C语言课程设计报告

**选题名称： 题目17：电子英汉词典**

**姓名： 罗戬浩 学号： 20170060220**

**院系： 数学与大数据学院**

**班级： 17数据科学与大数据技术专业2班**

**指导教师： 姜春涛、南小可**

**完成地点： 建模与仿真实验室**

**完成日期： 2018/07/06**

目录

[1题目、设计任务分析 4](#_Toc518660701)

[2编译环境及工具 6](#_Toc518660702)

[3新增功能及相关改进 7](#_Toc518660703)

[3.1改进思路 7](#_Toc518660704)

[4总体架构图以及各模块设计 9](#_Toc518660705)

[4.1总体架构图 9](#_Toc518660706)

[4.2各模块设计 10](#_Toc518660707)

[5核心函数介绍 15](#_Toc518660708)

[6主要用到的数据结构及操作 22](#_Toc518660709)

[6.1根据2中思路确定关键数据的数据类型及传输形式 22](#_Toc518660710)

[7主题流程图 24](#_Toc518660711)

[8测试情况 25](#_Toc518660712)

[9小结 37](#_Toc518660713)

[9.1设计中遇到的问题 37](#_Toc518660714)

[9.2问题的解决方法 37](#_Toc518660715)

[9.3设计体会 37](#_Toc518660716)

[10参考文献 39](#_Toc518660717)

[11部分源代码 40](#_Toc518660718)

电子英汉词典

罗戬浩 佛山科学技术学院 2017级数据科学与大数据技术2班

**摘要：**当今社会英语是一门必不可少的语言，英语一直是国家注重的学科之一，但是厚重昂贵的英汉词典无疑给需要学习英语的人带来生理和财产上的压力。而电子英汉词典系统却完美地避开了这两个毛病，还有的是，电子英汉词典系统操作简单，还可以根据个人的需求增删词汇，电子英汉词典系统是真正意义上的实现简单快捷学习英语单词的功能。本程序设计具有词条信息的输入、搜索、编辑、删除、保存、读取和加密等功能，操作界面简洁美观，易于操作。程序用了条件语句、循环结构、指针、结构体、链表等知识点。软件运用菜单实现交互式管理，用户轻松的按键既可实现对软件的操作，操作简单易懂，功能丰富，可以很好的满足的需要。通过对电子英汉词典的设计，进一步理解和掌握C语言这门课程的知识点，能够熟练地调用函数，把C语言的语句串联起来，提高自己C语言程序设计的能力，为今后解决实际问题打下良好基础。

**关键词：**电子英汉词典；条件语句；循环结构；指针；结构体；链表；

# 1题目、设计任务分析

|  |  |
| --- | --- |
| 设计题目 | 题目17：电子英汉词典 |
| 题目设计要求 | 实现简单电子英汉词典的功能，具体管理操作包括单词的添加、显示、查找、删除、修改和保存等。  功能要求：  1、采用结构体数组，每个数据的结构应当包括：单词的英文拼写，单词的中文释义。  2、系统功能：  a. 词条录入：即添加单词记录。  b. 信息显示：将所有的单词按字母顺序显示。  c. 词条修改：对已经输入的单词信息进行修改。  d. 词条删除：删除某个单词记录。  e. 单词查询: 输入单词英文拼写,输出该单词的中文释义。  f. 信息保存：将单词信息保存到文件。  g. 退出系统  3、系统使用说明：执行一个具体的功能之后，程序将重新显示功能菜单。系统的功能并不限于上述，可以对其进行扩充完善，如在对信息进行修改和删除时，可以考虑系统的安全性，在执行前若输入正确密码，才可进行操作。  测试数据： 要求被选用的词条有30个左右，简单单词为主。 |
| 题目设计任务分析 | 1、采用结构体数组。  2、（1）词条录入；（添加词条）  （2）信息显示；（显示词条）  （3）词条修改；（编辑词条）  （4）词条删除；(删除词条)  （5）单词查询；（搜索词条）  （6）信息保存；（保存词条）  （7）退出系统。（退出字典）  （备注：要求实现的功能；（程序内命名）） |

# 2编译环境及工具

|  |  |
| --- | --- |
| 编译环境及工具 | （1）编译环境：   1. ubuntu16.04   2、windows 7  （2）编译工具：  1、vim   1. visual studio 2010 |

# 3新增功能及相关改进

|  |  |
| --- | --- |
| 新增功能及相关改进 | 1、相关改进：  （1）采用链表。  （考虑到设计电子英汉词典经常会插入和删除元素，而链表在这些方面比结构体数组更具有优势，因此选用链表来设计电子英汉词典系统。）  （2）细分查询词条功能，将其细分为（英译汉、汉译英）两个功能。  2、新增功能：   1. 读取词条；   （可以实现读取已保存的词条内容，免去每次都重新添加词条的麻烦。）   1. 功能加密；   （对编辑词条和删除词条两个功能进行加密，以增强程序的安全性。）   1. 模糊搜索；   （输入英文单词的几个关键字符就可搜索出字典中存在的英文单词。）   1. 按词性分类；   （系统可以按照单词的词性，如（名词n、动词v……）把词条筛选分类出来，并能统计每种词性对应的词条的个数。）  3、相关细节添加：  （1）用户在输入密码的时候屏幕上会显示”\*”号，以免密码泄露。  （2）统计使用者使用字典的总时长。 |

## 3.1改进思路

1、根据题意构建大体框架，题目要求运用结构体数组，而考虑到设计电子英汉词典经常会插入和删除元素，而链表在这些方面比结构体数组更具有优势，因此选用链表来设计电子英汉词典系统。

2、题目要求实现包括退出系统内的七个功能，依次如下（1）词条录入、（2）信息显示、（3）词条修改、（4）词条删除、（5）单词查询、（6）信息保存、（7）退出系统。而考虑到电子英汉词典系统是为学习英语词汇的人而设计的理念，于是在此基础上添加（8）读取词条、（9）功能加密、（10）模糊搜索、（11）按词性分类词条等功能，并且针对题目要求的功能（5）单词查询进行改进，将其分为（英译汉和汉译英）两个子功能。

3、界面友好是一个系统必不可少的特性，为了实现界面友好，尽可能地将系统的输入输出界面规范化，菜单的显示也用统一的类型来展示。

## 4总体架构图以及各模块设计

## 4.1总体架构图

系统大体由主函数 main(int argc, char \*argv[])和一系列的功能函数组成，可实现添加词条、精确搜索词条（英译汉、汉译英）、模糊搜索词条、按词性分类词条、编辑词条、删除词条、显示词条、保存词条、读取词条、功能加密和退出字典等功能。每次使用都会调用主函数。主函数主要是利用switch语句对数据进行处理，流程图如下图4.1所示。

AddContent(struct listNode \*list)

添加词条

case

英译汉

SearchContent(struct listNode \*list)

精确搜索词条

汉译英

case

case

FuzzySearch(struct listNode \*list)

模糊搜索词条

case

wordFormation(struct listNode \*list)

按词性分类词条

EditContent(struct listNode \*list)

编辑词条

密码正确

case

密码错误

case

main(int argc, char \*argv[])

主函数

DeleteContent(struct listNode \*list)

删除词条

密码正确

密码错误

case

switch

DisplayAll(struct listNode \*list)

显示词条

case

SaveContent(struct listNode \*list)

保存词条

case

case

Goodbye()

退出字典

case

ReadContent(struct listNode \*list)

读取词条

## 4.2各模块设计

**1、AddContent(struct listNode \*list) 添加词条**

apple 苹果 n

apple 苹果 n

run 跑 v

easily 容易地 adv

funny 有趣的 adj

InsertList(list,add)

