**Program 5 – Part 1**

**Demetrius Johnson**

**CIS 350 – Summer II**

**Professor Thomas G. Steiner**

**August 8, 2021**

1. **Problem Statement**

Write a program to read in a grid of letters from a file, and then interactively allow a user to enter phrases until the user wants to quit. For each phrase the program must output whether or not the phrase is found and, if found, specifically where it is located. In addition, an output file will record what is displayed on the screen plus steps to find/not find the word or phrase.

1. **Requirements**
   1. **Assumptions**

Input file values will be integers stating grid dimensions of character grid, followed by the grid

File FORMAT is correct

Check that all characters in the grid are characters

else out put error and terminate

Check that values on line 1 is greater than 0 (grid dimensions)

else output error message and terminate program

Uppercase letters in the file will be converted to lowercase – warning message written but program continues

Perform File Error Checks (file exists/can be opened..etc)

No letter may appear more than one time in any part of a solution.

(i.e. phrase “go out” the letter “o” at position [3] [5] cannot be used for both “go” and “out”; there needs to be two “o”s.

File will contain only one character grid

The grid of letters for the program will be stored in a text file formatted as follows:

Line 1: Two integers separated by a single blank space. These will

represent the number of rows.

Remaining lines: number of rows lines each containing number of columns characters.

The user input will be phrases of words, with a single space between each word. Ignore multiple space if they occur – no message – test case!

The program needs to edit the user entered data.

Each phrase will be entered on a single line (assumption!).

User input will be converted to all lowercase letters

User will be informed of invalid word or phrase and be prompted to reenter word

Program terminates when user quits

A recursive, backtracking algorithm must be used to search the grid

No letter may appear more than one time in any part of a solution

i.e. phrase “go out” the letter “o” at position [3] [5] cannot be used for both “go” and “out”; there needs to be two “o”s.

Hierarchy of direction choice for a search: right, down, left, up

If the last letter in a word is at location (i, j), the first letter of the next word must be at one of locations (i, j+1), (i+1, j), (i, j-1), or (i-1, j).

The direction chosen to find the first letter of a word is the same direction that must be used for all of the letters of the word.

If a phrase is not found in the grid, output should simply state that

If phrase found, program must find one occurrence of the phrase and indicate it

* 1. **Specifications**

MAIN FUNCTION

Display message “Welcome to the Backtracking Word Search Test Program” to user

Display message “Enter output file name: ” to user

Read and use the user entered output file name

If output file cannot be used

Display message “file <user output file name> cannot be opened – program terminated” to user

Display message “Welcome to the Backtracking Word Search Test Program” to output file

Display message “Testing Default Scenario” to user and output file

Create a 4x4 character table and test functionality

Call printTable function

Call a test phrase to search using backtrack algorithm

Call printPhraseTable

Display message “Enter file name for character table data: ” to user

Read user entered input data file name

Display message “File name for character table data: <input file name>” to output file

Display message “Testing File Data” to user and output file

Perform file validation

If cannot open

Display message “file <user input file name> cannot be opened or does not exist – program terminated”

If file exists but is empty

Display message “file <user input file name> contains no data – program terminated”

If number of rows is invalid (<= 0)

Display message “Error: Invalid number of rows: ” <row> “. Program terminated.”

If number of columns is invalid (<= 0)

Display message “Error: Invalid number of columns: ” <col>

“. Program terminated.”

Read in all characters from the input file into a 2D dynamic array While not end of file

Display message <row> “x”<col> “ 2D array (word search character table) created from user file…building table…” to user and output file

If a character from the table is invalid (non-alphabetical)

Display message “Error: Invalid character in table: ” <char> “in position ” <[i][j]> “. Program terminated.” to user and output file

If a character from the table is uppercase

Display message “Warning: Uppercase letter in table input: ” <char> “in position ” [i][j] “. Converted to lower case.” to user and output file

If EOF reached

Display message “User input table read successfully…program launch is complete: ready to search words. User table: ” To user and output

