|  |  |  |
| --- | --- | --- |
| Assignment Code | : | C.L.P0001 |
| Assignment Name | : | Spell Checker |
| Student Name | : | Le Thi Thanh Nhan |
| Time/Date | : | 00h30,26/9/2019 |

Approach

Covert data from file dictionary.txt to a two-dimensional character array.

Input word, check it in the dictionary or not and print out anouncement.

Some function to find the possible words:

*int substring(char shortstr[], char longstr[]);*

Compare the first character of the word to each character of every word shorter than it in the dictionary. If they are the same, compare anothers charaters of the word in dictionary to anothers characters of the word.

*int subsequence(char shortstr[], char longstr[]);*

Compare the every character of the word to each character of every word longer than it in the dictionary. If they are the same, compare the next character of the word to the rest characters of theword in dictionary.

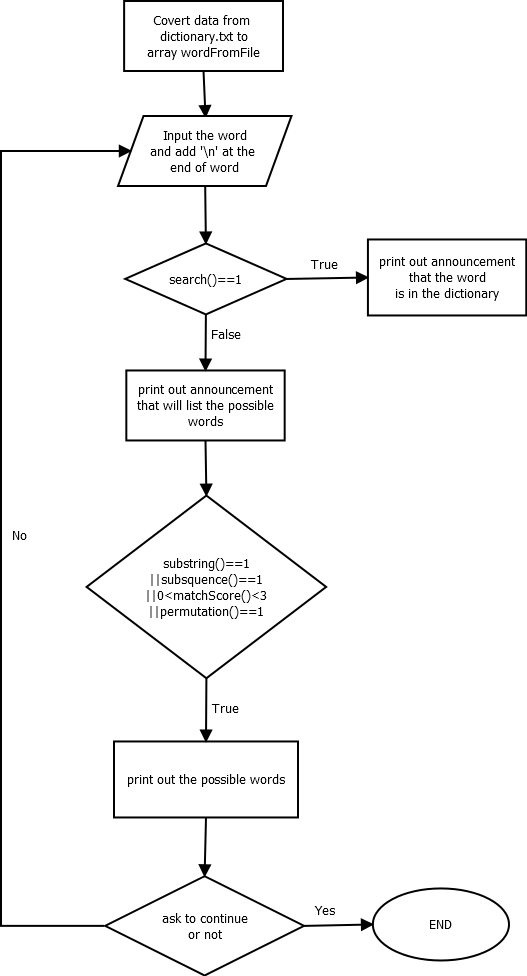
*int permutation(char string1[], char string2[]);*

Create a array count to count the characters different between the word and the word in dictionary by if a character appear in word, count[]++; if that character appear in the word in dictionary, count[]--. After count all characters in 2 words, If all character appearthe same time in 2 words, print out the word in dictionary.

*int matchscore(char string1[],char string2[]);*

Count the characters different between the word and the word in dictionary. If there are smaller 3, bigger or equal 1 character, print out the word in the dictionary.

Flowchart



Source code

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <stdint.h>

#define FILENAME "dictionary.txt"

#define WORDLENGTH 30

#define NUMOFWORDS 62000

#if !(defined \_POSIX\_C\_SOURCE) || \_POSIX\_C\_SOURCE < 200809L

#if !(defined SSIZE\_MAX)

#define SSIZE\_MAX (SIZE\_MAX >> 1)

#endif

ssize\_t getline(char \*\*pline\_buf, size\_t \*pn, FILE \*fin)

{

    const size\_t INITALLOC = 16;

    const size\_t ALLOCSTEP = 16;

    size\_t num\_read = 0;

    if ((NULL == pline\_buf) || (NULL == pn) || (NULL == fin))

    {

        errno = EINVAL;

        return -1;

    }

    if (NULL == \*pline\_buf)

    {

        \*pline\_buf = malloc(INITALLOC);

        if (NULL == \*pline\_buf)

            return -1;

        else

            \*pn = INITALLOC;

    }

    {

        int c;

        while (EOF != (c = getc(fin)))

        {

            num\_read++;

            if (num\_read >= \*pn)

            {

                size\_t n\_realloc = \*pn + ALLOCSTEP;

                char \*tmp = realloc(\*pline\_buf, n\_realloc + 1);

                if (NULL != tmp)

                {

                    \*pline\_buf = tmp;

                    \*pn = n\_realloc;

                }

                else

                    return -1;

                if (SSIZE\_MAX < \*pn)

                {

                    errno = ERANGE;

                    return -1;

                }

            }

            (\*pline\_buf)[num\_read - 1] = (char)c;

            if (c == '\n')

                break;

        }

        if (EOF == c)

