**程序设计报告（3,5未写）**

姓名：吴文益 班级：软件193 学号：2019082323

课程设计题目：设计实现一个小型英汉双解词典

**1、问题描述：**

设计一个英汉双解电子词典，支持查找、插入、删除等功能。

**基本要求**

实现字典常用的数据结构包括有序表、AVL树、Patricia Tree（简称PAT tree，它是一棵压缩存储的二叉树结构）、散列表等，选一种数据结构，实现字典的基本操作，查找单词、插入单词（插入时，先查找，找不到则插入，找到则提示用户）、删除单词（删除时，先查找，找到则删除，找不到则提示用户）等。字典是按字母顺序排列的，不能用顺序查找，插入或删除单词后，要保持字典的有序性。

测试数据：任一英文单词。

**考核要求**

（1）选用两种以上的数据结构实现字典的查找、插入、删除等操作，并比较不同实现算法的时间复杂度和空间复杂度。

（2）如果采用线性结构且无序，成绩为不及格。

（3）选择合适的数据结构，达到基本要求，成绩为优秀。

（4）鼓励设计图形用户界面。

提示：字典可以自己建立，但必须按字母a~z建立26个文件，建议从网上下载，文件类型为txt。

**2、需求分析：**

软件的基本功能：从键盘输入英语单词，可以对字典进行增删查操作。

**3．概要设计：**

**（1）抽象数据类型**

ADT orderedlist hashmap trietree

Data

三种数据结构对应的数据关系

Operation

SearchWord

输入：要查找的英语单词

功能：查找英语单词并输出它的中文释义

输出：找到的中文释义

DeleteWord

输入：待删除的英语单词

功能：删除单词

输出：删除是否成功

InsertWord

输入：待添加的英语单词和中文释义

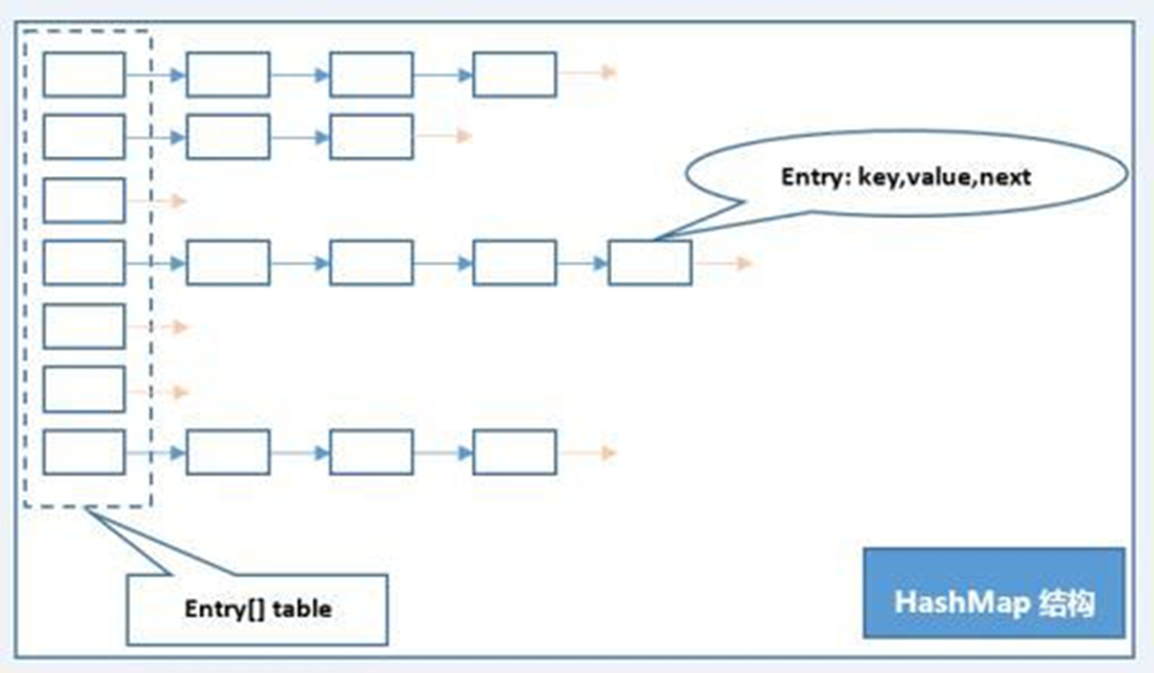
功能：添加单词

输出：插入是否成功

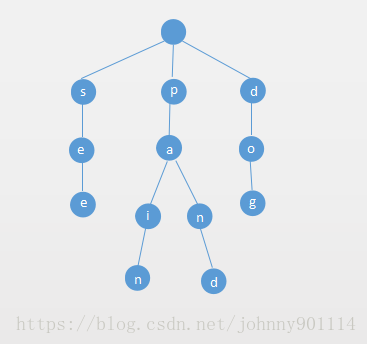
End ADT

**（2）部分数据结构图示**

Hashmap

****

Trietree



**（3）程序设计流程图**

插入

删除

查找

输入单词和中文释义

输入单词

输入单词

逻辑判断

逻辑判断

逻辑判断

反馈信息

反馈信息

输出中文释义

**（4）模块调用关系:**

三个数据结构对应的函数为searchWord，insertWord，deleteWord，主方法为main。

Main

Trietree

Hashmap

Orderedlist

对应的三个函数

对应的三个函数

对应的三个函数

**4、详细设计：**

查找、插入、删除操作分别采用三种不同的数据结构实现，界面采用QT框架写。

**（1）实现概要设计的数据类型**

**有序表**

*/\**

*\** *@auther:wwy*

*\** *@date:2020/12/14*

*\** *@introduction:OrderedList有序表,可以实现对字典的查找插入删除*

*\** *@email:773323518@qq.com*

*\*/*

*class* **OrderedList**

{

*private*:

*/\**

*\** *@introduction:存放单词数据*

*\*/*

ListNode data[2000];

*/\**

*\** *@introduction:存放文件路径*

*\*/*

QString filePath;

*/\**

*\** *@introduction:线性表长度*

*\*/*

int length;

*public*:

*/\**

*\** *@introduction:无参构造方法*

*\** *@pass:NULL*

*\** *@return:NULL*

*\*/*

**OrderedList**();

*/\**

*\** *@introduction:构造方法*

*\** *@pass:fp读取文件的路径*

*\** *@return:NULL*

*\*/*