Call printTable

Display message “Enter a letter, word, or phrase to search (alphabetical characters and spaces only). Enter a single non alphabetical character to quit the program.” to user

While user input is not a single alphabetical character

Display message “Enter a word or phrase to search, or enter a single nonalphabetical character to quit: ”

Get user input and store it in a string

Display message “User search phrase: ” <user input string> to output file

Call string parse function to delete leading and additional white spaces

If string size is 1 and contains a nonalphabetical

Display message “A single nonalphabetical character <user input string> has been entered: quitting program…” to user and output and then terminate program

Otherwise check that user input has only spaces and alphabetical characters

While there are characters in the user input string to check

If a character is nonalphabetical

Display message “Error: Invalid word or phrase: ‘<user input string>’. Re-enter word/phrase: ” to user

If a character is an uppercase letter

Convert letter to lowercase

Call backtrackingWordSearch function for user input

If the phrase is not found in the grid

Display message “The user input word/phrase <user input string> was not found in the table.”

Otherwise the program has found an occurrence of the phrase

Display message “The user input word/phrase was found!”

Call outputPhraseCoordinates function

Call printPhraseFoundTable

BACKTRACKING WORD SEARCH FUNCTION (priority: right, down, left, up) – display all messages to output file

While there are words to search (recursive algorithm operating on each word), search the characters of all words to see if the word is in the table, each word is consecutive to the previous, if current word reaches a dead end, then go back to the end of the previous word and start there (use boolean array to mark a location when all directions have been searched so that we know to go back to yet the next word’s ending; if we reach the first word on backtracking, then the phrase is not in the table)

Display message “Start (0,0) looking for ‘<user input[0]>’

If user input[0] is not found at (0,0)

Display message “ – not found”

Move right of (i,j) for user input[x]

If user input[x] is not found at (i,j)

Display message “ – not found”

Move to the down(or left, up, depending on w of (0,0)

Execute the same above logic for each direction in the direction hierarchy

If all directions at a given [i][j] has been checked, then mark a boolean array noting that, or if a direction does contain the letter being searched but has already been navigated, another boolean array will mark it so we know it is already a traveled dead end.

STRING PARSE FUNCTION – deletes leading or additional white spaces

PRINT TABLE FUNCTION – simple loop to print the 2D array from the input file

PRINT PHRASE FOUND TABLE FUNCTION – prints 2D array, capitalizing all positions where the word/phrase was found

OUTPUT PHRASE COORDINATES FUNCTION – print the coordinates of the first and last letter of each word in the phrase to user and output file

CREATE TABLE FUNCTION – creates the 2D array from input file

1. **Decomposition Diagram** (Used to break program down into components visually. Can have as many components as needed. Defines functionality that will solve the problem – does NOT define a flow )

Main

* Input
  + User file name
    - File validation
  + File Data
    - File data edits
      * Format:
        + Rows, column
        + Letter grid
* Process
  + Create table
  + Perform backtracking word search
* Output
  + Welcome message
  + Input error messages
  + Print user input table
  + Print phrase found table
  + Print phrase word coordinates
  + End message

1. **Test Strategy**

File Testing (exist, empty)

Valid data

Invalid data

1. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| File Testing | 1 | File does not exist |  |  |  |  |
| File Testing | 2 | File exists but empty |  |  |  |  |
| Valid data | 3 | Valid data from input file |  |  |  |  |
| Invalid data | 4 | Invalid number of rows or columns from input file |  |  |  |  |
| Invalid data | 5 | Invalid character in input table |  |  |  |  |
| Invalid data | 6 | Invalid user input for phrase |  |  |  |  |
| Valid data | 7 | Quit program nonalphabetical entered by user |  |  |  |  |
| Valid data | 8 | Find a phrase, restart loop |  |  |  |  |
| Valid data | 9 | Do not find a phrase or word |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Initial Algorithm**

Data: Object Definitions

Class Backtracking

Data:

String\* userStringPtr

Vector<bool> allDirectionsSearched

Actions:

parseString

printPhraseCoord

Class Table

Data:

