数据结构概述

定义

我们如何把现实中大量而复杂的问题以特定的数据类型和特定的存储结构保存到主存储器(内存)中，以及在此基础上为实现某个功能(比如查找某个元素，删除某个元素，对所有元素进行排序)而执行的相应操作， 这个相应的操作也叫算法

数据结构 = 个体的存储 + 个体的关系存储

算法 = 对存储数据的操作

再次讨论什么是数据结构

狭义:

数据结构是专门研究数据存储问题的课程

数据的存储包含两方面: 个体的存储 + 个体关系的存储

广义:

数据结构既包含数据的存储也包含数据的操作

对存储数据的操作就是算法

算法

解题的方法和步骤

衡量算法的标准

1. 时间复杂度

大概程序要执行的次数，而非执行的时间

2. 空间复杂度

算法执行过程中大概所占用的最大内存

3. 难易程度

4. 健壮性

再次讨论什么是算法

狭义

算法是和数据的存储方式密切相关

广义

算法和数据的存储方式无关

这就是泛型思想

数据结构的地位

数据结构是软件中最核心的课程

程序 = 数据的存储 + 数据的操作 + 可以被计算机执行的语言

预备知识

指针

指针的重要性:

指针是C语言的灵魂

定义

地址

地址就是内存单元的编号

从0开始的非负整数

范围: 0 -- FFFFFFFF【0-4G-1】

指针:

指针就是地址 地址就是指针

指针变量是存放内存单元地址的变量

指针的本质是一个操作受限的非负整数

分类:

1. 基本类型的指针

2 指针和数组的关系

结构体

为什么会出现结构体

为了表示一些复杂的数据，而普通的基本类型变量无法满足要求

什么叫结构体

结构体是用户根据实际需要自己定义的复合数据类型

如何使用结构体

两种方式:

```
struct Student st = {1000, "zhangsan", 20};
struct Student * pst = &st;
```

1. 通过结构体变量名来实现
   st.sid
2. 通过指向结构体变量的指针来实现【重点】
   pst->sid
   pst所指向的结构体变量中的sid这个成员

注意事项:

结构体变量不能加减乘除，但可以相互赋值

普通结构体变量和结构体指针变量作为函数传参的问题

动态内存的分配和释放

略

模块一: 线性结构【把所有的结点用一根直线穿起来】

连续存储[数组]

1. 什么叫数组

元素类型相同，大小相等

2. 数组的优缺点 — 与链表相比较:

优点

存取速度很快

缺点

事先必须知道数组的长度

插入删除元素很慢

空间通常是有限制的

需要大块连续的内存块

离散存储[链表]

定义:

n个节点离散分配

彼此通过指针相连

每个节点只有一个前驱节点，每个节点只有一个后续节点

首节点没有前驱节点 尾节点没有后续节点

专业术语:

首节点:

第一个有效节点

尾节点:

最后一个有效节点

头结点：
头结点的数据类型和首节点类型一样
是第一个有效节点之前的那个节点
头结点并不存放有效数据
加头结点的目的主要是为了方便对链表的操作
头指针：
指向头结点的指针变量
尾指针：
指向尾节点的指针变量

如果希望通过一个函数来对链表进行处理，我们至少需要接受链表的哪些参数
只需要一个参数：头指针
因为我们通过头指针可以推算出链表的其他所有参数

分类　：
单链表
双链表：
每一个节点有两个指针域

循环链表
能通过任何一个节点找到其他所有的结点
非循环链表

算法：
遍历
查找
清空
销毁
求长度
排序
删除节点
插入节点

算法：
狭义的算法是与数据的存数方式密切相关
广义的算法是与数据的存储方式无关
泛型：
利用某种技术达到的效果就是：不同的存数方式，执行的操作是一样的

链表的优缺点　--与数组相比较：
优点
空间没有限制
插入删除元素很快
缺点
存取速度很慢

线性结构的两种常见应用之一 栈
定义
一种可以实现"先进后出"的存储结构
栈类似于箱子

分类
静态栈
动态栈

算法
出栈

压栈

应用

函数调用
中断
表达式求值
内存分配
缓冲处理
迷宫

线性结构的两种常见应用之二 队列

定义:

一种可以实现“先进先出”的存储结构

分类:

链式队列 -- 用链表实现

静态队列 -- 用数组实现

静态队列通常都必须是循环队列

循环队列的讲解:

