

图是在树的基础上演变出来的一种很好玩的结构，它的内容比较多变，后面章节会慢慢讲的。图结构示意图如图 1-5 所示。

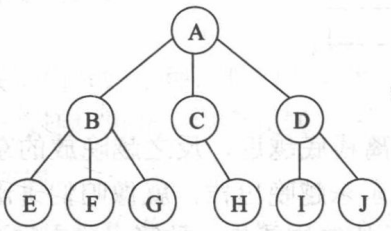

图 1-4

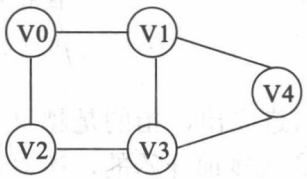

图 1-5

接下来的内容里这些结构咱们都会接触到的。所以，一切，没有那么难哦。那接下来就开始吧。

哦，不对，在此之前，还有一点点东西要介绍，它就是……

1.3 什么是抽象数据类型(ADT)?

好吧，这章或许该改名成“十万个为什么”。

不过这章内容确实是在学数据结构之前应该知道的内容，知道了它们就不会再害怕数据结构，也就能学得更轻松啦。

在讲抽象数据类型前先问问你哈，还记得数据抽象这回事吗？

在《脑洞大开——C 语言另类攻略》书中 C++ 那部分，咱们讲过一节内容叫“抽象的艺术”，当时说的是类的抽象，其实这种抽象是可以推广的。其实扩展说来，任何编程语言里的任意一种数据类型都是抽象出来的。

对于计算机而言，所有的数据到它那儿无非就两种形态：0 和 1，也就是所谓的二进制数据。但是咱们人类可忍不了啊，所以编程语言将二进制抽象成了咱们编程时需要的各种数据类型，然后在编译的时候根据该变量被定义的类型将二进制的机器语言编译成相应内容。

估计这段话已经让你懵掉了……拿 C 语言举个例子吧。

在 C 语言书中刚开头的时候我说过，给每个变量定义其数据类型是为了方便编译器知道应该用哪种编码对其进行编译，当时我还举了个例子——下面这个二进制数据：

01100111011011000110111101100010

如果不规定数据类型，编译器可以用 int、double 和 char 的编译指令分别翻译出 1 735 159 650、1.116 533 × 10^24 和 glob 三种结果。

这其实就是一个很鲜明的数据抽象的例子，明明是一样的二进制数据，为啥会出现不同的结果呢？因为 int、double 和 char 本身都是 C 语言抽象出来的，在你定义某个数据的类型时编译器就已经知道该以何种方式抽象它，并且会按它所属的类型进行分类存储和管理，以便提高效率和灵活应用，所以同样的二进制数据会因为被定义的类型不同而被抽象成不同的内容。

比方说 int a; 就是咱们人为地将 a 的内容的二进制数据抽象成了整型，它的内容就必须符合整型数据的规矩，即不能超出 int 的取值范围。

所以说啦，说白了数据本身没有差别，有差别的是被抽象的方式，二进制是一切信息的本质，咱们人为地把它抽象成各种类型的数据以便人类更好地运用它们罢了。

这里可能不太好理解，需要你好好品味一下。

说了这么多，那抽象数据类型又是啥呢？刚才说过啦，所有数据类型不管是基本的 int、char、double，还是复合型的函数结构体，都是抽象出来的。但是这就有问题啦，编程语言千千万，你咋就敢确定你在这种语言上写的实现方法在另一种语言上也能成功运行呢？换言之，你咋就敢确定同一段实现代码在各种语言的编译器下都能被正确抽象成它应有的类型呢？这个是不可能的啦，比方说 C++、JAVA 这些面向对象编程语言的代码在 C 的编译器下几乎都没法通过。因为 C 不知道啥是类，啥是继承和多态，所以当编译时它就无法正确抽象涉及这些知识的代码。别说是 C 了，即使是 JAVA 和 C++ 之间的代码都不见得能互用，毕竟各种语言之间都会有多多少少的语法和功能差异嘛。

但是这样写书的人就头疼啦，因为他不能也不可能把同一段代码用 N 种语言各实现一次，但是还要考虑到大家对编程语言各有所好的特点，不能以一概之。

哦，那就只写一种模型吧。这种模型不会用某种特定的语言将所有实现代码都一字不差地写出来，而是只说明一个实现该解决方案的大概逻辑。写这些比较模糊的大概逻辑的代码语言也不是特定的某种语言，而是将各种语言中的差异忽略后剩余的在各语言中规则基本相同的语法形成的“风格语言”。

这样写出的代码也就是所谓的伪代码了，而用这种代码实现的各种自定义数据类型就是传说中的抽象数据类型(ADT)。

说白了，抽象数据类型就是不受语言环境影响的描述某种问题的实现方法的一种模型，

通过理解这种模型的运行机理就可以在不同语言平台上写出对该问题的具体解决方案代码。

接下来我的实现代码将完全用 C 语言实现，并且会仔细解释其具体实现。通过理解 C 语言具体实现的方法，用其他语言实现也是分分钟哦，而且完全用 C 讲解实现代码可比看着伪代码听老师讲的时候抓狂却依然不知所云好理解得多哦。同时我会更加注重实用性，不会只讲那些应试的概念性内容，那样没意义，也远非编程的艺术所在。

扯远了……ADT 和其具体实现实际上反映了程序或软件设计的三层抽象。ADT 相当于是在概念层(或称为抽象层)上描述问题，即只是一个逻辑，而具体实现相当于是在实现层上描述问题，就是实打实地实现代码。其实还包含着第三层抽象，即在效率层上描述问题，就是说这个具体实现的效率如何。在第 2 章讲算法的时候会扯出叫算法时间复杂度和空间复杂度的东西，那个就是用来描述具体实现的效率的东东。嘿嘿，这个到时候再讲啦。

接下来要讲的就剩下逻辑结构和物理结构啦。那么现在就开始吧。

1.4 什么是逻辑结构?