**OrderedList**(*const* QString &fp);

*/\**

*\** *@introduction:读取文件数据到data数组*

*\** *@pass:fn读取文件的路径*

*\** *@return:NULL*

*\*/*

void **fileRead**(*const* QString &fp);

*/\**

*\** *@introduction:将data数组数据写入文件*

*\** *@pass:fn写入文件的路径*

*\** *@return:NULL*

*\*/*

void **fileWrite**(*const* QString &fp);

*/\**

*\** *@introduction:查找英语单词并输出它的中文释义*

*\** *@pass:ew要查找的英语单词*

*\** *@return:找到的中文释义*

*\*/*

QString **searchWord**(*const* QString &ew);

*/\**

*\** *@introduction:添加单词*

*\** *@pass:待添加的英语单词和中文释义*

*\** *@return:添加是否成功*

*\*/*

bool **insertWord**(*const* QString &ew, *const* QString &cm);

*/\**

*\** *@introduction:删除单词*

*\** *@pass:待删除的英语单词*

*\** *@return:删除是否成功*

*\*/*

bool **deleteWord**(*const* QString &ew);

*/\**

*\** *@introduction:返回线性表长度*

*\** *@pass:NULL*

*\** *@return:线性表长度*

*\*/*

int **getLength**()

{

*return* *this*->length;

}

*/\**

*\** *@introduction:返回当前对应文件路径*

*\** *@pass:NULL*

*\** *@return:返回当前文件对应路径*

*\*/*

QString **getFilePath**()

{

*return* *this*->filePath;

}

void **olDebug**(){

*for*(int i = 0; i < *this*->length; i++){

qDebug() << data[i].getEnglishWord()<<data[i].getChineseMeaning();

}

qDebug() <<"end";

}

*/\**

*\** *@introduction:交换两个ListNode的值*

*\** *@pass:l1,l2为待交换的数据*

*\** *@return:NULL*

*\*/*

*static* void **swapln**(ListNode &l1, ListNode &l2);

};

**散列表**

*/\**

*\** *@auther:wwy*

*\** *@date:2020/12/16*

*\** *@introduction:Hashlist哈希表(djb2),可以实现对字典的查找插入删除*

*\** *@email:773323518@qq.com*

*\*/*

*class* **HashMap**

{

*private*:

*/\**

*\** *@introduction:哈希表存储*

*\*/*

LinkedList map[510];

*/\**

*\** *@introduction:哈希表最大范围0~range,range<500*

*\*/*

int range;

*/\**

*\** *@introduction:当前表中单词数*

*\*/*

int length;

*/\**

*\** *@introduction:存放文件路径*

*\*/*

QString filePath;

*public*:

