#include<iostream>

#include<fstream>

#include<string>

#include<ctime>

#include<Windows.h>

#include<conio.h>

const int MAX\_SUGGESTIONS = 10;

using namespace std;

class trieNode

{

public:

trieNode\* children[26];

bool isWordEnd;

string meaning;

//constructor

trieNode()

{

isWordEnd = false;

//initaialize the array

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

{

children[i] = 0;

}

meaning = "";

}

};

class Trie {

public:

trieNode\* root;

Trie() {

root = new trieNode;

}

//insert the word

bool insert(string word, string meaning)

{

trieNode\* t = root;

int index;

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

{

index = word[i] - 'a';

if (t->children[index] == nullptr)

{

t->children[index] = new trieNode;

}

t = t->children[index];

}

// First ilerated all insertion a,p,p,l,e then make e bool as " is word end"

if (t->isWordEnd == true) {

cout << "\n Word Already Exist \n\n";

return false;

}

t->isWordEnd = true;

t->meaning = meaning;

return true;

}

//search

bool search(string word) {

trieNode\* t = root;

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

{

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

if (!t->children[key])

{

cout << "Word not found!\n";

return false;

}

t = t->children[key];

}

if (t->isWordEnd == true)

{

cout << "Word : " << word << " | Meaning : " << t->meaning << endl;

}

return t->isWordEnd;

}

//update

void update(string word)

{

trieNode\* t = root;

//search the word present or not

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

{

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

if (!t->children[key])

{

cout << "\nWord not found!\n\n";

return;

}

t = t->children[key];

}

// word not found

if (t->isWordEnd == false)

{

cout << "\nWord not found!\n\n";

return;

}//if word found

else

{

cout << "Word : " << word << " | Meaning : " << t->meaning << endl;

//upadating the previous meaning

cout << "Do! You Wand to Update?\nPress 1 to Update\nPress 2 to Quit\nChoice : ";

string choice;

cin >> choice;

if (choice == "1")

{

string newMeaning;

cout << "Enter New Meaing : "; cin >> newMeaning;

t->meaning = newMeaning;

//Update Meaning in file

ifstream read("Dictionary.txt");

ofstream open("temp.txt");

string w, m;

cout << "Updating....\n";

while (!read.eof()) {

read >> w;

read >> m;

if (!read.eof()) {

if (w == word)

{

open << w << "\t" << newMeaning << endl;

}

else

{

open << w << "\t" << m << endl;

}

}

}

read.close();

open.close();

remove("Dictionary.txt");

rename("temp.txt", "Dictionary.txt");

cout << "Meaning Updated SuccessFully.\n";

}

else if (choice == "2")

{

return;

}

}

}

//delete the word

void deleteWord(string word) {

trieNode\* t = root;

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

{

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

if (!t->children[key])

{

cout << "\nWord not found!\n\n";

return;

}

t = t->children[key];

}

// word not found

if (t->isWordEnd == false)

{

cout << "\nWord not found!\n\n";

return;

}//if word found

else {

cout << "Word : " << word << " | Meaning : " << t->meaning << endl;

//upadating the previous meaning

cout << "Do! You Wand to Delete?\nPress 1 to Delete\nPress 2 to Quit\nChoice : ";

string choice; cin >> choice;

if (choice == "1")

{

t->isWordEnd = false;

//Deleting the word in Dictionary File

ifstream read("Dictionary.txt");

ofstream open("temp.txt");

string w, m;

cout << "Deleing....\n";

while (!read.eof()) {

read >> w;

read >> m;

if (w != word)

{

open << w << "\t" << m << endl;

}

}

read.close();

open.close();

remove("Dictionary.txt");

rename("temp.txt", "Dictionary.txt");

cout << "Word Deleted SuccessFully\n";

}

else if (choice == "2")

{

return;

}

}

}

void suggestWordsHelper(trieNode\* node, string prefix, string suggestions[MAX\_SUGGESTIONS], int& count) {

if (node == nullptr || count >= MAX\_SUGGESTIONS) {

return;

}

if (node->isWordEnd) {

suggestions[count] = prefix;

count++;

}

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

if (node->children[i] != nullptr) {

suggestWordsHelper(node->children[i], prefix + char('a' + i), suggestions, count);

if (count >= MAX\_SUGGESTIONS) {

return;

}

}

}

}

void suggestWords(string prefix) {

trieNode\* node = root;

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

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

if (node->children[index] == nullptr) {

cout << "No suggestions for prefix " << prefix << endl;

return;

}

node = node->children[index];

}

string suggestions[MAX\_SUGGESTIONS];

int count = 0;

suggestWordsHelper(node, prefix, suggestions, count);

cout << "Suggestions for prefix " << prefix << ": \n\n";

for (int i = 0; i < count && i < MAX\_SUGGESTIONS; i++) {

cout <<"\t "<< i+1<<" ) " << suggestions[i] << " \n";

}

cout << endl;

}

~Trie() {

// Call the recursive helper function to delete nodes

destroyTrie(root);

}

void destroyTrie(trieNode\* node) {

if (node == nullptr) {

return;

}

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

destroyTrie(node->children[i]);

}

delete node;