按顺序插入链表

输入

funny 有趣的 adj

easily 容易地 adv

run 跑 v

……

……

系统提示用户输入英文单词、输入该英文单词的解释、输入改英文单词的词性，接着这些信息保存在字典的词条中，这些词条经过调用InsertList(list,add)函数，把词条按顺序插入链表。

**2、SearchContent(struct listNode \*list) 精确搜索词条**

apple 苹果 n

apple

英译汉

输入

apple 苹果 n

苹果

汉译英

系统根据用户的选择进行英译汉和汉译英操作，若选择英译汉操作系统将提示用户输入英文单词，然后若存在该单词则显示该单词所在的词条，若不存在则显示不存在该词条；若用户选择汉译英操作系统将提示用户输入中文，然后若存在该中文则显示该中文所在的词条，若不存在则显示不存在该词条。

**3、FuzzySearch(struct listNode \*list) 模糊搜索词条**

apple 苹果 n

app

输入

系统提示用户输入英文单词的关键词，若存在开头为所输入的关键词的英文单词则显示该词条，否则显示不存在该词条。

**4、wordFormation(struct listNode \*list) 按词性分类词条**

数量：1

apple 苹果 n

名词n

数量：1

run 跑 v

动词v

输入

数量：1

funny 有趣的 adj

形容词adj

数量：1

easily 容易地 adv

副词adv

系统提示用户选择按何种词性分类词条，词性分别有名词n、动词v、形容词adj、副词adv，根据用户的选择筛选并显示出字典中所有该词性的词条，最后进行统计所有该词性词条的数量。

**5、EditContent(struct listNode \*list) 编辑词条**

banana 香蕉 n

修改

存在

apple

输入单词

正确

不存在

加密

重新输入密码

不正确

系统将提示用户输入密码，若密码正确则提示用户输入英文单词，若该英文单词存在则提示用户要把改英文单词修改成什么单词并输入其中文解释和词性，否则提示该此条不存在，若密码错误则提示用户密码错误无法进入编辑词条界面。

**6、DeleteContent(struct listNode \*list) 删除词条**

删除

存在

apple

输入单词

正确

不存在

加密

重新输入密码

不正确

系统将提示用户输入密码，若密码正确则提示用户输入英文单词，若该英文单词存在则删除该单词所在的词条，否则提示该此条不存在，若密码错误则提示用户密码错误无法进入删除词条界面。

**7、DisplayAll(struct listNode \*list) 显示词条**

apple 苹果 n

apple 苹果 n

run 跑 v

easily 容易地 adv

funny 有趣的 adj

按顺序输出

funny 有趣的 adj

easily 容易地 adv

run 跑 v

冒泡排序法

……

……

系统会显示字典中所有的词条内容。

**8、SaveContent(struct listNode \*list) 保存词条**

apple 苹果 n

easily 容易地 adv

apple 苹果 n

easily 容易地 adv

funny 有趣的 adj

run 跑 v

Dictionary.txt

位于与cpp同目录下

funny 有趣的 adj

run 跑 v

……

系统会把字典中的所有词条内容保存在同cpp目录下的dictionary.txt中，若保存成功系统将提示用户保存成功。

**9、ReadContent(struct listNode \*list) 读取词条**

apple 苹果 n

easily 容易地 adv

apple 苹果 n

easily 容易地 adv

funny 有趣的 adj

run 跑 v

funny 有趣的 adj

Dictionary.txt

位于与cpp同目录下

run 跑 v

系统会把同cpp目录下的dictionary.txt的内容读取到字典当中，若读取成功系统将会提示用户保存成功。

1. **Goodbye() 退出字典**

（结束时间-开始时间）以秒为单位

退出

系统将会显示出结束标语，并统计用户使用字典的时间。

# 5核心函数介绍

**1、核心代码（1）：判断词条是否已经存在**

struct listNode \*ExistOrNot(struct listNode \*list,char \*inputWord)

{

//创建链表结构体辅助词条数据的判断是否存在的功能

struct listNode \*listNext;

//把辅助链表结构体初始化为list链表头的下一个结点

listNext=list->next;

while(listNext!=NULL)

{

//比较inputWord与(listNext->data).word是否相同，即判断该单词是否存在于字典内

if(strcmp((listNext->data).word,inputWord)==0)

{

//返回满足条件的链表结构体

return listNext;

}

//继续遍历直到listNext=NULL

listNext=listNext->next;

}

//若该单词不存在字典内就返回NULL

return NULL;

}

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| list | 信息传递链表 | struct |
| inputWord | 所添加的英文单词 | char \* |
| listNext | 辅助链表 | struct |
| next | 链表的下一个节点 | struct \* |

**2、核心代码（2）：把新链表按首字母字母表升序顺序插入**

void InsertList(struct listNode \*list,struct listNode \*insert)

{

//创建链表结构体辅助词条数据的按顺序插入的功能

struct listNode \*listHead;

//把辅助链表结构体初始化为list链表头

listHead=list;

//插入词条（前一个词条单词比该词条单词的ASCII小，后一个词条单词比该词条单词的ASCII大）

while(listHead->next!=NULL&&strcmp((listHead->next->data).word,(insert->data).word)<0)

{

//辅助链表结构体指向其下一个结点

listHead=listHead->next;

}

//insert链表结构体的下一个结点指向辅助链表结构体的下一个结点

insert->next=listHead->next;

//辅助链表结构体的下一个结点指向insert链表结构体

listHead->next=insert;

}

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| list | 信息传递链表 | struct |
| insert | 需要插入的链表 | struct |
| listHead | 辅助链表 | struct |
| next | 链表的下一个节点 | struct \* |

1. **核心代码（3）：释放已创建链表的内存**

void FreeList(struct listNode \*list)

{

//创建链表结构体辅助词条数据的清理链表结构体的功能

struct listNode \*listHead;

//把辅助链表结构体初始化为list链表头

listHead=list;

//逐个释放链表空间

while(listHead->next!=NULL)

{

//辅助链表结构体指向其下一个结点

listHead=listHead->next;

//释放list链表结构体

free(list);

//list链表结构体指向辅助链表结构体

list=listHead;

}

//释放辅助链表结构体

free(listHead);

}

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| list | 信息传递链表 | struct |
| listHead | 辅助链表 | struct |
| next | 链表的下一个节点 | struct \* |

1. **核心代码（4）：保存字典词条内容**

void SaveContent(struct listNode \*list)

{

//创建链表结构体辅助词条数据的保存的功能

struct listNode \*listNext;

//把辅助链表结构体初始化为list链表头的下一个结点

listNext=list->next;

//自定义saveFile以写入的方式打开dictionary.txt

FILE \*saveFile = fopen("dictionary.txt","w");

//当saveFile创建不成功，即不存在dictionary.txt文件

if(saveFile==NULL)

{

printf("不存在dictionary.txt文件\n");

return;

}

else

{

//保存表头

//fprintf(saveFile,"%10s %10s %10s\n","英文单词","中文解释","单词词性");

//保存数据

while(listNext!=NULL)

{

//按格式吧词条数据写入dictionary.txt文件

fprintf(saveFile,"%10s %10s %10s\n",(listNext->data).word,(listNext->data).mean,(listNext->data).format);

//继续遍历直到listNext=NULL

listNext=listNext->next;

}

//提示所有词条已保存在当前目录的dictionary.txt中

printf("所有词条已保存在当前目录的dictionary.txt中");

//关闭saveFile

fclose(saveFile);