*/\**

*\** *@introduction:无参构造方法*

*\** *@pass:NULL*

*\** *@return:NULL*

*\*/*

**HashMap**();

*/\**

*\** *@introduction:构造方法*

*\** *@pass:fp文件路径,rg初始化范围*

*\** *@return:NULL*

*\*/*

**HashMap**(*const* QString &fp, int rg);

*/\**

*\** *@introduction:从文件读取数据*

*\** *@pass:fp文件路径*

*\** *@return:NULL*

*\*/*

void **fileRead**(*const* QString &fp);

*/\**

*\** *@introduction:将数据写入文件*

*\** *@pass:fn写入文件的路径*

*\** *@return:NULL*

*\*/*

void **fileWrite**(*const* QString &fp);

*/\**

*\** *@introduction:查找英语单词并输出它的中文释义*

*\** *@pass:ew要查找的英语单词*

*\** *@return:找到的中文释义*

*\*/*

QString **searchWord**(*const* QString &ew);

*/\**

*\** *@introduction:添加单词*

*\** *@pass:待添加的英语单词和中文释义*

*\** *@return:添加是否成功*

*\*/*

bool **insertWord**(*const* QString &ew, *const* QString &cm);

*/\**

*\** *@introduction:删除单词*

*\** *@pass:待删除的英语单词*

*\** *@return:删除是否成功*

*\*/*

bool **deleteWord**(*const* QString &ew);

*/\**

*\** *@introduction:djb2哈希函数*

*\** *@pass:ew为英语单词,mod为对hash值取余(防止溢出)*

*\** *@return:取余后的哈希值;*

*\*/*

void **hsDebug**()

{

*for*(int i = 0; i < *this*->range; i++)

{

*if* (map[i].getLength()>0)

{

qDebug() << i;

map[i].lsDebug();

}

}

qDebug() << "end";

}

*/\**

*\** *@introduction:返回当前对应文件路径*

*\** *@pass:NULL*

*\** *@return:返回当前文件对应路径*

*\*/*

QString **getFilePath**()

{

*return* *this*->filePath;

}

*/\**

*\** *@introduction:返回当前表单词数量*

*\** *@pass:NULL*

*\** *@return:单词数量*

*\*/*

int **getLength**()

{

*return* *this*->length;

}

int *static* **djb2**(*const* QString &ew,int mod);

};

**Trie树**

*/\**

*\** *@auther:wwy*

*\** *@date:2020/12/17*

*\** *@introduction:字典树数据结构*

*\** *@email:773323518@qq.com*

*\*/*

*class* **TrieTree**

{

*private*:

*/\**

*\** *@introduction:根结点*

*\*/*

TreeNode \*root;

*/\**

*\** *@introduction:文件路径*

*\*/*

QString filePath;

*/\**

*\** *@introduction:当前单词总数*

*\*/*

int length;

*public*:

*/\**

*\** *@introduction:无参构造方法*

*\** *@pass:NULL*

*\** *@return:NULL*

*\*/*

**TrieTree**();

*/\**

*\** *@introduction:构造方法*

*\** *@pass:fp读取文件的路径*

*\** *@return:NULL*

*\*/*

**TrieTree**(*const* QString &fp);

*/\**

*\** *@introduction:添加单词*

*\** *@pass:待添加的英语单词和中文释义*

*\** *@return:添加是否成功*

*\*/*

bool **insertWord**(*const* QString &ew, *const* QString &cm);

*/\**

*\** *@introduction:查找英语单词并输出它的中文释义*

*\** *@pass:ew要查找的英语单词*

*\** *@return:找到的中文释义*

*\*/*

QString **searchWord**(*const* QString &ew);

*/\**

*\** *@introduction:删除单词*

*\** *@pass:待删除的英语单词*

*\** *@return:删除是否成功*

*\*/*

bool **deleteWord**(*const* QString &ew);

*/\**

*\** *@introduction:从文件读取数据*

*\** *@pass:fp文件路径*

*\** *@return:NULL*

*\*/*

void **fileRead**(*const* QString &fp);

*/\**

*\** *@introduction:将数据写入文件*

*\** *@pass:fn写入文件的路径*

*\** *@return:NULL*

*\*/*

void **fileWrite**(*const* QString &fp);

*/\**

*\** *@introduction:返回当前对应文件路径*

*\** *@pass:NULL*

*\** *@return:返回当前文件对应路径*

*\*/*

QString **getFilePath**()

{

*return* *this*->filePath;

}

*/\**

*\** *@introduction:返回当前表单词数量*

*\** *@pass:NULL*

*\** *@return:单词数量*

*\*/*

int **getLength**()

{

*return* *this*->length;

}

};

**（2）数据结构算法效率分析与特点**

采用有序表，散列表，Trie三种数据结构，测试数据量为7982个单词

复杂度分析

**有序表**

介绍

采用顺序表存储，空间复杂度为O(n)

