**实验文档**

16307130015 王赢铭

**一，概述**

本次实验运用已学过的Cpp知识实现统计文档中英文单词次数，并按出现次数从小到大进行排列，经统计，本input.txt文档中有着超过170W以上的英文单词，其中重复出现次数超过10W的有the，高达102500+次之多，如附件中output.txt文件中所示。

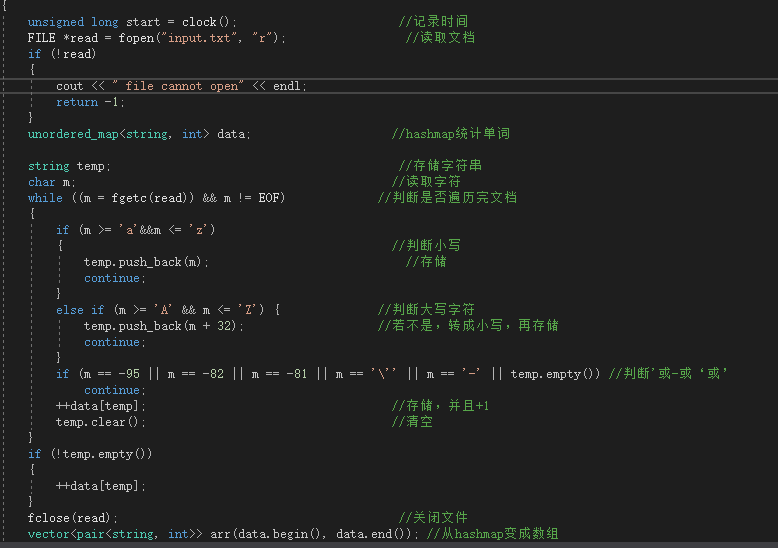
**二，实验步骤**

**本实验可以初步将问题分成两部分：**

**1.统计出现过的所有单词**

在遍历文档过程中主要运用的Hash\_map的算法，之前调试过用map遍历查找并且计数，但是效果并不尽如人意，整个过程耗时约半分钟，map<string, int> wordCount;

 然后改用哈希map，编码由Debug改成Release，整个程序耗时有明显的改进，缩短至1秒左右。



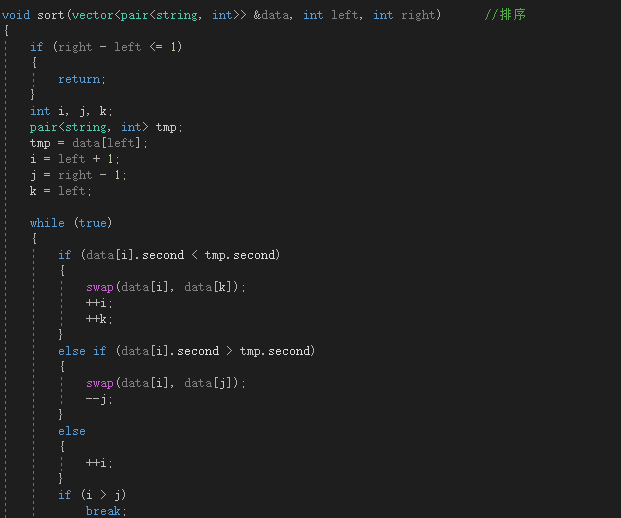
unordered\_map 比 map要快上很多，因为它使用哈希表（调用的时间是O(1)，map调用时间是O(nlogn)），效率得到较大改进但是代价就是它不是按顺序储存的。

Hash\_map基于hash table（哈希表）。 哈希表最大的优点，就是把数据的存储和查找消耗的时间大大降低，几乎可以看成是常数时间；而代价仅仅是消耗比较多的内存。然而在当前可利用内存越来越多的情况下，用空间换时间的做法是值得的。另外，编码比较容易也是它的特点之一。

虽然用法跟map非常相似，但在本实验中，hash\_map体现出了key-value效率的价值。

**2.按次数从小到大输出**

排序算法如图所示



算法则是运用Arrays的sort方法，以出现次数小到大为最优先排位方式，如果出现次数一致，则以辞典编纂从小到大的顺序排位：

**实验改进：**

**为了统计的高效性：**

对**源单词集合**和**无需统计的单词集合**根据字典序进行快速排序（平均O(nlogn)），再进行**顺序查找**（O(n)）以获得**相同单词**的重复次数，得到的词在无需统计的单词集合中查找（O(nlogn)），只有在无需统计的单词集合中不出现才能被统计如结果集，最后对结果集以出现次数为关键字进行降序排序（O(nlogn)），再输出结果，总复杂度为（O(nlogn)）。

在HashMap中，其会根据hash算法来计算key-value的存储位置并进行快速存取。

**另外根据实验要求，还对程序输出的美观性做了优化：**

（1）程序输出保持固定且较强适应性的输出格式，程序会自动过滤掉输入文件中允许任意次序穿插的符号、汉字、标点、空格，破折号等非英文字符，此外程序尽量降低了因某些特殊情况的发生造成格式混乱的可能。

（2）在当输入文件为空时，按照统一的格式输出相应异常提示，并结束程序。

（3）由于输出条件的控制造成输出为空时，也按照统一的格式输出相应异常提示，并结束程序。