实现关键词索引表建立程序

（采用Visual Studio 2017 15.6.4 Windows SDK 10.0.16299.0）

这是这学期第三次上机作业了。感觉这次是真的没事给自己找了不少的麻烦。当时老师讲，用顺序表记录关键词索引，目的是为了加速查找。当时并不知道二分查找的方法，就觉得这样效率没有变好很多，于是就想了蛮久，想了一个o(1) 的方法，但实现起来感觉真的难太多了。

0、程序要完成的内容

完成对图书馆书目和关键词索引表的建立，提高索引表的访问效率。

1、最终程序的结构是这样的：

用户直接和一个命令行UI的类作交互，这个类调用一个管理书目信息单链表和关键词索引表的底层操作类，然后这个类再分别调用两个数据结构上定义的操作和类。

书目信息链表按编号顺序记录书目名称和相关信息，然后索引表因为老师专门强调了访问速度，就觉得只要是搜索，总还是慢了。所以就想到了这样的一种数据结构：

每个节点由一个指针数组和指向这个节点储存的数据的指针组成。指针数组的下标对应相应字符在现在所使用的编码中对应的unsigned char的数值，取值范围0-255。当访问一个字符串时只要一个字符一个字符地访问下一个字符对应的节点就可以找到字符串最终对应的节点上的数据。

虽然这看起来是一个一个ASCII码进行录入和存储的，但实际上这个程序只要文件和标准输入输出编码不变，完全可以做到存储非ASCII（比如中文）的书目。只要把字符串看作一个数据流，这里一层层访问的“字符”实际上只代表这一“位”上的字符，而MBCS的非ASCII字符完全可以当作多个字符存储。（标准的ASCII码只包含0-127的编码，非ASCII的字符会用多个128-255的字符当作一个字符来存储。UTF-8中，中文编码一般3个字节，GB2312中，中文一般两个字节）。

2、一些做法和想法来源

这样存储书目索引的方法的复杂度与关键词的数目无关。设有m个关键词，平均一个关键词有n个字符组成，那么算法的复杂度是。而如果使用顺序表二分查找，则复杂度一定大于，其中m为索引的最大数目（不是现有数目）。当程序需要管理的书目很多时，这个结构的优势就体现出来了。

虽然这个数据结构动态开数组时有很多空间存放的都是空指针，但是因为这样可以在实际应用（图书馆藏书上万）时大大提高效率，就没有再去追究这个问题。而且这个数据结构能存放的数据量只和操作系统能分配的内存大小有关，故可以存放的内容大大超过固定大小的顺序表。尤其是当索引很多时，不仅是访问，添加、减少条目的效率也远超顺序表。

因为程序要实际应用肯定要能保存现有的录入的状态，故就要有保存和加载文件的功能。对与顺序表，格式化输出还算简单。但如果要格式化输出一个这种“256叉”树，就比较麻烦了。现在我采用的方法是用连续的字符表示逐层深入的先根遍历，然后用制表符‘\t’（更好的选择可能是退格符‘\b’）表示回到上一级并已经完成之前的这一个字符的分支，下一个字节就是下一个存在分支的子女。这样一个字符一个字节地读出和写入。

比如“ARE”和“AT”在加入树并输出以后会变成这个线性的字符串：

“ARE\t\tT\t\t”

当然，完全可以利用书目链表的输出重新导入一遍关键词索引，但这会大大提高加载内容的时间和操作的量，增大对计算机来说的复杂度。这在书目繁多的时候是一个大的劣势。

然后对于存有数据的节点用一对大括号包围起来，中间格式化输出数据内容（在这次的实际应用中，对应书目编号串）。

然后可以用一般的方法格式化输出书目信息链表。这个比较正常，比较简单，就不说了。（大概就是每行输出一本书（就是用‘\n’分隔），然后每行按一定顺序输出内容）。

对于功能比较多的程序，一个合理的UI也是比较有用的。虽然目前还没有“--help”命令或者“man”命令，但是至少要有能处理简单命令，调用底层函数的功能。

然后同时有文件输入输出，标准输入输出的功能。然后就要分别对不同的命令调用不同的函数从而正确进行读出和写入。为此，用string类输入写了底层的一个条目的添加（删除），然后用认为命令是文件路径，看是否能打开命令对应的文件为判定，为真调用函数打开文件，逐行调用底层添加（删除），为假则调用底层，添加（删除）此行剩下的对应的一个条目。

对于和人交互的输出，肯定不能像前面那样输出一个树。为此写了一个函数进行遍历和输出。具体方法就是利用c++ 的字符串传递之前的访问路径的信息，先根遍历访问到达数据节点的时候输出这个字符串和节点的数据（按引用传递输出流，输出流包括文件输出流，字符串输出流，标准输出流等（c++ 继承）。。。可以按需要调用）。

因为要大量地动态分配内存空间，采用了使用析构函数对动态分配的内存进行释放的方式。这样的话，比如单链表，只要使用delete删除头节点对应的空间，delete就会自动调用这个节点的析构函数，“delete” 下一个节点，从而实现自动的递归删除。（如果一个节点，一个节点地删除，运算复杂度比利用函数栈要大一些）。这样递归调用删除的方法适用于所有逻辑关系不带有循环的单向的数据结构。

但这样的解决方式有一个问题。如果我用全局变量的这样一个类对局部变量的类进行赋值，那么局部变量在出它的作用域之后会调用它的析构函数，释放全局变量的空间。这肯定不是我们想要的，所以在实际的代码中要避免这样的情形出现尽量使用按引用或者指针传递数据。

3、一些可以改进的地方

这次的作业，底层的功能还是有不少并没有应用到最终的UI里面。这些功能当时是为了完整的达成这些结构需要的功能才写的。以后可以考虑把这些功能加入到程序中。（不过有一些可能的确更适合用来debug程序，比如查找一个节点在数据结构中的路径的函数（遍历+比较））。不过这些函数没有进行大量在实际运行中的debug ，故不一定很稳定。

还有就是发现对于树的数据结构，遍历操作是很常见的，但是因为访问操作对于不同功能会有所不同，并不能写成一个模板。我认为以后如果再遇到这种情况，可以考虑给遍历的函数传入一个信息（比如一个字符），然后根据这个字符（命令）调用不同函数进行访问操作。这应该可以大大减少总的代码量，并且可能可以方便debug 。只要一处遍历没错，就都不会有问题，比较符合模块化编程的做法。

4、总结

这次作业收获其实还是蛮大的，主要是对析构函数的理解和对模块化的实际经验。这次真的是花了不少精力，个人感觉最终效果还是不错的。

以后可以考虑更多地使用函数间传递功能选择的信息（这次只有一个底层的查找的函数因为找不找得到都要调用它，所以定义了一个这个函数的返回值的结构体）的方式来进行更灵活的编程，通过模块化减少代码量，提高效率。

虽然最终使用的数据结构和老师PPT上面的示例不同，但既然老师没有专门说要用什么，只说了索引不能用单链表，因为访问慢（这隐含了条目多的信息），所以我认为我用一个访问更快的数据结构应该是件好事。