}

}

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| list | 信息传递链表 | struct |
| listNext | 辅助链表 | struct |
| saveFile | 打开要的保存文件 | FILE |
| next | 链表的下一个节点 | struct \* |

1. **核心代码（5）：读取文件词条内容**

void ReadContent(struct listNode \*list)

{

//创建链表结构体辅助词条数据的读取的功能

struct listNode \*add;

//释放已创建链表

FreeList(list);

//重新创建链表结构体

list=(struct listNode \*)malloc(sizeof(struct listNode));

if(list!=NULL)

{

//把list的下一个结点指向NULL

list->next=NULL;

}

//定义一个整形作为统计词条数量

int num;

//初始化统计数据为0

num=0;

//自定义readFile以读取的方式打开dictionary.txt

FILE \*readFile = fopen("dictionary.txt","r");

//当readFile创建不成功，即不存在dictionary.txt文件

if(readFile==NULL)

{

printf("不存在dictionary.txt文件\n");

return;

}

else

{

//当读取文件不为文件结束符

while(!feof(readFile))

{

//创建链表结构体辅助数据的读取功能

add=(struct listNode \*)malloc(sizeof(struct listNode));

if(add!=NULL)

{

//num统计文件所传值到add的数据个数

num=fscanf(readFile,"%10s %10s %10s\n",(add->data).word,(add->data).mean,(add->data).format);

//当传值到add的数据个数小于3则跳出循环

if(num<3)

{

break;

}

//英文单词首字母按字母表的顺序把add的数据插入到list中

InsertList(list,add);

}

}

//打印读取成功

printf("dictionary.txt中的所有词条已保存在字典中");

}

}

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| list | 信息传递链表 | struct |
| add | 辅助链表 | struct |
| readFile | 打开要的保存文件 | FILE |
| num | 每次读取信息的数量 | int |
| next | 链表的下一个节点 | struct \* |

1. **核心代码（6）：功能加密**

int Password()

{

//设置原始密码

char password[]="123456";

//设置密码标志

int flag;

//定义一个字符数组作为用户输入的密码

char getWord[WORDNUM+1];

//为实现输入密码是显示"\*"号的功能，定义一个整形接收每一个输入的ASCII值，定义一个整形作为用户输入密码的长度

int oneWord;

int len;

//初始化密码标志为0

flag=0;

//每一个输入的ASCII值初始化为0

oneWord=0;

//用户输入密码的长度初始化为0

len=0;

//要求用户输入密码

printf("请输入密码：");

getchar();

//ASCII中13为回车键，即当输入的字符不为回车键

while ((oneWord = getch()) != 13)

{

//把每一个输入的字符ASCII值转化为字符类型，并利用len的自增将密码保存在getWord中

getWord[len++] = oneWord;

//每一个输入用“\*”号代替

putc('\*', stdout);

}

//把用户输入密码的最后一位赋值为0

getWord[len] = 0;

printf("\n");

//判断密码是否正确

if(strcmp(getWord,password)==0)

{

//若密码正确则密码标识为1

flag=1;

}

else

{

//提示密码不正确

printf("输入密码错误\n");

flag=0;

}

//返回flag的值为对应函数利用

return flag;

}

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| password | 管理员密码 | char[] |
| flag | 密码标识 | int |
| getWord | 获取用户输入密码 | char[] |
| oneWord | 用户输入密码的每一个字符串的ASCII值 | int |
| len | 用户输入密码的长度 | int |

# 6主要用到的数据结构及操作

## 6.1根据2中思路确定关键数据的数据类型及传输形式

1、图标形式展示：

图6.1链表及结构体符合图

2、部分代码形式展示：

//数据内容

struct content

{

//设置英文长度

char word[WORDNUM+1];

//设置中文长度

char mean[MEANNUM+1];

//设置单词词性长度

char format[WORDNUM+1];

};

//链表

struct listNode

{

//数据域，存放数据

struct content data;

//指向下一个链表节点

struct listNode \*next;

};

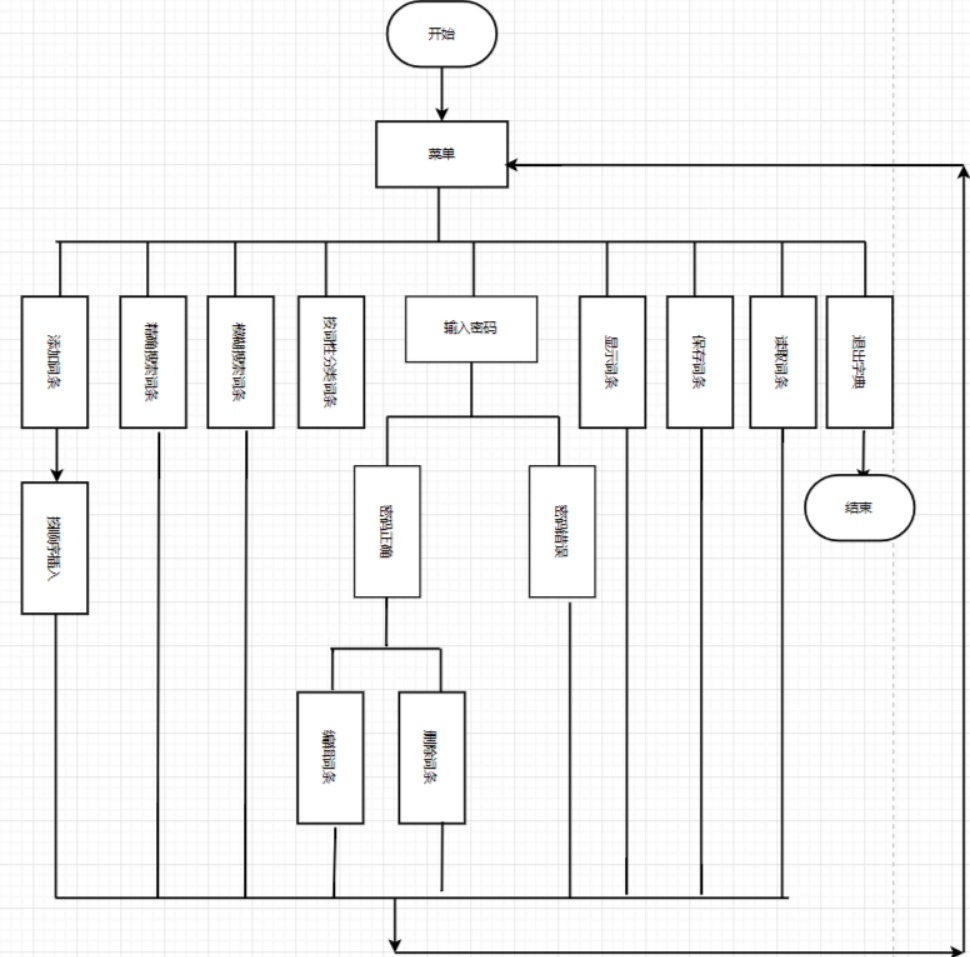
（1）链表字段详细（传递功能）

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| data | 链表数据域 | struct |
| next | 链表指针域 | struct \* |

（2）结构体字段详细（词条内容）

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段说明 | 字段类型 |
| word | 英文单词 | char(31) |
| mean | 中文解释 | char(31) |
| format | 单词词性 | char(31) |

# 7主题流程图



# 8测试情况

1、由于系统基础框架在vim上搭建，而附加功能在visual studio 2010上构建，经过多次调试，gcc编译器执行预处理, 编译, 汇编, 链接等工序系统执行成功，如下是main(int argc, char \*argv[])函数里面的DisplayMenu()函数，负责显示主界面的工作。如下图8.1

