LINUX-2019

学院	信息学院	专业班级	16 硬件二班
学号	161403209	姓名	关尧

实验目的：

1.了解 Linux 编程环境

2.掌握在 Linux 环境下编程的常用工具，例如：shell, vim, make, gedit, git 及各种语言的集成开发环境

3.了解 Linux 系统的内存、进程、线程、同步、通信的基本原理和其在实际程序实际中的应用

4.掌握 Linux 环境下的应用程序设计、开发和项目管理

5.通过 16 学时的实验，在 Linux 环境下设计并实现一个代码量不小于 2000 的项目

实验题目：

c 语言配合数据结构算法，实现：

1.堆排序算法

2.栈解析算术表达式

3.红黑树

4.b+树

编程语言和工具：

选择 c 语言作为编程语言，在 ubuntu18.04 下使用 vim 做文件的简单编写编译运行，后安装集成开发环境 VS Code 进行开发。

实验内容：

1. 堆排序

所谓堆，它是一个数组，也能够被看成一个近似的全然二叉树。树上每一个结点相应数组的一个元素。二叉堆分为二种：最大堆和最小堆

最大堆的特点：对于随意某个结点，该结点的值大于左孩子、右孩子的值，可是左右孩子的值没有要求。

堆排序算法

首先，按堆的定义将数组 R[0..n]调整为堆（这个过程称为创建初始堆），交换 R[0]和 R[n]；

然后，将 R[0..n-1]调整为堆，交换 R[0]和 R[n-1]；

如此反复，直到交换了 R[0]和 R[1]为止。

以上思想可归纳为两个操作：

（1）根据初始数组去构造初始堆（构建一个完全二叉树，保证所有的父结点都比它的孩子结点数值大）。

这里可以利用完全二叉树的结构，从最后一个非终端节点开始对子元素进行排序筛选。

（2）每次交换第一个和最后一个元素，输出最后一个元素（最大值），然后把剩下元素重新调整为大根堆

当输出完最后一个元素后，这个数组已经是按照从小到大的顺序排列了。

即每次调整都是从父节点、左孩子节点、右孩子节点三者中选择最大者跟父节点进行交换(交换之后可能造成被交换的孩子节点不满足堆的性质，因此每次交换之后要重新对被交换的孩子节点进行调整)。

有了初始堆之后就可以进行排序了。

2. 利用链栈实计算

栈有栈底和栈顶指针，元素是先进后出；对于栈的操作最主要的是创建、压栈、弹栈；在实现计算器时，我们输入的表达式叫做中缀表达式，我们需要将其转为后缀表达式；然后利用后缀表达式求取表达式的值；

创建了两个结构体；LinkList 是一个单向链表，保存数据和指向下一个节点的指针（next）；LinkStack 是栈，保存着栈顶指针和计数；

链栈是没有头结点的，将第一个节点数据域置位 NULL，计数器设为-1；表示栈底；

压栈与前面介绍链表的博文中增加链表节点的方式有差别；增加节点是把链表指针地址传递给子函数，这里是把指针传递给子函数；然后为节点申请空间，完成节点入栈；

节点入栈是：节点指针域指向当前的 top，然后 top 指向增加的节点（上移）；完成入栈；

pop 需要判断栈是否为空；完成数据赋值和计数减一，释放节点空间。

3. b+树

B+ 树是一种树数据结构，通常用于数据库和操作系统的文件系统中。B+ 树的特点是能够保持数据稳定有序，其插入与修改拥有较稳定的对数时间复杂度。B+ 树元素自底向上插入，这与二叉树恰好相反。

在 B+ 树中的节点通常被表示为一组有序的元素和子指针。如果此 B+树的序数（order）是 m ，则除了根之外的每个节点都包含最少 个元素最多 m-1 个元素，对于任意的节点有最多 m 个子指针。对于所有内部节点，子指针的数目总是比元素的数目多一个。因为所有叶子都在相同的高度上，节点通常不包含确定它们是叶子还是内部节点的方式。

每个内部节点的元素充当分开它的子树的分离值。例如，如果内部节点有三个子节点（或子树）则它必须有两个分离值或元素 a1 和 a2。在最左子树中所有的值都小于等于 a1，在中间子树中所有的值都在 a1 和 a2 之间((a1，a2])，而在最右子树中所有的值都大于 a2。

4. 红黑树

R-B Tree，全称是 Red-Black Tree，又称为“红黑树”，它一种特殊的二叉查找树。红黑树的每个节点上都有存储位表示节点的颜色，可以是红(Red)或黑(Black)。

红黑树的特性:

（1）每个节点或者是黑色，或者是红色。

（2）根节点是黑色。

（3）每个叶子节点（NIL）是黑色。 [注意：这里叶子节点，是指为空(NIL 或 NULL)的叶子节点！]

（4）如果一个节点是红色的，则它的子节点必须是黑色的。

（5）从一个节点到该节点的子孙节点的所有路径上包含相同数目的黑节点。

注意：

(01) 特性(3)中的叶子节点，是只为空(NIL 或 null)的节点。

(02) 特性(5)，确保没有一条路径会比其他路径长出俩倍。因而，红黑树是相对是接近平衡的二叉树。

红黑树的基本操作是添加、删除。在对红黑树进行添加或删除之后，都会用到旋转方法。为什么呢？道理很简单，添加或删除红黑树中的节点之后，红黑树就发生了变化，可能不满足红黑树的 5 条性质，也就不再是一颗红黑树了，而是一颗普通的树。而通过旋转，可以使这颗树重新成为红黑树。简单点说，旋转的目的是让树保持红黑树的特性。

旋转包括两种：左旋 和 右旋。

代码实现：

见具体代码。