Char\*\* arrayTable2D //2D array

Actions:

buildTable

printTable

printPhraseTable

destroyTable

MAIN FUNCTION

Display message “Welcome to the Backtracking Word Search Test Program” to user

Display message “Enter output file name: ” to user

Read and use the user entered output file name

If output file cannot be used

Display message “file <user output file name> cannot be opened – program terminated” to user

Display message “Welcome to the Backtracking Word Search Test Program” to output file

Display message “Testing Default Scenario” to user and output file

Create a 4x4 character table and test functionality

Call printTable function

Display message “Testing File Data” to user and output file

Display message “Enter file name for character table data: ” to user

Read user entered input data file name

Display message “File name for character table data: <input file name>” to output file

Perform file validation

If cannot open

Display message “file <user input file name> cannot be opened or does not exist – program terminated”

If file exists but is empty

Display message “file <user input file name> contains no data – program terminated”

If number of rows is invalid (<= 0)

Display message “Error: Invalid number of rows: ” <row> “. Program terminated.”

If number of columns is invalid (<= 0)

Display message “Error: Invalid number of columns: ” <col>

“. Program terminated.”

Read in all characters from the input file into a 2D dynamic array While not end of file

Display message <row> “x”<col> “ 2D array (word search character table) created from user file…building table…” to user and output file

If a character from the table is invalid (non-alphabetical)

Display message “Error: Invalid character in table: ” <char> “in position ” <[i][j]> “. Program terminated.” to user and output file

If a character from the table is uppercase

Display message “Warning: Uppercase letter in table input: ” <char> “in position ” [i][j] “. Converted to lower case.” to user and output file

If EOF reached

Display message “User input table read successfully…program launch is complete: ready to search words. User table: ” To user and output

Call printTable

Display message “Enter a letter, word, or phrase to search (alphabetical characters and spaces only). Enter a single non alphabetical character to quit the program.” to user

While user input is not a single alphabetical character

Display message “Enter a word or phrase to search, or enter a single nonalphabetical character to quit: ”

Get user input and store it in a string

Display message “User search phrase: ” <user input string> to output file

Call string parse function to delete leading and additional white spaces

If string size is 1 and contains a nonalphabetical

Display message “A single nonalphabetical character <user input string> has been entered: quitting program…” to user and output and then terminate program

Otherwise check that user input has only spaces and alphabetical characters

While there are characters in the user input string to check

If a character is nonalphabetical

Display message “Error: Invalid word or phrase: ‘<user input string>’. Re-enter word/phrase: ” to user

If a character is an uppercase letter

Convert letter to lowercase

Call backtrackingWordSearch function for user input

If the phrase is not found in the grid

Display message “The user input word/phrase <user input string> was not found in the table.”

Otherwise the program has found an occurrence of the phrase

Display message “The user input word/phrase was found!”

Call outputPhraseCoordinates function

Call printPhraseFoundTable

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| File Testing | 1 | File does not exist | Wrong file name | “file not found” |  |  |
| File Testing | 2 | File exists but empty | Empty file | “file is empty” |  |  |
| Valid data | 3 | Valid data from input file | Valid test file | “test table created…user input table created” |  |  |
| Invalid data | 4 | Invalid number of rows or columns from input file | Test file with incorrect number of rows and columns | “invalid row or column value for table” |  |  |
| Invalid data | 5 | Invalid character in input table | Input $ into table | “non alphabetical char in table…” |  |  |
| Invalid data | 6 | Invalid user input for phrase | Input test$ | “re-enter a phrase, invalid char found” |  |  |
| Valid data | 7 | Quit program nonalphabetical entered by user | Enter ` | “` entered; program quitting” |  |  |
| Valid data | 8 | Find a phrase, restart loop | Search “test phrase” | Phrase found; output coordinates; output table |  |  |
| Valid data | 9 | Do not find a phrase or word | “test phrase not here” | Phrase not found |  |  |
|  |  |  |  |  |  |  |

Part 1 ends here!!!!!!