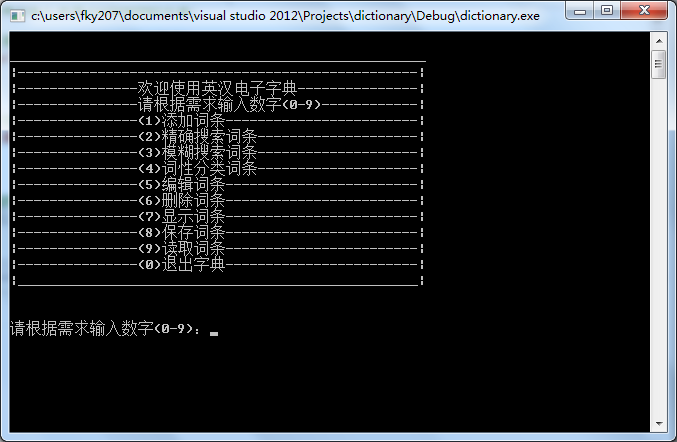


图8.1主界面

用户在允许的输入范围内输入不同的数字可实现不同的功能。

2、用户若在主界面中输入数字【1】，则可实现添加词条的功能。如图8.2为添加词条界面

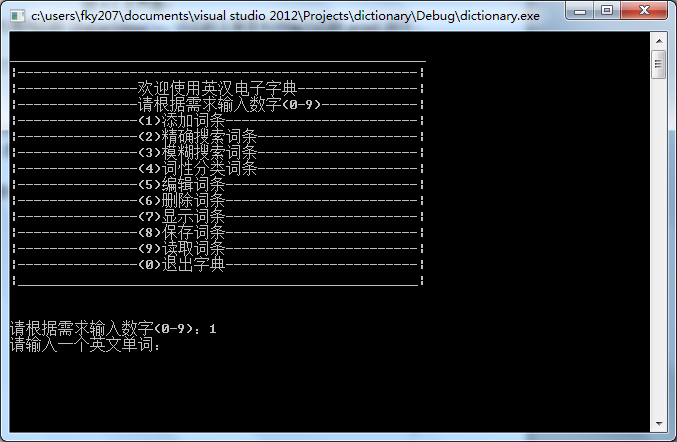


图8.2添加词条界面

系统会提示使用者输入英文单词、中文解释以及单词的词性，从而把词条内容录入字典实现添加词条的功能。

3、用户若在主界面中输入数字【2】，则可实现精确搜索词条的功能。如图8.3为精确搜索词条界面

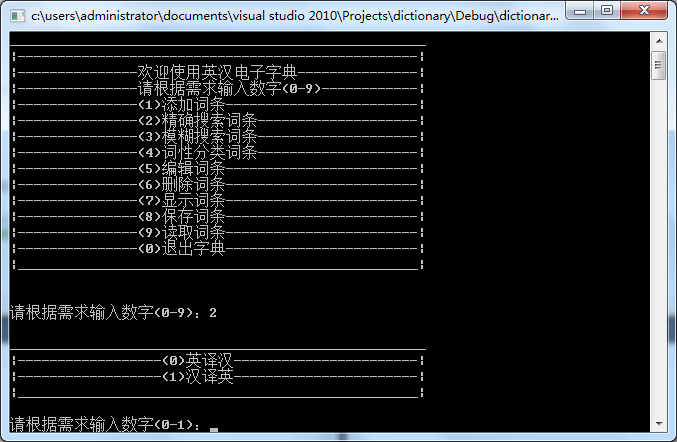


图5.3精确搜索词条

系统界面会提示用户选择何种方式（英译汉、汉译英）进行搜索，该功能也可以考验使用者对单词的掌握程度。

4、用户若在主界面中输入数字【3】，则可实现模糊搜索词条的功能。如图8.4为模糊搜索词条界面

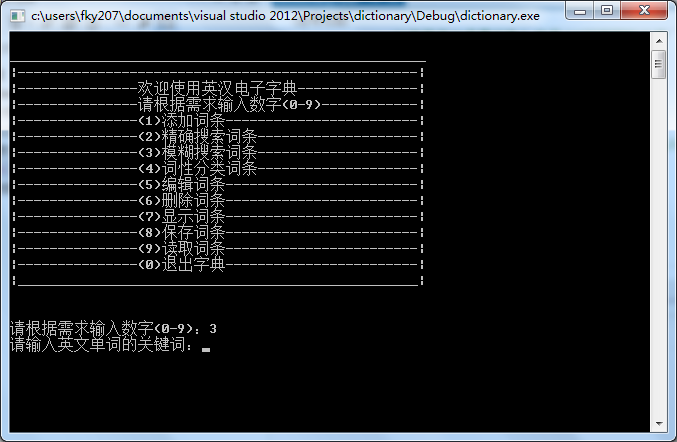


图8.4模糊搜索词条

用户可以输入关键词模糊搜索词条，例如字典中有“apple 苹果 n”词条，用户输入app即可模糊搜索出“apple 苹果 n”该词条。该功能可以为不是特别熟悉英文单词的使用者带来便利。

5、用户若在主界面中输入数字【4】，则可实现词性分类词条的功能。如图8.5为词性分类词条界面

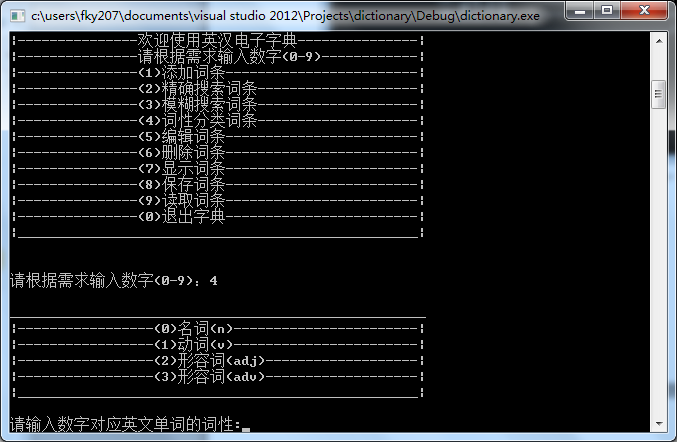


图8.5词性分类词条

下系统界面会提示用户按何种词性把字典中的词条进行分类（名词n、动词v、形容词adj、副词adv），然后系统会统计某种词性的英文单词有多少个。该功能实现了词条的分类统计的功能。

6、用户若在主界面中输入数字【5】，则可实现编辑词条的功能。如图8.6为编辑词条界面

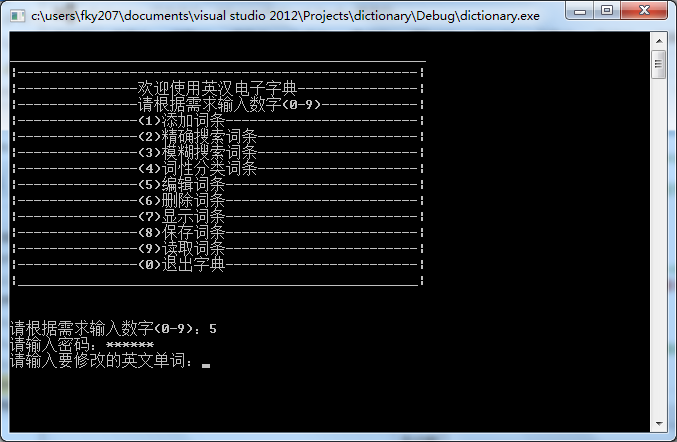


图8.6编辑词条

为了提高系统的安全性，系统界面提示用户首先要输入密码，当用户输入密码时系统将用“\*”号把密码掩盖，以防密码泄露，只有密码正确的情况下才能执行编辑词条的功能。该功能为系统管理员提供，系统管理员有系统的密码可以执行修改词条数据的功能。

