Library and Recent List Search

by Mark Aquiapao

CECS 326 Operating Systems

09/10/2019

Description of Program:

The program consists of a library which contains 1024 documents and a recent list which contains 128 documents where each document will be randomly initialized with letters of the alphabet and have a memory capacity between 2-3 MB. The user function of the program is for the user to input any keyword from the dictionary mentioned in the assignment. Once the user has correctly entered a word from the dictionary, the program will search for each document in the recent list and eject any document that does not contain the word. The ejected document is then put at the end of the library and is reinitialized, where the first document of the library is pushed at the end of the recent list.

In order to create the documents for the library and recent list, the software declared each as a structure where the library's documents was an array and recent list used pointers. Each document was randomly assigned a memory size variable using the rand() to choose a number between 2,000,000(2 MB) and 3,000,000(3MB). The software than used that variable as the number of times that a for-loop will iterate assigning a random letter(character=1byte) using the rand(). In finding a keyword within a string, the function called SearchWord() used the find() to pass a true or false if the keyword was found in the document.

For the algorithm in running the function of ejecting and swapping documents between the recent list and library there are (3) simple steps:

1) Save the ejected document's size from the recent list and the first document from library as temporary variables.
2) Shift the documents in each structure using a for-loop that will assign the current document with the contents of the next document until the last document where it will be assigned a blank value. This method is also used to delete/eject the document that did not match the keyword entered.
3) The last step to then reinitialize the last document in the shifted library with the ejected document's saved size and copy the saved first document form the library into the last document in the shifted recent list.