}

void load() {

string word, meaning;

system("cls");

cout << "\t\t|--------------------------------|\n";

cout << "\t\t| Dictionary is Loading...... |\n";

cout << "\t\t|--------------------------------|\n\n";

//open the file

ifstream read("Dictionary.txt");

//insert the dictionary in trie

while (!read.eof())

{

read >> word;

read >> meaning;

//checking for empty space

// we only insert all non space words and meanings

if (word.length() != 0 && meaning.length() != 0) {

insert(word, meaning);

cout << word << " " << meaning << "\n";

}

}

system("cls");

//close the file

read.close();

}

};

void menu() {

cout << "\n \n \n \n \n";

cout << " \t \t |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n";

cout.flush();

Sleep(500);

cout << " \t \t |\t \t \t \t \t \t \t|\n";

cout << " \t \t |\t WELCOME FAST Dictinory System!!!!\t \t|\n";

cout.flush();

Sleep(500);

cout << " \t \t |\t HOPE YOU will our your Work \t \t\t|\n";

cout.flush();

Sleep(600);

cout << "\t \t |\t Special Thank to Sir shahbaz Ayyaz,Dr. Hashim |\n";

cout.flush();

Sleep(600);

cout << " \t \t |\t ..........................................\t| \n";

cout << " \t \t |\t \t \t \t \t \t \t|\n ";

cout << " \t \t |\t Name : Ahmad Raza ( 22F-3325 )\t \t\t|\n";

cout << " \t \t |\t Name : Shehryar Khan ( 22F-3257 )\t \t|\n";

// cout << "\t \t \t \t | \n";

cout << " \t \t |\t \t \t \t \t \t \t|\n ";

cout.flush();

Sleep(700);

cout << " \t \t |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n";

cout << "\n\n\n\n\n\n\n\n\n\n";

}

void boolnotfound() {

cout << "\n\n\n\n\n";

system("pause");

system("cls");

}

int main() {

Trie o;

//o.load();

bool loadstart = false;

// menu();

system("color 0b");

system("pause");

//o.load();

system("cls");

menuu:

while (true)

{

cout << "\n\t\t\t |----------------------------------|\n";

cout << "\t\t\t | Dictionary Loaded SuccesFully |\n";

cout << "\t\t\t |\t\t\t\t |\n";

cout << " \t \t |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n";

cout << " \t \t | Enter 1-Load File\t\t\t|\n";

cout << " \t \t | Enter 2-Search\t\t\t |\n";

cout << " \t \t | Enter 3-Update\t\t\t |\n";

cout << " \t \t | Enter 4-Add Word\t\t |\n";

cout << " \t \t | Enter 5-Delete word\t\t\t|\n";

cout << " \t \t | Enter 6-Suggestion\t\t\t|\n";

cout << " \t \t | Enter 7-Exit \t\t |\n";

cout << " \t \t |\t\t\t\t \t\t\t|\n";

cout << " \t \t |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n";

string choice;

cout << " \n\n \t\t -> Enter Choice : ";

cin >> choice;

if (choice == "1") {

o.load();

loadstart=true;

}

else if (choice == "2") {

if (loadstart == false) {

cout << "\n \n \t \t -> File is not loaded \n";

boolnotfound();

goto menuu;

}

string w;

cout << "Enter Word : ";

cin >> w;

cout << "\n\n";

if (!o.search(w)) {

cout << "Word not found\n";

}

cout << "\n\n\n\n\n";

system("pause");

system("cls");

}

else if (choice == "3")

{

if (loadstart == false) {

cout << "\n \n \t \t -> File is not loaded \n";

boolnotfound();

goto menuu;

}

string w; cout << "Enter Word You want to Update: "; cin >> w;

o.update(w);

cout << "\n\n\n\n\n";

system("pause");

system("cls");

}

else if (choice == "4")

{

if (loadstart == false) {

cout << "\n \n \t \t -> File is not loaded \n";

boolnotfound();

goto menuu;

}

string w; cout << "Enter Word You want to Delete : "; cin >> w;

o.deleteWord(w);

cout << "\n\n\n\n\n";

system("pause");

system("cls");

}

else if (choice == "5")

{

if (loadstart == false) {

cout << "\n \n \t \t -> File is not loaded \n";

boolnotfound();

goto menuu;

}

string word, meaning;

cout << " Enter Word toy want to Add : "; cin >> word;

cout << " Enter Meaning : "; cin >> meaning;

if (o.insert(word, meaning)) {

// now insert the word you enter in the dictonary file

ofstream open("Dictionary.txt", ios::app);

open << "\n" << word << "\t" << meaning << endl;

open.close();

cout << "Word Inserted Successfully\n";

cout << "\n\n\n\n\n";

system("pause");

system("cls");

}

}

else if (choice == "6")

{

if (loadstart == false) {

cout << "\n \n \t \t -> File is not loaded \n";

boolnotfound();

goto menuu;

}

string wordss;

cout << " Enter WORD you want to get suggestion : ";

cin >> wordss;

cout << "\n";

o.suggestWords(wordss);

cout << "\n\n\n\n\n";

system("pause");

system("cls");

}

else if (choice == "7")

{cout << " \t \t | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| \n";

cout << "\n\n \t\t \*\*\* Thank you for your Time \*\*\*";

cout << "\n\n\n\n";

break;

}

}

return 0;

}