Hacettepe University

Computer Science and Engineering Department

Fall 2017

Name and Surname	***: Tolgahan DİKMEN***
ID Number	***: 21327929***
Course	***: BBM203 Programming Lab.***
Experiment	***: Assignment 3***
Subject	***: Linked List***
Due Date	***: 03.12.2017***
Advisor	***: R.A Pelin CANBAY***
E-mail	***: tolgahandikmen95@gmail.com***

1. INTRODUCTION

In this assignment, a basic document analysis program design is done using the C programming language. The primary goal of this experiment is to get you practice on linked lists and matrices.

Document analytics is the process of deriving information from text. The documents are examined to obtain information about authors by analyzing the texts. The design consists of two different parts. The first part starts reading input which is given by user via .txt file, then parse the file formed in a similar way. The second part is to create a linked list which has holds recent words. The implementation has to be created dynamically because of structure of linked list.

According to design, the commonly used similarity measure is the Cosine Similarity. Cosine similarity is a measure of similarity between non-zero vectors of an inner product space that measures the cosine of the angle between them, in information retrieval approaches.

2. SOFTWARE USING DOCUMENTATION

2.1. Software Usage

The program does not need any arguments to work. File analysis program is worked with only a few commands.

The commands which is used in the program:

- **-r :**

  Format: ’-r’ ’path of file’

  In this command, the application will read the given file word by word. Then creates a linked list according to the words read.

- **-a :**

  Format: ’-a’ ’word’ ’word value’ ‘path of file’

  In this command, the application will add the given word and its count to the linked list of the given file.

- **-n2 :**

  Format: ’-n2’ ’path of file’

  In this command, the application will return the first 3 pairs of words that pass the most recent on the given file.

- **-d :**

  Format: ’-d’ ‘word’ ’path of file’

  In this command, the application will delete the node which holds the information of given word, from the linked list of the given file.

- **-s :**

  Format: ’-s’ ‘path of file1’ ’path of file2’

  In this command, the application will return the cosine similarity of the given file ‘file1’ and the given file ‘file2’.

- **-q :**

  Format: ’-q’

  After this command, the application will close.

2.2. Error Messages

The program may return error when trying to read a file which does not exist in –s, -d, -a, -n2 commands. But it does not matter if you enter these commands after running the file with the -r command.

Also when the program is compiled, there is a warning like that:

warning: ‘gets’ is deprecated

warning: the `gets' function is dangerous and should not be used.

This warning comes because of safety. In order to use 'gets', it had to know exactly how many characters it will be reading, so that it can make its buffer large enough.

3. SOFTWARE DESIGN NOTES

3.1. Definition About The Values

Variables and arrays which has been used:

- **char command[1000]:** Holds to received commands from the user.
- **Header headers[100]:** keeps the head values of the linked lists created and their file name(as path) of the files.

3.2. Definition About The Functions and Structures

The program consists three structure of struct type.

```
typedef struct Word {
    char word[50];                  // string of a word
    int wordCount;                  //value of word(recent time number)
    struct Word * next;                     // points to next node
    struct RelatedWord * relatedWord;     // points to next related node
} Word;
```

That structure a node of the basis of the linked list structure. Word struct keeps the word which passed in the file.

```
typedef struct RelatedWord {
    char word[50];                  // string of a word
    int wordCount;                  // value of word(recent time number)
    struct RelatedWord * next;              // points to next node
} RelatedWord;
```

That structure a node of the basis of the nested linked list structure(for the related words). RelatedWord struct keeps the ‘next word’ information.

```
typedef struct OriginalText {
    char word[50];                  // string of a word
    struct OriginalText * next;     // points to next node
} OriginalText;
```

That structure a node of the basis of the linked list structure(for the all words). OriginalText struct keeps the all words information which exists in the file.

```
typedef struct Header {
    char *fileName;                 // string of a file(as path)
    Word * fileHeader;              // head information of the given file
    int control;                    // control for empty node or not
} Header;
```

That structure is implements with an array. Header struct keeps name of file information and its header pointer of linked list.

Functions about queue structure:

Word* readInputFileToCreateLinkList (char *fileName, Word*, OriginalText *)

Function to read a given file and create to linked list.

Word* addWord (Word *, char word[50], OriginalText *,int *readWordSize)

Function to add an element to linked list.

RelatedWord* addRelatedWord (RelatedWord *, char word[50])

Function to add to related words of a word to the linked list.

OriginalText* addOriginalWord (OriginalText* , char word[50])
Function to create a linked list which keeps all file sequentially.

Word * bubbleSort (Word *start)

Function to bubble sort the given linked list.

void swap (Word *a, Word *b)

Function to swap data of two nodes a and b.

void push mostPairs (Word *)

Function to print most 3 word pairs.

double similarity (Word *, Word *)

Function to calculate cosine similarity of given linked lists.

char* strlwr(char* s)

Function to use strlwr() function. Because is not allowed in ansi.

3.3. Algorithm

1. Take commands from the user until ‘-q’ command is came
2. Find which command is entered
3. If command is ‘–r’ check is there any linked list about the file, there is no linked list about the file, create one linked list and fill it up, keep head of the linked list in the array
4. If command is ‘-a’ parse the command, add to linked list the given word with its value
5. If command is ‘-n2’
   - 5.1. Find the most recent 3 word pairs of the linked list of the given file
   - 5.2. Print them to screen
6.
   - 6.1. If command is ‘-d’ delete the given word from the linked list of the given file
   - 6.2. Free to deleted node from the memory
7. If command is ‘-s’
   - 7.1. Find the top 10 words for each file
   - 7.2. Fit these words to a matrix without repetition
   - 7.3. Calculate cosine similarity of given two files according to created matrix

4. COMMENTS

This assignment is very educational for linked list logic, string operations and dynamic memory management. Also pointers need to be used for structures and linked list. In a given sense this assignment comes up with the time and difficulties in a parallel way. The materials provided and the answering to each question was very good.

5. REFERENCES

- https://www.tutorialspoint.com/cprogramming/c_structures.htm
- https://www.programiz.com/c-programming/c-structures
- http://www.learn-c.org/en/Linked_lists
- http://www.zentut.com/c-tutorial/c-linked-list/
- http://www.geeksforgeeks.org/data-structures/linked-list/
- https://www.tutorialspoint.com/c_standard_library/math_h.htm
- https://www.tutorialspoint.com/c_standard_library/string_h.htm