Critique: By both searching and sorting by one document at a time, the program run slower than expected which would be more efficient if it searched for all documents first then resorted them.

```
/*

 ----------------------------------------------------------------------
 CECS 326 Assignment 1: Programming Review
 Instructor: Ratana Ngo
 File: Assign1.cpp
 Located in Assign1_ProgramReview folder
 ----------------------------------------------------------------------
 Created by Mark Aquiapao on 9/2/2019.
 Copyright © 2019 Mark Aquiapao. All rights reserved.
 ----------------------------------------------------------------------
 * Rev A by Mark Aquiapao on 9/5/2019
 Description: Redid initialization for library & recent list.
 Added function for finding keyword in document.

 * Rev B by Mark Aquiapao on 9/7/2019
 Description: Revised function for searching a substring within a string.
 Added and revsied algrorithm for sorting out the library
 and recent list structures if a document in relecent list
 does not match.
 Added logic to verify entry in dictionary and revised display.

 * Rev C by Mark Aquiapao on 9/9/2019
 Description: Revised GUI(Termninal) to be user friendly and simple.

 ----------------------------------------------------------------------
 Performance of Software on 9/8/2019:
 - CPU: 71%-99%
 - Memory: > 3 GB
 - Energy Impact: High
 ----------------------------------------------------------------------

 */

// Header Files
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
using namespace std;

// Prototype:
bool SearchWord(string, string);

// Declare Constants and Globals
const int   DOC_SIZE = 1024, POINTER = 128, MAX_MEM = 3000000, MIN_MEM = 2000000;

// Structures for library and recent list
```

```
struct library {
   string doc;
};

struct recent_list {
   string rl_doc;
};

// Dictonary to verfiy user input
string dictionary[15] = {   "FIRST",
   "CPP",
   "REVIEW",
   "PROGRAM",
   "ASSIGNMENT",
   "CECS",
   "BEACH",
   "ECS",
   "FALL",
   "SPRING",
   "OS",
   "MAC",
   "LINUX",
   "WINDOWS",
   "LAB"      };

int main()
{
   // Declare main variables
   library lib_document[DOC_SIZE];
   recent_list rl_document[POINTER];
   recent_list* ptr; // pointer for recent list
   ptr = rl_document;
   string key;
   bool valid = false; bool quit = false;
   int mem_size, count;
   int rl_memSize[POINTER];

   cout << "\n";
   cout << "************************************\n";
   cout << "Welcome to Assignment 1 of CECS 326!\n";
   cout << "************************************\n\n";
   cout << "Documents in library are initializing... \n";

   // Initialize 1024 documents for library
   for(int i=0; i<DOC_SIZE; i++)
   {
      // Choose memory size between 2MB and 3MB randomly
      // char = 1 byte -> 1 MB = 1,000,000 bytes
```

```
        mem_size = (rand() > RAND_MAX/2) ? MIN_MEM : MAX_MEM;
        for(int j=0; j<mem_size; j++)
        {
            lib_document[i].doc += rand() % 25 + 'A'; // pick randomly from A-Z
        }
    }

    cout << "Documents in recent list are initializing... \n\n";

    // Initialize 128 documents for recent_list
    for(int i=0; i<POINTER; i++)
    {
        // Choose memory size between 2MB and 3MB randomly
        // char = 1 byte -> 1 MB = 1,000,000 bytes
        rl_memSize[i] = (rand() > RAND_MAX/2) ? MIN_MEM : MAX_MEM;
        for(int j=0; j<rl_memSize[i]; j++)
        {
            (ptr+i)->rl_doc += rand() % 25 + 'A'; // pick randomly from A-Z
        }
    }

    do
    {
        valid = false;
        cout << "-----------------------------------------------------------------\n";
        cout << "Dictionary: FIRST, CPP, REVIEW, PROGRAM, ASSIGNMENT, CECS, BEACH,
ECS,\n";
        cout << "            FALL, SPRING, OS, MAC, LINUX, WINDOWS, LAB \n";
        cout << "-----------------------------------------------------------------\n\n";
        cout << "Enter keyword from dictionary: ";
        cin >> key;
        // Check if Key is in Dictionary
        do
        {
            for(int i=0; i<16; i++)
            {
                if(key==dictionary[i])
                    valid = true;
            }
            if(valid == false)
            {
                cout << "Invalid entry! Try again: ";
                cin >> key;
            }
        }while(valid==false);

        cout << "\n";
        count = 0;
```

```
for(int pos=0; pos<POINTER; pos++)
{
   // For all documents that do NOT match keyword:
   if(SearchWord((ptr+pos)->rl_doc, key)==false)
   {
      cout << "Document [" << pos << "] ejected...\n";
      count++; // document found, increment count

      string lib_temp = lib_document[0].doc;  // Save first document from library
      int temp_memSize = rl_memSize[pos];     // Save size of ejected doc from recent list

      // Eject document from recent list and shift other documents
      for(int i=pos; i<POINTER; i++)
      {
         if(i!=POINTER-1) // delete document that doesn't MATCH and shift the rest
            (ptr+i)->rl_doc =(ptr+(i+1))->rl_doc;
         else
            (ptr+i)->rl_doc = " "; // Empty value for last index
      }

      // Shift documents in library up and leave last element empty/null
      for(int j=0; j<DOC_SIZE; j++)
      {
         if(j!=DOC_SIZE-1) // Shift the library documents
            lib_document[j].doc = lib_document[j+1].doc;
         else
            lib_document[j].doc = " "; // Empty value for last index
      }


      // Reinitialized last library document with same size as ejected document
      for(int j=0; j<temp_memSize; j++)
      {
         lib_document[DOC_SIZE-1].doc += rand() % 25 + 'A'; // pick randomly from A-Z
      }
      // Insert swapped 1st document of library to end of recent list
      (ptr+(POINTER-1))->rl_doc = lib_temp;
   }
}

cout << "-----------------------------------------------------------------\n";
if(count==0)
   cout << key << ": " << count << " document(s) ejected \n";
else
   cout << key << ": " << count << " document(s) ejected and reinitialized \n";
cout << "-----------------------------------------------------------------\n\n\n";

cout << "Quit program? (Y/N): ";
```

```
        cin >> key;
        if(key=="Y" || key=="y")
        {
            cout << "\n";
            cout << "************************************\n";
            cout << "Program Terminated... \n";
            cout << "Goodbye! \n";
            cout << "************************************\n\n";
            quit = true;
        }
        else
            quit = false;
    }while(quit==false);

    return 0; // terminate program

}

bool SearchWord(string document, string key)
{
    bool found = false;
    if(document.find(key) != string::npos)
        found = true; // key is found in document
    return found;
}
```