来，继续我们的十万个为什么语文课嘿嘿，因为这章基本全是概念性问题嘛。不过也一样，如果不对这些概念有一定的了解，学数据结构会有点吃力的，所以还是尽快解决掉最好啦。

其实逻辑结构和物理结构在 1.1 节已经讲了一点啦。

逻辑结构说白了就是数据结构中各元素在逻辑上的相互关系，根据其相互关系的不同主要分为集合结构、线形结构、树形结构以及图形结构。

集合结构算是最松散的结构了吧……它和我们高中时学的集合一样，就是一些元素构成的一个整体。在这个整体里虽然各元素都属于这个整体，但是它们彼此之间没有任何关系。代表：森林(一种数据结构)，示意图如图 1-6 所示。

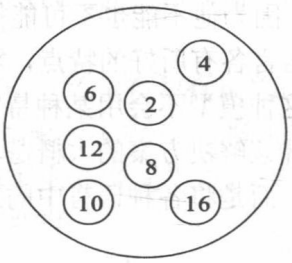

图 1-6

线形结构指的是各元素之间有联系，并且这种联系呈链性的结构。而且，这种链性是一对一的哦。代表：链表，示意图如图 1-7 所示。

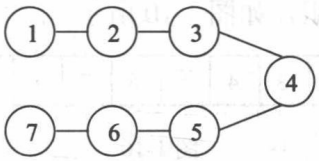

图 1-7

树形结构就是指像树那种数据结构一样，各元素间存在一对多以及层次关系的一种结构啦。代表：树，示意图如图 1-8 所示。

图形结构可能算是关系比较复杂的结构啦，它的各元素之间的关系是多对多的关系。代表：图，示意图如图 1-9 所示。

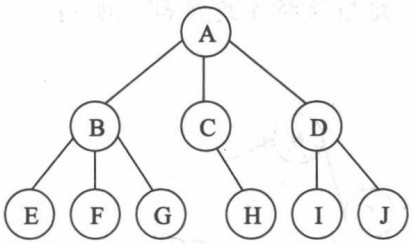

图 1-8

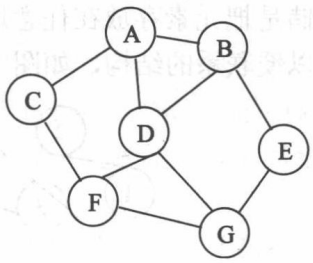

图 1-9

上面各图中的连线都是没有方向的，也就是说是双向的。如果在实际使用中要用到单向的时候，画示意图的时候需要将线换成单向箭头哦。

上面的每种结构都很常用，咱们要做的就是在合适的时候选择合适的结构去实现代码就好啦。

说完逻辑结构，再来看看物理结构吧。

1.5 什么是物理结构?

十万个为什么语文课的最后两节啦，嘿嘿。

那么什么是物理结构咧？刚才说过啦，逻辑结构是指数据结构中各元素之间的逻辑关系，而物理结构则是指这些元素在计算机里的存储的形式。也就是说，每种结构都兼有逻辑结构和物理结构，逻辑结构用来描述该结构中各元素在逻辑上的关系，物理结构用来描述该结构中各元素在计算机中的存储形式。物理结构和逻辑结构密不可分，如何存储各数据元素间的逻辑关系是实现物理结构的重点和难点。

物理结构只有两种：顺序存储和链式存储。

顺序存储就是通过从系统中申请足够长度的单元，然后将元素以连续地址的方式存放

的存储结构。其典型代表就是数组，如图 1-10 所示。

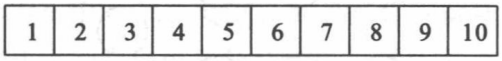

图 1-10

顺序存储虽好，但是有个致命弱点啊，就是你必须知道要存储数据的长度，至少要知道预计的最大长度，不然很容易出现空间不足或者剩余过多资源浪费的情况，而且它不太适合存储要经常变化位置或需经常插入或删除的内容，因为在这种结构中元素移位就意味着要移动大量元素。

所以，第二种存储方式就出现啦，它就是链式存储。

链式存储是把元素存放在任意地址单元里，地址连续不连续都无所谓，通过指针记录彼此的地址以便联系的结构，如图 1-11 所示。

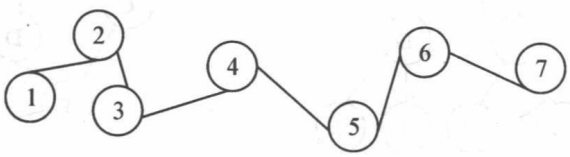

图 1-11

链式结构相比于顺序结构的优点在于它的存储更加灵活，不需要知道要存储的内容的长度，因为它可以随时申请，随时使用，而且在进行数据插入和删除时比顺序存储方便得多，只要删除对应结点然后将它前后两个结点彼此联系就行啦，如图 1-12 所示。

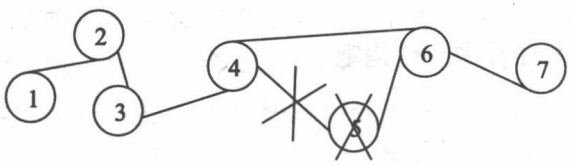

图 1-12

链式存储之间的联系可以是单向的，也可以是双向的，其典型代表就是单向链表和双向链表。

1.6 为什么会有这么多数据结构咧?

是啊，为啥会有这么多的数据结构呢？嘿嘿，其实这就和人类生产生活进化史息息相关了。

假设你现在有一座农场，养了一群马。一开始，这群马完全是散养的，没有给它们做