ICT283评估练习2

目标：

要学习

- BST概念
- 要构建一个简单的二进制搜索树（BST）
- 展示递归与迭代的方法
- 任务准备 2.
- 必须提交练习b。它的评估方式与实验室4一样，是通过个人演示/防卫来进行的。

实验8的问题4需要先完成。

不落下这些练习是非常重要的。

练习b是通过个人示范/防御来评估的。

将练习b提交到LMS。在下周的实验课上，向你的导师演示练习b。这是对你工作的当面答辩。

练习

在这个实验中，使用**递归例程**来处理树的问题。参见PPT说明。教科书中使用了非递归（迭代）例程，但对于本实验，你需要编写递归例程。你可能想使用Cormen等人的单元参考书*《算法入门》*。

作业2可以使用迭代和/或递归程序。你将需要证明你的选择是正确的。

在作业中，通常会要求你为你的数据结构提供一个理由。https://www.youtube.com/watch?v=9Jry5-82I68&index=5&list=PLUl4u3cNGP61Oq3tWYp6V_F-5jb5L2iHb.在这个视频中，BST的理由是一个特别的例子。在视频中，树形算法被修改以迎合新的要求。这种方法是不可接受的--见《开放封闭原则》。想出一个更好的解决方案。除此以外，该视频很好地解释了BST及其使用。

a. （准备：在你开始练习b之前）

实现一个简单的（**非模板化的**）二进制搜索树（BST）类，名为***intBst***，它存储整数。提供以下递归操作：插入，删除树，搜索，inOrder 遍历，preOrder 遍历，postOrder 遍历。删除树是公开的还是私有的？必须提供对树在管理内存方面的正确操作至关重要的方法。你应该翻阅讲义（和课本上的想法）。

处理一个节点只需要打印出该节点的内容，在这种情况下就是存储在该节点中的整数。

数据应该来自一个包含整数的数据文件。你来决定格式。主程序应该打开数据文件，插入树中，并演示其他树的操作。

这个练习的重点是证明你对BST的理解。没有必要过分追求它，把通常在教科书和参考书《算法导论》中没有要求的操作放进去。

```
class intBst{... //intBST.h

   ///声明intBst类的方法和数据成员，并附上doxygen注释

};

// 实现可以放在intBST.cpp中，但为了方便起见，请保留。
```

// intBST.h中的其余代码带有正常（非脱氧）的代码注释

//整数的BST以后将被改为模板BST，这就是原因。

//因为现在所有的东西都在inBST.h中了--方便转换到

//模板BST。

为了在测试程序中使用你的BST，你将在主程序的例程main()中进行如下声明。

intBst intTree; //声明非模板化的整数树

// 其他代码遵循上述内容：

//插入各种整数值

// 测试树的方法

确保BST被正确测试。因此，这将涉及到通过值和引用将其传递给子程序。

b. **（下周在课堂上演示--在家里开始工作）。**

在尝试本练习的评估之前，请完成上面的练习*a*。递归操作是需要的。请不要过分追求，跳过上面的练习a。你将不会节省时间。

这个练习是为作业2做准备。这并不意味着下面的内容就是你的作业中要使用的二进制搜索树（BST）。本练习的目的是让你开始使用作业所需的二进制搜索树的基本模板。以后的实验将提供额外的修改。

将上面练习*a*中的BST转换成**模板二进制搜索树(BST)**。请注意，在这个评估的实验工作中，需要进行递归操作。

你可以在作业2中使用非递归版本，但你必须知道递归版本，以便你能证明迭代与递归算法的优点/缺点。理由必须来自使用本单元提供的数据的实际经验，所以要提供实际的比较时间和与使用的数据有关的解释。

本练习的*Bst.h*文件现在将有：

模板<class T> class

Bst{...。

///声明Bst类的方法和数据成员，并附上doxygen注释

};

// Bst.h中的其余代码带有正常（非脱氧）的代码注释

由于这是一个模板类，**没有*Bst.cpp***。

由于*Node*存储数据，*Node*也将作为一个模板来实现。

用练习a中的整数数据文件测试你的模板BST。将上面练习**a**中的测试程序（main）中的声明改为：：

Bst<int> intTree；

// 其余的代码与主程序中的练习a相同。

这是唯一的变化。如果需要进一步的改变，请注意从设计的角度和实施的角度评估这些改变。

一个读取子程序将读取并加载如下所示的*dateTree*数据结构。你需要的数据在实验8中提供的*数据*文件夹中。你的程序将读取*data/met_index.txt*中列出的所有文件。**实验8的第4个问题**必须完成才能做这个练习。测试你的模板BST，只插入日期值。

`Bst<Date> dateTree; // Date`是任务1中的日期类。确保你测试了`dateTree`上的所有

`BST`操作。

任务1中的Date类能否与BST一起工作，或者你需要修改Date类或提供额外的例程/方法？还需要什么，为什么？这些例程应该是Date类的方法还是Date命名空间的辅助例程？策略设计模式[1]，是否有用？解释一下你的理由。