7、用户若在主界面中输入数字【6】，则可实现删除词条的功能。如图8.7为删除词条界面

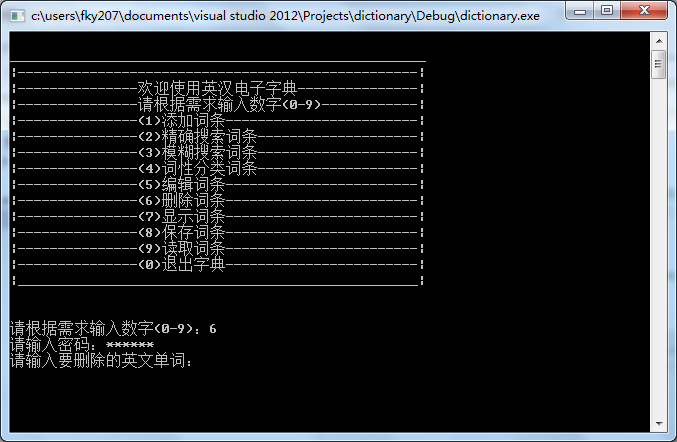
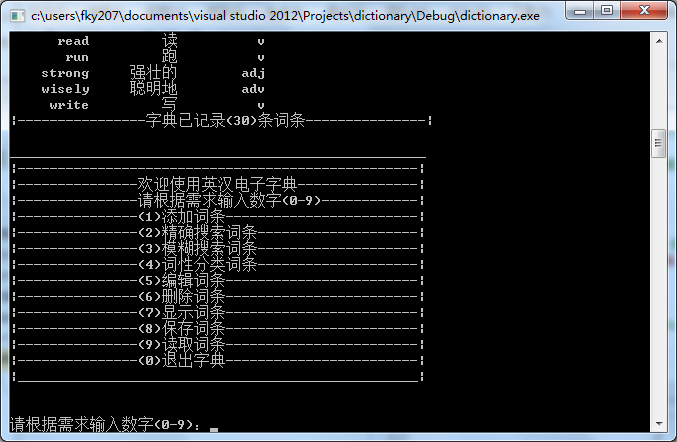


图8.7删除词条

为了提高系统的安全性，系统界面提示用户首先要输入密码，当用户输入密码时系统将用“\*”号把密码掩盖，以防密码泄露，只有密码正确的情况下才能执行删除词条的功能。该功能为系统管理员提供，系统管理员有系统的密码可以执行删除词条数据的功能。

8、用户若在主界面中输入数字【7】，则可实现显示词条的功能。如图8.8为显示词条界面

图8.8显示词条

系统界面会显示所有储存在字典中的词条，并且还有词条数量统计的功能。该功能可以为系统使用者提供英文单词预习的功能。

9、用户若在主界面中输入数字【8】，则可实现保存词条的功能。如图8.9为保存词条界面，图8.10为词条数据保存在与cpp同目录下的dictionary.txt文件视图。

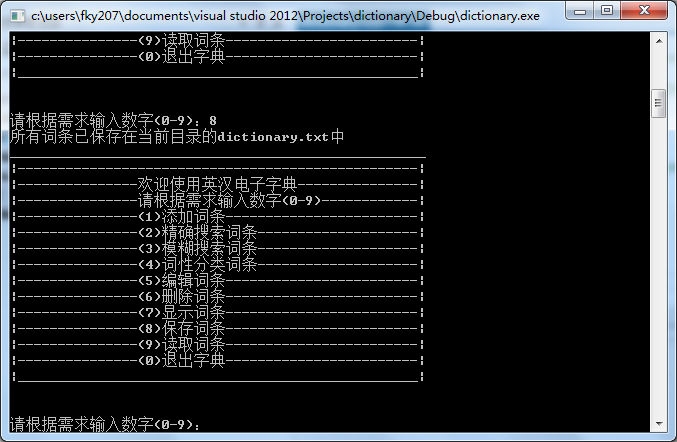


图8.9保存词条

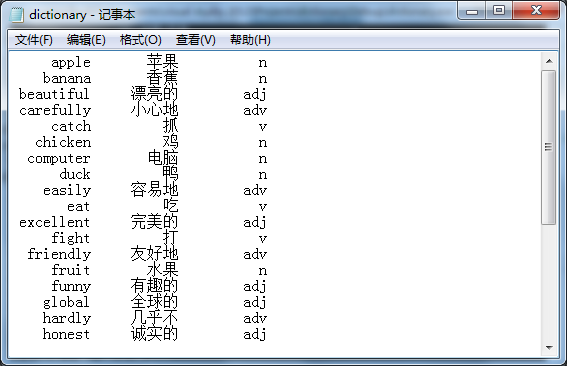


图8.10 dictionary.txt视图

若保存成功系统将会提示使用者单词保存成功。该功能可以为使用者复习英文单词所用。

10、用户若在主界面中输入数字【9】，则可实现读取词条的功能。如图8.11为读取词条界面

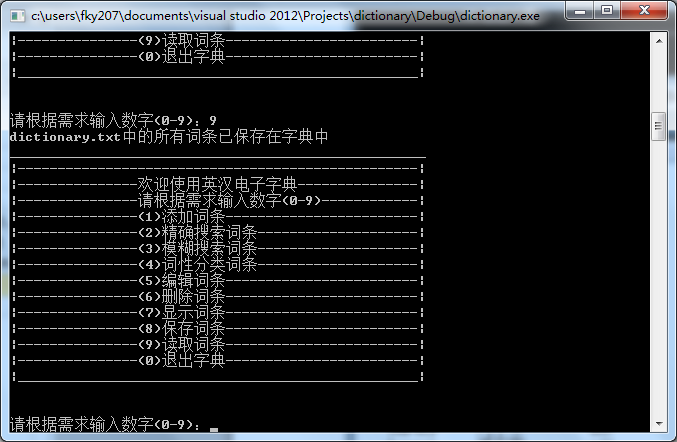


图8.11 读取词条

若读取成功，系统将提示使用者读取成功。该功能可以把dictionary.txt内的词条数据读取进字典当中，从而为使用者可以利用读取的单词实现字典的十个功能。

11、用户若在主界面中输入数字【0】，则可实现退出字典的功能。如图8.12为退出字典界面

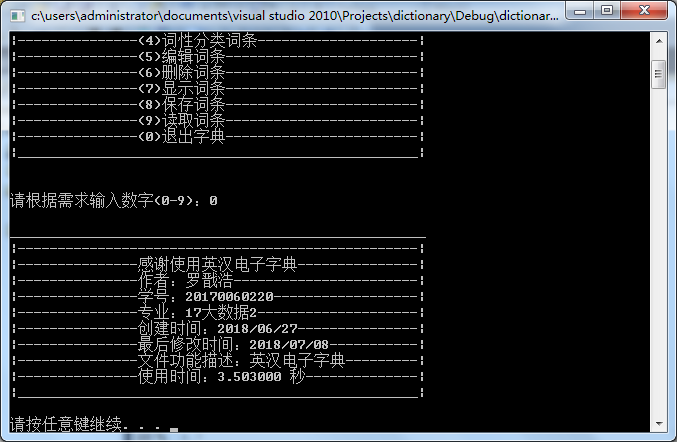


图8.12 退出字典

系统将提示用户感谢使用系统，系统会统计使用者使用字典的时间，从而检验使用者学习单词的效率。

**9小结**

9.1**设计中遇到的问题**

在设计电子英汉词典系统时遇到的问题主要有如下几个：（1）因系统的基本架构在ubuntu系统下的vim进行编译的，而新增功能和相关细节修改则在学校机房windows 7系统下的visual studio 2010进行编译，编译环境和编译工具的不同导致设计电子英汉词典系统出现不同的报错；（2）设计把dictionary.txt文件内容读取进电子英汉词典的功能时遇到字典内容乱码的现象。

