



中国科学技术大学  
University of Science and Technology of China

## 《程序设计 II》实验报告

设计题目： 小型英汉词典程序设计

姓 名： 王章瀚

学 号： PB18111697

专业班级： 18 级计算机系本科一班

2019 年 5 月

## 一、 开发环境与工具

本次程序设计使用的开发环境为 Windows10 操作系统，使用 C++语言，编译环境为 Visual Studio 2017。

## 二、 程序设计

### a) 设计原理

- ① 词条在内存中的存储。首先，由于一个词条所占内存至多就是 1KB，因此可以考虑将 dict.dat 文件中的 1000 余个词条一次性全部读入。其次，由于字典对于查找速度要求较高，应优先考虑查找速度快的数据结构，对于查找而言，使用哈希表等类似的数据结构速度将极快，但结合增删词条等功能要求以及 dict.dat 文件内词条本身的有序性方面的考虑，**决定使用数组来存储词条。**
- ② 词条的查找。由于初始情况下，词条已经按字典序排好。而即便添加词条，利用冒泡排序也可以在一次遍历内将修改了的词条排到相应位置，故词条的有序性及程序性能得以保障。在有序的条件下，可以使用**二分查找法**来查找词条。以本实验的 dict.dat 数据为例，共约 1000 个词条，粗略估计，至多 10 次二分查找的循环过程即可根据单词查找到相应词条。因此其效率较高。同时，为了进一步提高其速度，对单词做了**首字母索引**，可以有索引来快速找到单词将出现的范围。
- ③ 词条的添加，修改与删除。对于这三个功能，如果每次都去修改文件中间的内容，显然相对不合理。故可以考虑把需要添加的词条加入文件末尾，把需要删除的词条做标记，把需要修改的词条的原词条做删除标记，并在文件末尾加入新词条。并在用户需要的条件下，执行整理功能，删除旧词条，以避免文件中有过多不必要的词条。

### b) 程序功能模块描述及代码

（注：以下只展示函数的声明，详细代码可查看附件）

#### ① 主要结构体说明：

```

1. typedef struct entry {
2.     bool exist;        //是否被删除
3.     long flocation;    //在文件中位置
4.     char* word;        //单词
5.     char* explain;     //解释
6.     char* sentence;    //例句
7. } entry;
8.
9. typedef struct entryArray {
10.    entry** enArray;     //词条数组
11.    entry* lastInFile;   //文件中最后一个词条
12.    int startLetterIndex[26]; //记录单词首字母位置的索引
13.    int wordsNum;        //表示可用总词条数
14.    int fwordsNum;       //表示文件中总词条数（包括未删除的）
15.    int size;            //表示词条数组 enArray 的大小
16. } entryArray;

```

1. 结构体 entry: 用以描述一个词条;
2. 结构体 entryArray: 用以描述一个词条数组及其相关信息。具体内容已在代码中注释。

## ② 交互功能部分的函数

```

1. void listHelp();
2. void dealCommand(entryArray* entries, char* command);
3. void printEntry(entry* en);
4. void printAllEntries(entryArray* entries, int mode);

```

1. 函数 listHelp: 是用于输出命令列表, 相当于使用说明。
2. 函数 dealCommand: 是用于处理用户输入的命令字符串 command, 并作用于词条数组 entries
3. 函数 printEntry: 用于打印词条 en 的内容;
4. 函数 printAllEntries: 在 mode==0 时打印所有词条, 包括已删除的; 在 mode==1 时打印所有未删除的词条。

## ③ 排序功能部分的函数

```

1. int entryCompare(const entry* en1, const entry* en2);
2. void sortEntryArray(entryArray* entries);

```

1. 函数 entryCompare: 定义词条结构体 entry 的 compare 比较函数, 规则为: 已删除词条大于未删除词条, 在此基础上, 按字典序排序。
2. 函数 sortEntryArray: 对词条数组 entries->enArray 进行冒泡排序。

## ④ 主要功能实现部分的函数

1. `void readEntries(entryArray* entries);`
2. `entry* searchEntry(entryArray* entries, char* word);`
3. `void addEntry(entryArray* entries, entry* en);`
4. `void deleteEntry(entryArray* entries, char* word);`
5. `void editEntry(entryArray* entries, char* word, char* newData, int mode);`
6. `void clear(entryArray* entries);`