        {

            errno = 0;

            return -1;

        }

    }

    (\*pline\_buf)[num\_read] = '\0';

    return (ssize\_t)num\_read;

}

#endif

int substring(char shortstr[], char longstr[]);

int subsquence(char shortstr[], char longstr[]);

int matchScore(char str1[], char str2[]);

int permutation(char str1[], char str2[]);

int search(char str[][WORDLENGTH], int n, char s[]);

int main()

{

    char \*lineBuf = NULL;

    size\_t lineBufSize = 0;

    int lineCount = 0;

    ssize\_t lineSize;

    FILE \*fp = fopen(FILENAME, "r");

    char wordInput[WORDLENGTH], wordFromFile[NUMOFWORDS][WORDLENGTH], wordInp[WORDLENGTH], choice = 'y';

    printf("Welcome to the Spell Checker!\n");

    if (!fp)

        printf("Cannot read file %s\n", FILENAME);

    else

        printf("The dictionary has been loaded!\n");

    lineSize = getline(&lineBuf, &lineBufSize, fp);

    while (lineSize >= 0)

    {

        lineCount++;

        strcpy(wordFromFile[lineCount], lineBuf);

        lineSize = getline(&lineBuf, &lineBufSize, fp);

    }

    lineCount++;

    while (choice == 'y')

    {

        fflush(stdin);

        printf("Please enter the word you would like checked.\n");

        gets(wordInput);

        strcpy(wordInp, wordInput);

        strcat(wordInput, "\n");

        if (search(wordFromFile, lineCount, wordInput) == 1)

            printf("Great, %s is in the dictionary!\n", wordInp);

        else

        {

            printf("Here are the possible words you could have meant:\n");

            for (int i = 0; i < lineCount; i++)

            {

                if (substring(wordInput, wordFromFile[i]) == 1 || subsquence(wordInput, wordFromFile[i]) == 1 || (matchScore(wordFromFile[i], wordInput) < 3 && matchScore(wordFromFile[i], wordInput) >= 0) || permutation(wordFromFile[i], wordInput) == 1)

                    printf("%s", wordFromFile[i]);

            }

        }

        fflush(stdin);

        printf("Would you like to enter another word? (y/n)\n");

        scanf("%c", &choice);

    }

    free(lineBuf);

    lineBuf = NULL;

    getchar();

    return 0;

}

int substring(char shortstr[], char longstr[])

{

    char \*p1, \*p2, \*p3;

    int i = 0, j = 0, flag = 0, lenShort = strlen(shortstr), lenLong = strlen(longstr);

    p1 = longstr;

    p2 = shortstr;

    if (lenShort > lenLong)

        return 0;

    for (i = 0; i < lenLong; i++)

    {

        if (\*p1 == \*p2)

        {

            p3 = p1;

            for (j = 0; j < lenShort; j++)

            {

                if (\*p3 == \*p2)

                {

                    p3++;

                    p2++;

                }

                else

                    break;

            }

            p2 = shortstr;

            if (j == lenShort)

            {

                flag = 1;

            }

        }

        p1++;

    }

    return flag;

}

int subsquence(char shortstr[], char longstr[])

{

    int i = 0, j = 0;

    if (strlen(shortstr) > strlen(longstr))

        return 0;

    while (shortstr[i] != '\0')

    {

        while (shortstr[i] != longstr[j] && longstr[j] != '\0')

            j++;

        if (longstr[j] == '\0')

            break;

        j++;

        i++;

    }

    return shortstr[i] == '\0' ? 1 : 0;

}

int matchScore(char str1[], char str2[])

{

    int len1 = strlen(str1), len2 = strlen(str2), count = 0;

    if (len1 != len2)

        return -1;

    for (int i = 0; i < len1; i++)

    {

        if (str1[i] != str2[i])

            count++;

    }

    return count;

}

int permutation(char str1[], char str2[])

{

    int len1 = strlen(str1), len2 = strlen(str2), i;

    int count[26] = {0};

    if (len1 != len2)

        return 0;

    for (i = 0; str1[i] && str2[i]; i++)

    {

        count[str1[i] - 'a']++;

        count[str2[i] - 'a']--;

    }

    for (i = 0; i < 26; i++)

        if (count[i])

            return 0;

    return 1;

}

int search(char str[][WORDLENGTH], int n, char s[])

{

    for (int i = 0; i < n; i++)

    {

        if (strcmp(str[i], s) == 0)

            return 1;

    }

    return 0;

}

Result



…



Reference

<https://drive.google.com/drive/u/1/folders/136qLj6Ra9VEeixKJeUqaNW5P2TcxUcqw>