9.2**问题的解决方法**

问题（1）的解决办法如下：在ubuntu系统下的vim设计好电子英汉词典系统的基础上，把剩下的设计工作全部改用在学校机房windows 7系统下的visual studio 2010下进行，统一编程环境。

问题（2）的解决办法如下：在读取dictionary.txt时运用到fscanf()函数（fscanf(FILE\*stream, constchar\*format, [argument...])，FILE \*stream：文件指针，char \*format：格式字符串，[argument...]：输入列表），经过加断点调试后发现是“char \*format：格式字符串”这个问题，随后就把所有的写入写出功能统一改为"%10s %10s %10s\n"，并且在此基础上去除表头的录入dictionary.txt（以防读取文件是把表头当作词条内容从而出现编译错误），最终问题得以解决。

9.3**设计体会**

学习了C语言程序设计这门课程之后，使我对程序设计有了一定的了解，为了熟练掌握C语言的语法特点、及其数据结构，提高自身编写程序的能力。通过课程设计实现理论知识与实际情况的结合，使所学知识能够在现实社会中起到一定的作用，防止所编写的程序脱离实际，让程序可以更好的发挥作用，以便减轻实际工作中所遇到的繁琐步骤，提高现实工作中的效率。此外通过课程设计提高逻辑思考能力和解决实际问题的能力。程序设计中，语句的语法和常量变量的定义都有严格的要求，有时输了一个中文标点、打错了一个字母，编译就不通过，程序无法正常运行。因此，程序设计初学阶段，学生经常会犯这样的错误，可能要通过几次乃至十多次的反复修改、调试，才能成功，但这种现象会随着学习的深入而慢慢改观。这当中就有一个严谨治学、一丝不苟的科学精神的培养，又有一个不怕失败、百折不挠品格的锻炼。电子英汉词典是生活中不可缺少的部分编写了一个电子英汉词典是十分必要的。通过程序设计，达到理论与实际应用相结合，帮助学习英语单词，能够根据实际应用，初步实现系统功能模块及算法的设计，通过编程基本实现电子英汉词典。

# 10参考文献

1. CSDN参考文献：<https://blog.csdn.net/snow_wu/article/details/53172721>
2. C语言库函数速查手册：<https://www.w3cschool.cn/cyykhsscsc/>
3. CSDN参考文献：<https://blog.csdn.net/hubaoquanu/article/details/53576219>

# 11部分源代码

/\*-----------------------------------------------------------------------

\*作者：罗戬浩

\*学号：20170060220

\*专业：17大数据2

\*创建时间：2018/06/27

\*最后修改时间：2018/07/08

\*文件功能描述：英汉电子字典

------------------------------------------------------------------------\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include <time.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

//getch()函数要用到头文件conio.h

//\_CRT\_SECURE\_NO\_WARNINGS:预编时处理,忽略安全检测

#define WORDNUM 30

#define MEANNUM 30

//数据内容

struct content

{

//设置英文长度

char word[WORDNUM+1];

//设置中文长度

char mean[MEANNUM+1];

//设置单词词性长度

char format[WORDNUM+1];

};

//链表

struct listNode

{

//数据域，存放数据

struct content data;

//指向下一个链表节点

struct listNode \*next;

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：函数声明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void AddContent(struct listNode \*list);

void SearchContent(struct listNode \*list);

void FuzzySearch(struct listNode \*list);

void wordFormation(struct listNode \*list);

void wordFormation1(struct listNode \*list,int choose);

void EditContent(struct listNode \*list);

void DeleteContent(struct listNode \*list);

void DisplayAll(struct listNode \*list);

struct listNode \*ExistOrNot(struct listNode \*list,char \*inputWord);

void InsertList(struct listNode \*list,struct listNode \*insert);

void FreeList(struct listNode \*list);

void DisplayHead();

void DisplayOne(struct listNode \*display);

void DisplayMenu();

void Goodbye(double start,double finish);

void SaveContent(struct listNode \*list);

void ReadContent(struct listNode \*list);

int Password();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：AddContent

\*添加词条内容

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void AddContent(struct listNode \*list)

{

//创建词条结构体来接受输入的词条信息

struct content input;

//创建链表结构体判断所添加的单词是否已经存在

struct listNode \*judge;

//创建链表结构体辅助词条数据按顺序插入list链表

struct listNode \*add;

//清除读写缓冲区

fflush(stdin);

//利用词条结构体输入词条数据

printf("请输入一个英文单词：");

scanf("%s",&input.word);

printf("请输入这个英文单词的解释：");

scanf("%s", &input.mean);

printf("请输入这个英文单词的词性：");

scanf("%s", &input.format);

//判断添加的单词是否已经存在（若存在则显示该词条，若不存在则添加该词条）

if((judge=ExistOrNot(list,input.word))==NULL)

{

//新建链表结构体add

add=(struct listNode \*)malloc(sizeof(struct listNode));

if(add != NULL)

{

//把词条内容复制到add链表结构体当中

strcpy((add->data).word,input.word);

strcpy((add->data).mean,input.mean);

strcpy((add->data).format,input.format);

//把add中的英文单词首字母按字母表顺序插入list链表中

InsertList(list,add);

}

}

else

{

//若该英文单词已经存在则显示“该词条已经存在”，并将该词条显示出来

printf("该词条已存在\n");

//打印表头

DisplayHead();

//打印一个该词条

DisplayOne(judge);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：EditContent

\*修改所选词条内容

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EditContent(struct listNode \*list)

{

//创建词条结构体接受新的词条数据

struct content edit;

//创建链表结构体判断字典是否存在该英文单词

struct listNode \*judge;

//定义一个字符数组接受用户输入的英文单词

char editWord[WORDNUM+1];

//清除读写缓冲区

fflush(stdin);

//输入要修改的词条的英文单词

printf("请输入要修改的英文单词：");

scanf("%s", &editWord);

//判断添加的单词是否已经存在（若存在则修改该词条）

if((judge=ExistOrNot(list,editWord))!=NULL)

{

//清除读写缓冲区

fflush(stdin);

//输入新的要修改的词条内容，并用词条结构体接收词条内容

printf("请输入要修改成的英文单词：");

scanf("%s", &edit.word);

printf("请输入要修改成的英文单词的中文解释：");

scanf("%s", &edit.mean);

printf("请输入要修改成的英文单词的词性：");

scanf("%s", &edit.format);

//利用链表结构体judge复制内容

strcpy((judge->data).word,edit.word);

strcpy((judge->data).mean,edit.mean);

strcpy((judge->data).format,edit.format);

}

else

{

//若不存在则显示“不存在该词条”

printf("不存在该词条\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名：SearchContent

\* 精确查找词条内容

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SearchContent(struct listNode \*list)

{

//创建链表结构体判断字典是否存在该英文单词

struct listNode \*judge;

//创建链表结构体辅助词条数据的搜索功能

struct listNode \*listNext;

