实验报告

一、实验目标

本实验为了让大家联系一下刚刚学到的数据结构，感受一下 C++封装性的美好之处，于是让我们先自己实现并封装几个数据类型类，再对 HTML 文档进行解析。

二、实验环境

系统环境：Windows 10 64 位
IDE：Visual Studio 2015 Community

三、抽象数据结构说明

- Stack

功能

实现一个基本的栈（LIFO 表）

类说明

成员变量

stackTop	标识目前栈内元素数量
size	标识目前栈空间大小
data	栈内元素的头指针

成员函数

push	向栈内压入新的元素
pop	弹出当前栈顶的元素
top	返回当前栈顶元素的拷贝
empty	清空栈
isEmpty	判断当前栈是否为空

- CharString

功能

实现一个字符串类，元素采用宽字符存储

类说明

成员变量

length	标识目前字符串长度
data	数据域

成员函数

indexOf	使用 KMP 算法查找子串，重载以适应不同的类型的查找对象
getLength	返回当前字符串长度
getData	返回当前数据域指针
operator[]	重载[]运算符以允许通过下标访问字符串中的字符
substring	返回从 pos 到 end 中间的字符串
concat	将原串与参数串相连接，参数串在后
operator=	多重载=运算符以允许不同的赋值方式，但都必须是宽字符或采用宽字符的数据类型
operator==	多重载==运算符以允许不同的比较方式，但都必须是宽字符或采用宽字符的数据类型
operator+	多重载+运算符以允许不同的连接新串的方式，功能与 concat 相同，但都必须是宽字符或采用宽字符的数据类型
operator<<	重载<<运算符以允许采用流的方式输出

- CharStringLink

功能

实现一个字符串链表类，数据域采取双向循环链表的方式存储。

类说明

成员变量

length	标识目前字符串链表长度
data	数据域，数据采用结构体 CharStringNode 来存储，其中包括数据域(data)、前向指针(last)、后向指针(next)、标签名(tagName)、标签类名(className)、节点类型(NodeType)和标签类型(TagType)

成员函数

getData	返回数据域的指针
add	添加新的字符串，多重载以适应不同的输入类型
remove	删除数据字符串与参数相同的节点
search	查找数据字符串与参数相同的节点并返回指针
print	向特定的流中打印整个链表中的所有字符串，不含参数则向标准流打印

四、算法说明

网页解析算法

先将整个文档读入进一个字符串中，然后不断查找左右尖括号将文档拆散成普通文本节点和标签节点，对标签节点进行进一步分析得到标签类型和类名，并将所有节点按照遍历顺序建立链表。之后从链表的最后一项开始，向前遍历并入栈。发现包含关键信息（作者、标题、问题、正文）的标签节点时，开始出栈，将该过程中所有弹出的文本标签用一个字符串拼接保存，直到标签闭合为止。标签闭合后，再根据内容判断是否写入 info 文件，合适则写入。

分词算法

在分词前，先建立词典树，并记录每一个字是一个词的前缀还是结尾。之后，取出 info 结构体中存储的正文内容，从第一个字开始扫描，记录 start、end 和 tmpEnd。如果当前在 start 和 tmpEnd 之间的词语是一个词的前缀，那么就将 tmpEnd 后移，如果当前在 start 和 tmpEnd 之间的词语是一个词，则将 end 后移至 tmpEnd 的位置。如果当前 start 和 tmpEnd 之间的词语不在词典树中，则将 start 和 end 中的词语取出，建立或加入到分词结果字典树中，将 start 后移到 end+1 的位置，并将 end 和 tmpEnd 赋值为 start。继续直到文档扫描结束。

五、流程概述

【遍历并记录 input 文件夹下所有 html 格式的文件的文件名】->【加载词典】->【创建 output 文件夹】->【分析网页内容】->【分词】->【打印结果】

六、输入输出及操作相关说明

将待解析的网页文件存放在 input 文件夹内，将词典文件放在可执行程序所在目录下，执行即可。如果出现缺少库提示，则将 lib 文件夹内的所有文件拷贝到可执行程序所在目录下再执行即可。

七、实验结果

算法最终会将网页文件的四个主要内容提取出来并且保存在.info 文件内，将分词结果保存在.txt 文件内。.info 中的正文内容屏蔽了无意义提示语和作者名称后缀的所有非中文和英文字符（如果有的话），.txt 文件中的所有结果不保留单字结果，不过在程序代码中预留了宏开关，可以改为保留单字。基本符合预期。

八、功能亮点说明

算法优化提速

【字典树建立速度优化】

在建立字典树时，建立新的字符节点的时候，将该字节点作为其父节点的直接左孩子，将原先的左孩子变成该节点的右孩子，在数据规模庞大且在一定范围内有相同前缀的时候，大量节省查找的时间。

【分词算法速度优化】

我没有采用老师在 PPT 中给出的分词算法，因为该算法存在大量不必要的检索字典树的过程。时间复杂度为 $O(n^2)$，我采用了一个时间复杂度为 O(n)的算法。

以上两个算法将程序执行总时间从 20s+压缩到了 4s 左右，而且其中有一半耗费在建立词典树上。

采用宽字符存储

用宽字符解决中文在遍历时会被拆成两个的问题。

使用了宏开关

假装这是一个亮点……

九、实验体会

这个地方才是真正写是否符合预期的地方，对吧……

凑活，小型工程写多了发现自己 debug 速度越来越快了，见识的错误类型也越来越多了。

任务量倒还不算特别大，就是放在期中考试周附近，这个时间节点有点坑爹。

助教给的推荐实验报告的格式看上去不太好看。看了一下说明文档的格式，决定还是按照自己喜欢的排版和字体方式来。