创建一个单独的文本数据文件，*date.txt*，其中只有使用与任务1相同的日期格式的日期。使用这个数据文件来测试你的*dateTree*。作业的数据文件可能有按顺序增加的日期。这个新的数据文件，*date.txt*，将有不按任何顺序排列的日期。检查你的树是否工作。

提交时，Codeblocks中会有一个解决方案（工作空间）和3个项目。所有三个项目将使用相同的BST.h，其中包含你的**模板BST类**。你的解决方案中必须只有一份BST.h的副本。

第一个项目是整数树，*intTree*，第二个项目是使用Lab 8的数据文件的*dateTree*。第三个项目是使用*date.txt*的*dateTree*。

对于评估，你的导师通常会要求你展示/解释你使用BST的方法及其在*dateTree*数据结构中的使用细节（2nd和/或3rd 项目的解决方案）。你也可能被要求解释第一个使用*intTree*的项目。

在本实验中不作评估，但在以后的工作中包括作业2中作评估。

如果你的树形数据结构的用户（客户）想做其他事情，而不仅仅是打印数值，你要怎么做？请看教科书中二叉树一章的 "二叉树遍历和作为参数的函数 "一节。在你看完课本上的章节后，请看代码文件funcPtr.cpp，因为该代码是建立在课本上的。

递归的进一步练习

1. 斐波那契数列

斐波那契数是一系列的数字，定义如下所示。

- F(1) = 1 // 第一个数字

- F(2) = 2 // 第二个数字
- F(n) = F(n-1) + F(n-2) // 随后的数字

这些数字已经被用于数学和计算机科学，有些人认为这些数字代表了自然界的一些模式（https://en.wikipedia.org/wiki/Patterns_in_nature）。关于这些数字的快速背景和它们的应用，请参见https://en.wikipedia.org/wiki/Fibonacci_number。 本单元教材，像大多数计算机科学教材一样，在递归一章中涉及斐波那契数。该单元的参考书《算法导论》对算法和各种问题有很好的介绍，包括斐波那契的并行算法版本。

在这个练习中，需要两个版本的常规F。

该程序的一个版本是**迭代的**（使用循环）。你需要实现这个版本。

该程序的第二个版本是**递归的**。这个递归版本在文件*Timing Fibonacci.cpp*中提供给你。检查该文件中的代码。

[1]这个模式以前是必读的。

i. 比较每个版本的算法。哪个版本的算法最符合上面给出的斐波那契数的定义？审查其算法性能。

ii. 比较每个例程F的运行时间性能，当n很大时。在你的机器上计算F(100)需要多长时间？在实验室的机器上？

iii. 绘制输出n与所需时间的关系图。日志文件中会有运行的输出数据。将实验结果与每种算法的分析进行比较。

2. 梵天塔(又名河内塔)

爱德华-卢卡斯，据说是 "斐波那契数列 "一词的创造者，他在1883年提出了一个传说，并发明了一个相关的谜题。这个传说是这样的

"*在贝拿勒斯的大梵天庙里，在标志着世界中心的穹顶下的一块铜板上，有64个纯金盘，祭司们根据梵天不变的法则，在这些钻石针之间一次搬运一个盘：任何盘都不得放在一个较小的盘上。在世界之初，所有64个圆盘在一根针上形成了梵天之塔。然而，现在，塔从一根针转移到另一根针的过程正在进行中。当最后一个圆盘最终到位，再次形成梵天塔，但在不同的针上，那么世界末日就会到来，所有的人都会变成灰尘*。(转载自《*算法和数据结构》*中的一个练习*：*Mehlhorn和Sanders的《*基本工具箱》*[2]）。我们单位的教科书在递归一章中也有这个例子。这个难题引起了数学家，以及后来的计算机科学家的极大兴趣。因此，解决这一难题的算法被广泛研究。

在完成本单元的所有评估后，可以进行以下工作。

- 使用日常用品（纽扣、瓶盖.....等）制作谜题，如https://www.scientificamerican.com/article/the-tower-of-hanoi/，并研究出解题的算法。使用上面图例中描述的规则。
- 将你的算法与本单元课本（递归一章）中的工作实例进行比较。
- 用C++实现该算法，并尝试解开几个磁盘的谜题。
- 如果故事中的牧师有一台像你这样的计算机，并试图以模拟方式解决问题，那么将所有64个磁盘从一根针转移到另一根针上需要多长时间？
- 如果祭司们不得不以每秒一个盘子的速度手动移动每个盘子，而且不能有任何休息，那么他们要花多长时间？将这个时间与我们所知道的宇宙的年龄相比较。
- 对于这个问题的复杂性，你能用递归说什么？考虑一下当问题中只有3个磁盘的时候会发生什么

[2]这是一本好书。

，然后随着磁盘的数量增加到64个。它会是一个多项式时间问题吗?

如果你想在学期间休息时读一些 "轻松 "的书，可以试试Hinz, Klavzar, Milutinovi and Petr的《河内塔--神话与数学》。你可能会发现你以前学过的和以后要学的一些概念的起源。

[2]这是一本好书。