//定义一个字符数组作为要搜索的英文单词

char searchWord[WORDNUM+1];

//定义一个字符数组作为要搜索的中文翻译

char searchMean[MEANNUM+1];

//定义一个整形数据接收用户对功能的选择

int choose;

//把用户选择初始化为0

choose = 0;

//把辅助链表结构体初始化为list链表头的下一个结点

listNext = list->next;

//搜索功能菜单

printf("\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|------------------(0)英译汉-----------------------|\n");

printf("|------------------(1)汉译英-----------------------|\n");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\n");

//清除读写缓冲区

fflush(stdin);

//接收用户的选择

printf("请根据需求输入数字(0-1)：");

scanf("%d", &choose);

//当用户的选择在0-1之外的范围则提示用户输入错误，请重新输入

while(choose<0||choose>1)

{

//清除读写缓冲区

fflush(stdin);

printf("输入错误，请重新输入\n");

scanf("%d", &choose);

}

//根据用户选择实现（英译汉，汉译英的功能）

switch (choose)

{

case 0:

//输入要搜索的词条的英文单词

printf("请输入要搜索的英文单词：");

getchar();

scanf("%s", &searchWord);

//判断添加的单词是否已经存在（若存在则显示该词条)

if ((judge = ExistOrNot(list, searchWord)) != NULL)

{

printf("|-------------------搜索结果如下-------------------|\n");

//打印表头

DisplayHead();

//打印一个该词条

DisplayOne(judge);

}

else

{

//若词条不存在则显示“不存在该词条”

printf("不存在该词条\n");

}

break;

case 1:

//输入要搜索的词条的中文

printf("请输入要搜索的中文：");

getchar();

scanf("%s", &searchMean);

//判断添加的单词是否已经存在（若存在则显示该词条)

while (listNext != NULL)

{

if (strcmp((listNext->data).mean,searchMean)==0)

{

printf("|-------------------搜索结果如下-------------------|\n");

//打印表头

DisplayHead();

//打印一个该词条

DisplayOne(listNext);

break;

}

//继续遍历直到listNext=NULL

listNext = listNext->next;

}

break;

default :

//提示用户输入错误

printf("输入错误，请重新输入\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名：FuzzySearch

\* 模糊查找词条内容

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void FuzzySearch(struct listNode \*list)

{

//创建链表结构体辅助词条数据的模糊搜索功能

struct listNode \*listNext;

//定义一个字符数组来接收用户输入的英文字符（串）

char input[WORDNUM+1];

//定义一个字符数组来接收用户输入的英文字符（串）的长度

int inputLen;

//定义一个整形来接收英文单词与用户输入的字符（串）的匹配个数

int matchNum;

//把辅助链表结构体初始化为list链表头的下一个结点

listNext=list->next;

//初始化用户输入的英文字符（串）的长度为0

inputLen=0;

//初始化英文单词与用户输入的字符（串）的匹配个数为0

matchNum=0;

//清除读写缓冲区

fflush(stdin);

//提示用户输入英文单词的关键词

printf("请输入英文单词的关键词：");

scanf("%s", &input);

//接收用户输入英文单词的关键词的长度

inputLen=strlen(input);

//显示表头

printf("|-------------------搜索结果如下-------------------|\n");

//打印表头

DisplayHead();

while(listNext!=NULL)

{

//memcmp比较(listNext->data).word与input的前inputLen个英文字符的匹配度，当两者前input个字符完全匹配时返回值为0

matchNum=memcmp((listNext->data).word,input,inputLen);

//matchNum=0时表示两者前input个字符完全匹配

if(matchNum==0)

{

//显示与input开头匹配的一个英文单词

DisplayOne(listNext);

}

//继续遍历直到listNext=NULL

listNext=listNext->next;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名：wordFormation

\* 按词性分类词条

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void wordFormation(struct listNode \*list)

{

//声明其封装的子函数

void wordFormation1(struct listNode \*list, int choose);

//创建链表结构体辅助词条数据的分类功能

struct listNode \*listNext;

//把辅助链表结构体初始化为list链表头的下一个结点

listNext=list->next;

//把英文单词的词性数字化，并初始化用户选择choose为N所对应的数字

enum {N,V,ADJ,ADV} choose=N;

//词条按词性分类的功能菜单

printf("\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|-----------------(0)名词(n)-----------------------|\n");

printf("|-----------------(1)动词(v)-----------------------|\n");

printf("|-----------------(2)形容词(adj)-------------------|\n");

printf("|-----------------(3)形容词(adv)-------------------|\n");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\n");

//清除读写缓冲区

fflush(stdin);

//提示用户输入数字对应英文单词的词性

printf("请输入数字对应英文单词的词性:");

scanf("%d",&choose);

//用户输入数字不在（0-3）间时提示用户输入错误，请重新输入

while(choose<N||choose>ADV)

{

//清除读写缓冲区

fflush(stdin);

printf("输入错误，请重新输入数字(0-3)：");

scanf("%d",&choose);

}

//按用户输入的数字实现不同的功能

switch(choose)

{

case N:

//显示所有词性为名词的词条

wordFormation1(list,N);

break;

case V:

//显示所有词性为动词的词条

wordFormation1(list,V);

break;

case ADJ:

//显示所有词性为形容词的词条

wordFormation1(list,ADJ);

break;

case ADV:

//显示所有词性为副词的词条

wordFormation1(list,ADV);

break;

default :

//用户输入数字不在（0-3）间时提示用户输入错误，请重新输入

printf("输入错误,请重新输入:\n");

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名：wordFormation1

\* 按词性分类词条（封装函数）

\* 接收函数：wordFormation中的choose变量

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void wordFormation1(struct listNode \*list,int choose)

{

//创建字符指针把上述用户选择符号化

char \*formation;

//创建链表结构体辅助词条数据的分类功能

struct listNode \*listNext;

//统计某词性的英文单词的数量

int num;

//把辅助链表结构体初始化为list链表头的下一个结点

listNext=list->next;

//统计个数初始化为0

num=0;

//把用户选择符号化

switch(choose)

{

case 0:

//strcpy(formation,"n");

formation = "n";

break;

case 1:

//strcpy(formation,"v");

formation = "v";

break;

case 2:

//strcpy(formation,"adj");

formation = "adj";

break;

case 3:

//strcpy(formation,"adv");

formation = "adv";

break;

default :

//用户输入数字不在（0-3）间时提示用户输入错误，请重新输入

printf("输入错误,请重新输入:\n");

break;

}

printf("|-------------------搜索结果如下-------------------|\n");

//显示表头

DisplayHead();

while(listNext!=NULL)

{

if(strcmp((listNext->data).format,formation)==0)

{

//显示一个该词性的词条

DisplayOne(listNext);

//统计该词性词条的个数

num++;

}

//继续遍历直到listNext=NULL

listNext=listNext->next;

}

//显示统计该词性的词条的个数

printf("|-----------字典已记录(%d)条词性为%s词条-----------|\n",num,formation);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名：DeleteContent

\* 删除所选词条内容

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DeleteContent(struct listNode \*list)

{

//创建链表结构体1辅助词条数据的删除功能

struct listNode \*listHead;

//创建链表结构体2辅助词条数据的删除功能

struct listNode \*listNext;

//定义一个字符数组接收要删除的英文单词

char deleteWord[WORDNUM+1];

//定义一个整形作为删除标识

int flag;

//把辅助链表结构体1初始化为list链表头

listHead=list;

//把辅助链表结构体2初始化为list链表头的下一个结点

listNext=list->next;

//初始化删除标识为0

flag=0;