1. 静态队列为什么必须是循环队列
2. 循环队列需要几个参数来确定需要2个参数来确定
   两个参数:
   front
   rear
3. 循环队列各个参数的含义
   2个参数不同场合有不同的含义
   建议初学者先记住，然后慢慢体会
   1). 队列初始化
   font和rear的值都是零
   2). 队列非空
   font代表的是队列的第一个元素
   rear代表的是队列的最有一个有效元素的下一个元素
   3). 队列空
   font和rear的值相等，但不一定是零
4. 循环队列入队伪算法讲解
   两步完成:
   1. 将值存入r所代表的位置
   2. 错误的写法rear = rear + 1;
      正确的写法是: rear = (rear + 1) % 数组的长度
5. 循环队列出队伪算法讲解
   front = (front+1) % 数组的长度
6. 如何判断循环队列是否为空
   如果 front 与 rear的值相等，
   则该队列就一定为空
7. 如何判断循环队列是否已满
   预备知识:
   front的值可能比rear大，
   front的值也可能比rear小
   当然也可能两者相等

   两种方式:【通常使用第二种方式】
   1. 多增加一个标识参数
   2. 少用一个元素
      如果r和f的值紧挨着，则表明队列已满

用C语言伪算法表示就是:

```
if (  (r+1)%数组长度 ==  f  )
    已满
else
    不满
```

专题：递归

定义

函数自己直接或间接的调用它自己

递归满足三个条件

1、递归必须得有一个明确的中止条件
2、该函数所处理的数据规模必须在递减
3、这个转化必须是可解的

举例

1. 1+2+3+4+...100的和
2. 求介乘
3. 汉诺塔
4. 走迷宫

模块二：非线性结构

树

定义：

专业定义:【递归定义】

如果是一个非空树,则必须要满足两点：

1. 有且只有一个称为根的节点
2. 有若干个互不相交的子树，这些子树本身也是一棵树

通俗的定义【非递归定义】

1. 树由节点和边组成
2. 每一个节点有且只有一个父节点但可以有多个子节点
3. 但只有一个节点除外，该节点没有父节点，此节点称为根节点

一些专业术语：

节点
父节点
子节点
子孙
堂兄弟
深度
从根节点到最低层节点的层数称为深度
叶子节点：
没有子节点的节点
非终端节点
实际就是非叶子节点
度
子节点的个数称为度

分类

一般树

任意一个节点的子节点个数都不受限制的树

二叉树

任意一个节点的子节点个数最多只有两个，且子节点的位置不可更改的树

分类：

一般二叉树
满二叉树
在不增加树层数的前提下，无法再多添加一个节点的二叉树叫满二叉树

完全二叉树
如果只删除了满二叉树最低层最右边的连续的若干个节点，
这样的二叉树叫完全二叉树

森林
N个互不相交的树的集合

存储
二叉树的存储
顺序存储表示法【要求必须是完全二叉树】
链式存储
一般树的存储
双亲表示法
求父亲容易，求孩子难
孩子表示法
求孩子容易，求父亲难
双亲孩子表示法
求父亲孩子都容易
二叉树表示法【也叫孩子兄弟链表表示法】
先把一般树转化为二叉树，再存储二叉树
一般树转化为二叉树的方法是：
设法保证任意一个节点的
左指针域指向它的第一个孩子
右指针域指向它的下一个兄弟
只要能满足此条件，就可以把一个普通树转化为二叉树来存储
森林的存储
先把森林转化为二叉树，再存储二叉树

二叉树操作
遍历
先序遍历[先访问根节点]
先访问根节点
再先序访问左子树
再先序访问右子树

中序遍历[中间访问根节点]
中序遍历左子树
再访问根节点
再中序遍历右子树

后序遍历[最后访问根节点]
先中序遍历左子树
再中序遍历右子树
再访问根节点

已知两种遍历序列求原始二叉树
通过先序和中序 或者 中序和后序我们可以
还原出原始的二叉树
但是通过先序和后序是无法还原出原始的二叉树的

换种说法:
只有通过先序和中序， 或通过中序和后序
我们才可以唯一的确定一个二叉树
应用
树是数据库中数据组织一种重要形式
操作系统子父进程的关系本身就是一棵树
面向对象语言中类的继承关系本身就是一棵树
赫夫曼树

图

模块三：查找和排序
折半查找

排序：
冒泡
插入
选择
快速排序
归并排序

排序和查找的关系
排序是查找的前提
排序时重点

Java中容器和数据结构相关知识
Iterator接口
Map
哈希表

再次讨论什么是数据结构
数据结构研究是数据的存储和数据的操作的一门学问
数据的存储分为两部分：
个体的存储
个体关系的存储
从某个角度而言，数据的存储最核心的就是个体关系的存储
个体的存储可以忽略不计

再次讨论到底什么是泛型
同一种逻辑结构，无论该逻辑结构物理存储是什么样子的
我们可以对它执行相同的操作

The end
郝斌
2009年12月2日