1. 函数 `readEntries`: 从文件 `dict.dat` 中读取或计算词条及其他相关信息, 并储存在词条数组 `entries` 中。
2. 函数 `searchEntry`: 对词条数组 `entries` 中的 `enArray` 进行二分查找, 从而返回单词 `word` 对应的词条结构体的地址。
3. 函数 `addEntry`: 将词条结构体 `en` 加入词条数组 `entries` 的末尾, 并在 `entries` 中的 `enArray` 空间不足时, 通过 `realloc` 函数增加数组空间, 一次增加 15 个 `entry*` 的空间。
4. 函数 `deleteEntry`: 将单词 `word` 对应的词条结构体中的 `exist` 设置为 `false`, 并将文件中该词条的开头标记更改为字符 ‘#’。
5. 函数 `editEntry`: 将单词 `word` 对应的词条结构体中的数据更新为 `newData`。当 `mode==0` 时表示修改单词; 当 `mode==1` 时表示修改解释; 当 `mode==2` 时表示修改例句。
6. 函数 `clear` 将词条数组 `entries` 中已删除的词条做彻底删除, 并且更新文件内容, 将更新了的词条内容重新按顺写入。

## 三、运行结果

以下是主要功能运行结果图示:

```
小型英汉词典命令介绍:
help          列出帮助列表
consult word  查询单词, (用需要查询的单词代替word)
add word_ex_sen  添加单词, (word: 操作单词; ex: 解释; sen: 例句)
del word       删除单词, (word: 操作单词)
edit word_item_data  编辑单词,
                    word: 操作单词;
                    item: 可以为w(单词), e(解释), s(例句);
                    data: 编辑的数据
clear          清理
cls           清屏
quit          退出
consult adamant 已查找到单词, 耗时0.000012秒!
单词:adamant
解释:kind of stone inflexible
例句:Eva was adamant that she would not come.
```

图1 查询单词结果

小型英汉词典命令介绍:	
help	列出帮助列表
consult word	查询单词, (用需要查询的单词代替word)
add word_ex_sen	添加单词, (word: 操作单词; ex: 解释; sen: 例句)
del word	删除单词, (word: 操作单词)
edit word_item_data	编辑单词,
	word: 操作单词;
	item: 可以为w(单词), e(解释), s(例句);
	data: 编辑的数据
clear	清理
cls	清屏
quit	退出
add hello_hi_hello!	, 关闭程序
增改词条成功	
consult hello	
已查找到单词, 耗时0.000013秒!	
单词:hello	
解释:hi	
例句:hello!	

图 2 添加单词结果

小型英汉词典命令介绍:	
help	列出帮助列表
consult word	查询单词, (用需要查询的单词代替word)
add word_ex_sen	添加单词, (word: 操作单词; ex: 解释; sen: 例句)
del word	删除单词, (word: 操作单词)
edit word_item_data	编辑单词,
	word: 操作单词;
	item: 可以为w(单词), e(解释), s(例句);
	data: 编辑的数据
clear	清理
cls	清屏
quit	退出
edit hello_s_hello, Jack!	, 关闭程序
增改词条成功	
consult hello	
已查找到单词, 耗时0.000010秒!	
单词:hello	
解释:hi	
例句:hello, Jack!	

图 3 编辑单词结果

```

1163  *_yarn_tale_story_fibers_for_knitting_ne
1164  #_hello_hi_hello!
1165  *_hello_hi_hello, Jack!
1166

```

图 4 添加和编辑单词后文件内容

```
1164 #_hello_hi_hello!
1165 #_hello_hi_hello, Jack!
1166
```




```
小型英汉词典命令介绍:
help          列出帮助列表
consult word  查询单词, (用需要查询的单词代替word)
add word_ex_sen 添加单词, (word: 操作单词; ex: 解释; sen: 例句)
del word      删除单词, (word: 操作单词)
edit word_item_data 编辑单词, (word: 操作单词; item: 可以为w(单词), e(解释), s(例句); data: 编辑的数据)

clear         清理
cls           清屏
quit         退出
del hello    删除单词hello成功!
```

图5 删除操作后文件内容及显示

```
1155 *_wean_to turn aw
1156 *_weigh_measure h
1157 *_welter_turmoil
1158 *_wend_to go proc
1159 *_whimsical_full
1160 *_wince_show bodi
1161 *_woo_try to win_
1162 *_writ_written or
1163 *_yarn_tale story
1164
```



```
小型英汉词典命令介绍:
help          列出帮助列表
consult word  查询单词, (用需要查询的单词代替word)
add word_ex_sen 添加单词, (word: 操作单词; ex: 解释; sen: 例句)
del word      删除单词, (word: 操作单词)
edit word_item_data 编辑单词, (word: 操作单词; item: 可以为w(单词), e(解释), s(例句); data: 编辑的数据)

clear         清理
cls           清屏
quit         退出
del hello    删除单词hello成功!
clear        完成清理!
```

图5 清理操作后文件内容及显示

## 四、 心得体会

本次实验过程中遇到了一些问题，并及时学习，吸取经验。例如：

①第一次使用 `realloc` 函数的时候，内存空间大小分配设置有问题，导致后面调试不断出错。

②为完成命令式的用户操作模式，需要对用户命令字符串有较好的处理，因此学习使用了 `string.h` 头文件中的 `strtok` 函数。

③经过本次实验，对文件操作有了更清楚的认识，并熟悉了 `malloc`, `calloc`, `realloc` 等内存分配函数。