//清除读写缓冲区

fflush(stdin);

//输入要删除的词条的英文单词

printf("请输入要删除的英文单词：");

scanf("%s", &deleteWord);

while(listNext!=NULL)

{

//判断字典中是否存在要删除的英文单词

if(strcmp((listNext->data).word,deleteWord)==0)

{

//把辅助链表结构体1的下一个结点指向辅助链表结构体2的下一个结点

listHead->next=listNext->next;

//释放辅助链表结构体2

free(listNext);

//删除标识标为1

flag=1;

//打印删除成功

printf("删除成功\n");

return;

}

//把辅助链表结构体2赋值给把辅助链表结构体1

listHead=listNext;

//继续遍历直到listNext=NULL

listNext=listNext->next;

}

//判断删除标识若为0则表示删除不成功，显示不存在该单词

if(flag==0)

{

printf("不存在该单词\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：DisplayAll

\*显示所有词条

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DisplayAll(struct listNode \*list)

{

//创建链表结构体1辅助词条数据的显示功能

struct listNode \*listNext;

//创建链表结构体2辅助词条数据的显示功能

struct listNode \*listHead;

//定义一个字符数组作为辅助英文单词

char tmpWord[WORDNUM+1];

//定义一个字符数组作为辅助中文解释

char tmpMean[MEANNUM+1];

//定义一个字符数组作为辅助单词词性

char tmpFormat[WORDNUM+1];

//定义一个整形作为统计单词的数量

int wordNum;

//把辅助链表结构体1初始化为list链表头的下一个结点

listNext=list->next;

//把辅助链表结构体2初始化为list链表头

listHead=list;

//统计单词的数量初始化为0

wordNum=0;

//利用冒泡法，把所有词条按照字母表顺序排序

while(listHead->next!=NULL)

{

while(listHead->next->next!=NULL)

{

if(strcmp((listHead->next->data).word,(listHead->next->next->data).word)>0)

{

strcpy(tmpWord,(listHead->next->data).word);

strcpy(tmpMean,(listHead->next->data).mean);

strcpy(tmpFormat,(listHead->next->data).format);

strcpy((listHead->next->data).word,(listHead->next->next->data).word);

strcpy((listHead->next->data).mean,(listHead->next->next->data).mean);

strcpy((listHead->next->data).format,(listHead->next->next->data).format);

strcpy((listHead->next->next->data).word,tmpWord);

strcpy((listHead->next->next->data).mean,tmpMean);

strcpy((listHead->next->next->data).format,tmpFormat);

}

listHead=listHead->next;

}

listHead=listHead->next;

}

//显示所有记录的词条

printf("|----------------显示所有记录的词条----------------|\n");

//显示表头

DisplayHead();

while(listNext!=NULL)

{

//显示一个词条

DisplayOne(listNext);

//统计所有词条的个数

wordNum++;

//继续遍历直到listNext=NULL

listNext=listNext->next;

}

//显示统计的所有词条的个数

printf("|----------------字典已记录(%d)条词条---------------|\n",wordNum);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：DisplayHead

\*显示表头

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DisplayHead()

{

//显示表头

printf("%10s %10s %10s\n","英文单词","中文解释","单词词性");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：DisplayOne

\*显示一个词条

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DisplayOne(struct listNode \*display)

{

//显示一个词条的内容

printf("%10s %10s %10s\n",(display->data).word,(display->data).mean,(display->data).format);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：Goodbye

\*感谢使用

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Goodbye(double start,double finish)

{

//显示GoodBye界面

printf("\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|--------------------------------------------------|\n");

printf("|---------------感谢使用英汉电子字典---------------|\n");

printf("|---------------作者：罗戬浩-----------------------|\n");

printf("|---------------学号：20170060220------------------|\n");

printf("|---------------专业：17大数据2--------------------|\n");

printf("|---------------创建时间：2018/06/27---------------|\n");

printf("|---------------最后修改时间：2018/07/08-----------|\n");

printf("|---------------文件功能描述：英汉电子字典---------|\n");

printf("|---------------使用时间：%lf 秒--------------|\n",(finish - start) / CLOCKS\_PER\_SEC);

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：DisplayMenu

\*显示菜单

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DisplayMenu()

{

//显示菜单界面

printf("\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|--------------------------------------------------|\n");

printf("|---------------欢迎使用英汉电子字典---------------|\n");

printf("|---------------请根据需求输入数字(0-9)------------|\n");

printf("|---------------(1)添加词条------------------------|\n");

printf("|---------------(2)精确搜索词条--------------------|\n");

printf("|---------------(3)模糊搜索词条--------------------|\n");

printf("|---------------(4)词性分类词条--------------------|\n");

printf("|---------------(5)编辑词条------------------------|\n");

printf("|---------------(6)删除词条------------------------|\n");

printf("|---------------(7)显示词条------------------------|\n");

printf("|---------------(8)保存词条------------------------|\n");

printf("|---------------(9)读取词条------------------------|\n");

printf("|---------------(0)退出字典------------------------|\n");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数名：main

\*主函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(int argc, char \*argv[])

{

//定义一个双精度浮点型记录系统开始运行时间

double start;

//定义一个双精度浮点型记录系统结束运行时间

double finish;

//创建链表结构体为子函数利用

struct listNode \*dictionary;

//数字化表示功能，并把function初始化为DISP对应的数字

enum {EXIT,ADD,SEARCH,FUZSER,CLASSIFY,EDIT,DEL,DISP,SAVE,READ} function=DISP;

//获取系统开始运行时间

start=clock();

//创建链表结构体

dictionary=(struct listNode \*)malloc(sizeof(struct listNode));

if(dictionary!=NULL)

{

//把dictionary的头的下一个结点指向NULL

dictionary->next = NULL;

}

while(function != EXIT)

{

//定义一个整形作为密码标志

int flag;

//把密码标识初始化为0

flag=0;

//显示菜单

DisplayMenu();

//提示用户根据需求输入数字

printf("\n请根据需求输入数字(0-9)：");

scanf("%d",&function);

//用户输入数字不在（0-9）间时提示用户输入错误，请重新输入

while(function<EXIT||function>READ)

{

printf("输入错误，请重新输入数字(0-9)：");

scanf("%d",&function);

}

switch(function)

{

case ADD:

//实现添加词条功能

AddContent(dictionary);

break;

case SEARCH:

//实现搜索词条功能

SearchContent(dictionary);

break;

case FUZSER:

//实现模糊搜索词条功能

FuzzySearch(dictionary);

break;

case CLASSIFY:

//实现按词性分类词条功能

wordFormation(dictionary);

break;

case EDIT:

//先通过密码验证，实现编辑词条功能

flag=Password();

if(flag==1)

{

EditContent(dictionary);

}

break;

case DEL:

//先通过密码验证，实现删除词条功能

flag=Password();

if(flag==1)

{

DeleteContent(dictionary);

}

break;

case DISP:

//实现显示词条功能

DisplayAll(dictionary);

break;

case SAVE:

//实现保存词条功能

SaveContent(dictionary);

break;

case READ:

//实现读取词条功能

ReadContent(dictionary);

break;

case EXIT:

//获取系统结束运行时间

finish=clock();

//实现退出字典功能，并显示GoodBye的界面

Goodbye(start,finish);

//让程序停止等待输入任意键实现退出程序

system("pause");

exit(0);

break;

default:

//用户输入数字不在（0-9）间时提示用户输入错误，请重新输入

printf("输入错误！请重新输入：");

break;

}

}

//结束程序前释放创建的链表结构体dictionary

FreeList(dictionary);

}