查找

对有序表查找采用二分查找，时间复杂度为O(logn)

插入

插入单词使有序表仍然保持有序，时间复杂度O(n)

删除

对顺序表删除，时间复杂度O(n)

**散列表**

介绍

采用djb2哈希函数进行数据分布，采用拉链法(双向链表)处理冲突，空间复杂度为O(n)

查找

没有冲突时，时间复杂度O(1)；产生冲突时，对双链表查找的时间复杂度为O(n)

插入

头插法，时间复杂度为O(1)

删除

没有冲突时，时间复杂度O(1)；产生冲突时，对双链表删除的时间复杂度为O(n)

**Trie树**

介绍

又称前缀树或字典树，是一种有序树，用于保存关联数组，其中的键通常是字符串。与二叉查找树不同，键不是直接保存在节点中，而是由节点在树中的位置决定。一个节点的所有子孙都有相同的前缀，也就是这个节点对应的字符串，而根节点对应空字符串。一般情况下，不是所有的节点都有对应的值，只有叶子节点和部分内部节点所对应的键才有相关的值。假设字符的种数有m个，有若干个长度为n的字符串构成了一个Trie树，最坏空间复杂度:O(m^n)

查找

时间复杂度为O(n)

插入

时间复杂度为O(n)

删除

先找到要删除的单词，然后修改数据域，时间复杂度为O(n)

**（3）相关函数介绍**

| **函数申明** | **参数** | **返回值** | **功能介绍** |
| --- | --- | --- | --- |
| bool insertWord(const QString &ew, const QString &cm); | 待添加的英语单词和中文释义 | 添加是否成功 | 插入单词 |
| bool deleteWord(const QString &ew); | 待删除的英语单词 | 删除是否成功 | 删除单词 |
| QString searchWord(const QString &ew); | 要查找的英语单词 | 找到的中文释义 | 查找英语单词并输出它的中文释义 |
| void fileWrite(const QString &fp); | 写入文件的路径 | NULL | 将数据写入文件 |
| void fileRead(const QString &fp); | 文件路径 | NULL | 从文件读取数据 |
| int getLength() | NULL | 单词数量 | 返回当前表单词数量 |
| int static djb2(const QString &ew,int mod); | 英语单词和取模值 | 字符串对应的哈希值 | djb2哈希函数 |
| static void swapln(ListNode &l1, ListNode &l2); | 两个节点 | NULL | 交换两个节点的值 |
| QStringList static handleData(QString data); | data传入的数据 | 数据处理后的列表 | 将数据分割 |
| int static trieIndex(QChar c); | data传入的字符 | 结点编号 | 将字符转换成TrieTree上的结点编号 |

**5、编码与调试分析：**

**编码与调试过程中遇到的问题及解决办法**：

1. **软件编码问题：qt读入文本编码不一致 解决办法：将字符串转为qt可识别的编码格式**
2. **Trie树不支持对含有特殊符号的英文单词扩展 解决办法：添加特殊符号表，尽量满足所有单词需求**

**6、使用说明：**

**进入界面后可以自主选择三个数据结构中的一个进行使用，也可以对三个数据结构进行比较和分析**

**7、测试结果：**

**有序表**

查找



添加



删除



**散列表**

查找



添加



删除



**Trie树**

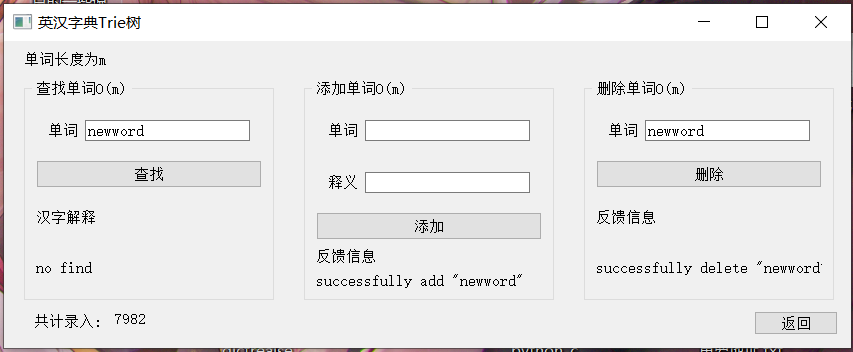
查找



添加



删除



**8、自学知识**

自学了c++的qt框架和字典树的相关知识。

**9、课程设计心得体会:**

数据结构课程设计让我们从底层开始一步一步完善自己的数据结构，从中我收获了很多，同时也加深了对各种数据结构的认识。通过对数据结构的应用、对各种算法的优化和对bug的修复，我感受到了编程的乐趣，同时也提升了编码的能力。