**DATA STRUCTURES AND ALGORITHM**

**ASSIGNMENT 5 ( TRIES )**

**INSTRUCTOR SIR ABID RAUF**

**BSCS 6-C**

**MEMBERS: HASSAAN AKBAR CHEEMA( REG NO. 174351 )**

**NOUMAN RIAZ ( REG NO. 197053)**

**NOTE: each time to run this code clean the solution then build the solution and then start debugging to avoid crash.**

**CODE:**

#include "iostream"

#include "fstream"

#include "istream"

#include "string"

using namespace std;

string text;

const int ALPHABET\_SIZE = 26;

// trie node

struct TrieNode

{

struct TrieNode \*children[ALPHABET\_SIZE];

// isEndOfWord is true if the node represents

// end of a word

bool isEndOfWord;

};

// Returns new trie node (initialized to NULLs)

struct TrieNode \*getNode(void)

{

struct TrieNode \*pNode = new TrieNode;

pNode->isEndOfWord = false;

for (int i = 0; i < ALPHABET\_SIZE; i++)

pNode->children[i] = NULL;

return pNode;

}

// If not present, inserts key into trie

// If the key is prefix of trie node, just

// marks leaf node

void insert(struct TrieNode \*root, string word)

{

struct TrieNode \*temp = root;

for (int i = 0; i < word.length(); i++)

{

int index = word[i] - 'a';

if (!temp->children[index])

temp->children[index] = getNode();

temp = temp->children[index];

}

// mark last node as leaf

try

{

temp->isEndOfWord = true;

}

catch (exception& e)

{

cout << "Standard exception: " << e.what() << endl;

}

}

// Returns true if key presents in trie, else

// false

bool search(struct TrieNode \*root, string word)

{

struct TrieNode \*temp = root;

for (int i = 0; i < word.length(); i++)

{

int index = word[i] - 'a';

if (!temp->children[index])

return false;

temp = temp->children[index];

}

return (temp != NULL && temp->isEndOfWord);

}

void print(struct TrieNode\* root, int index)

{

if (root == NULL)

{

return;

}

else

{

char data = (char)(index + 'a');

text += data;

if (root->isEndOfWord == true)

{

cout << text << endl;

}

for (int i = 0; i < 26; i++)

{

print(root->children[i], i);

}

}

string temp = text.substr(0, (text.length() - 1));

text = temp;

return;

}

int main()

{

string keys[3000];

ifstream bucky;

struct TrieNode \*root = getNode();

bucky.open("words.txt");

int i = 0;

while (!bucky.eof())

{

bucky >> keys[i];

insert(root, keys[i]);

i++;

}

bucky.close();

int run = 1;

while (run == 1) {

int choice = 0;

string alpha;

cout <<endl<<endl<< ".................welcome to the dictionary...................." << endl;

cout <<endl<< "to print all the words in the dictionary press 1:" << endl;

cout << "to insert a word in the dictionary press 2:" << endl;

cout << "to search a word in the dictionary press 3:" << endl;

cin >> choice;

if (choice == 1)

print(root, -97);

if (choice == 2) {

cout << "enter the word:";

cin >> alpha;

insert(root, alpha);

ofstream chucky;

chucky.open("words.txt", std::ios::app);

chucky << "\n" << alpha;

chucky.close();

}

if (choice == 3) {

cout << "enter the word:";

cin >> alpha;

search(root, alpha) ? cout << "the word " << alpha << " exist in the dictionary" << endl :

cout << "the word " << alpha << " does not exist in the dictionary" << endl;

}

cout << "to continue using the dictionary press 1 and to close it press 0:";

cin >> run;

}

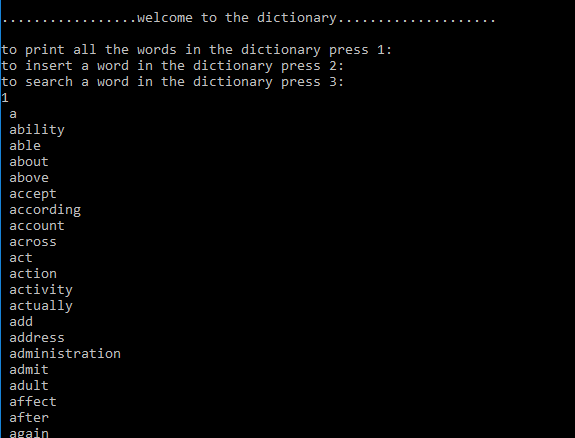
cout <<endl<< "...........................thankyou for using our dictionary" << endl;

return 0;

system("pause");

}

**OUTPUT:**



**THE WORDS CONTINUES…**

