#include <iostream>

#include <fstream>

#include <string.h>

#include <deque>

#include <queue>

#include <list>

#include <stack>

#include <dirent.h>

#include <conio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

using namespace std;

struct details{ // structure to be there in the list result

string name;

int rank;

};

class file\_handle{

deque<string> v; // deque for the list of html files in the webserver folder

deque<string>::iterator it; // common iterator for the type deques

deque<string> search; // deque having the search terms, which are filtered

list<string> query;

list<list<struct details> > result; // 2D list for having the file searches

list<list<struct details> > type\_file; // 2D list for having the tags

list<list<string> > ans; // 2D list for having the answer

char s[100]; // getting the raw search input

string separated[100];

public:

file\_handle(){}

int file();

void show\_files\_to\_search(deque<string>);

list<list<string> > get\_search(string);

void search\_files();

void type\_file\_search(string);

void type\_word\_search();

list<details> read\_file(string);

void display\_2dlist();

void compare\_and\_generate();

void final\_result();

void search\_dictionary(string);

};

// for Vanethi

// Function to determine the type of the file

string filetype(string filename){

int i = 0,num,j;

string type = "",name="";

for(i=0;filename[i]!='\0';i++);

num=i;

for(i=num-1;i!=0;i--){

if(filename[i]=='.'){

for(j=0;type[j]!='\0';j++);

for(int k=0;k<j;k++){

name+= type[j-k-1];

}

return name;

}

else

type += filename[i];

}

}

// for vanethi

// Function to put the file names in a deque

int file\_handle::file(){

ifstream fin;

string dir, filepath ;

int num;

DIR \*dp,\*dr;

struct stat filestat;

struct dirent \*dirp;

int n=0;

dir = "webserver";

if((dp = opendir(dir.c\_str())) == NULL) {

cout << "Error(" << errno << ") opening " << dir << endl;

return errno;

}

else

while ((dirp = readdir( dp )))

{

filepath = dir + "/" + dirp->d\_name;

if(filetype(dirp->d\_name)=="html" || filetype(dirp->d\_name)=="htm"){

//cout<<"Name : "<<dirp->d\_name<<endl;

v.push\_back(dirp->d\_name);

}

}

closedir(dp);

}

void file\_handle::display\_2dlist(){

list<list<struct details> >::iterator it;

list<struct details>::iterator it2;

int i=0;

for(it=result.begin();it!=result.end();it++){

cout<<i<<ends;

i++;

for(it2 = it->begin(); it2!=it->end(); it2++){

cout<<it2->name<<ends;

}

cout<<endl<<endl;

}

}

void file\_handle::show\_files\_to\_search(deque<string> vec){

for(it=vec.begin();it!=vec.end();it++){

cout<<\*it<<endl;

}

}

// for Karthik

// Funtion to check the term is need for the searching purpose

bool filter\_term(string term){

// the a an in is of at , " .

string list[100]={"the","a","an","am","in","is","of","at","on","'","\"","."};

for(int i=0;list[i]!="";i++){

if(list[i]==term){

return false;

}

}

return true;

}

// for Karthik

// Funtion to check the term is need for the searching purpose

bool filter\_character(char c){

// : \_ - + = ,

char list[100]={':','\_','-','+',' ','.','=',','};

for(int i=0;list[i];i++){

if(list[i]==c)

return false;

}

return true;

}

// for karthik

// Function to store the search term separately in a deque

list<list<string> > file\_handle::get\_search(string s){

string term;

//cout<<"Enter the search term : ";

//cin.getline(s,100);

//cin>>s;

//cout<<s<<endl;

for(int i=0;s[i]!='\0';i++){

if(s[i]==' ' || s[i]=='+'){

if(!filter\_term(term));

else

if(s[i]=='+') search.push\_back(term+"+");

else search.push\_back(term);

term = "";

}

else{

term += s[i];

}

}

search.push\_back(term);

//show\_files\_to\_search(search);

search\_files();

return ans;

}

// for Sekharan

// Function used mainly for the seeing whether the given search term is a special search or the ordinary search

// If the special search, the function will return the appropriate strings used for the search...

string separate(string term,string type){

int len;

string t="";

bool found=false;

for(len=0;term[len]!='\0';len++);

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

if(term[i] == ':'){

if(type=="main") return t;

t="";

found = true;

}

else{

t += term[i];

//cout<<t<<endl;

}

}

if(found)

if(type=="type") return t;

else;

else

return "";

}

// for Sekharan

// Searching done by having the datas from the deque "search"

void file\_handle::search\_files(){

// getting the type of the search as either a dictionary word or type of the searching document (as docx,pdf,pptx...)

string prefer = separate(search.front(),"main"),word=separate(search.front(),"type");

if(search.front()=="Define:" || search.front()=="define:" || prefer=="define" || prefer=="Define"){

// we have to search for the dictionary word

//cout<<"Prefer Dictionary...\n";

//cout<<"Word is "<<word<<endl;

search.pop\_front();

search.push\_front(word);

search\_dictionary(word);

}

else if(search.front()=="Filetype:" || search.front()=="filetype:" || prefer=="filetype" || prefer=="Filetype" ){

search.pop\_front();

type\_file\_search(word);

// we will use struct details in the 2D list type\_file

}

type\_word\_search();

// we will use struct details in the 2D list result

// now start searching the word search

compare\_and\_generate();

}

bool compare(list<struct details> l1, list<struct details> l2){

// get the two ranks

list<struct details>::iterator it;

int temp1=0,temp2=0;

it = l1.end();

it--;

temp1 = it->rank;

it = l2.end();

it--;

temp2 = it->rank;

if(temp2<temp1) return true;

return false;

}

void file\_handle::search\_dictionary(string word){

ifstream fin("dictionary.txt");

if(fin && !fin.eof() ){

bool found = false,found2 = false;

char line[200];

string word\_meaning="",meaning="";

while(!found && !fin.eof() ){

fin.getline(line,200);

int i=0;

word\_meaning="";

while(!found2){

if(word[i]>=65 && word[i]<=90) word[i] = char(int(word[i])+32);

if(line[i]>=65 && line[i]<=90) line[i] = char(int(line[i])+32);

if(line[i] == word[i]){

word\_meaning += word[i];

}

else if(int(line[i]) == 45){

// extraction starts here

meaning = "";

i++;

while(line[i]!='.'){

meaning += line[i];

i++;

}

// insert in the result

list<list<string> >::iterator it;

list<string>::iterator it2;

list<string> temp;

temp.push\_back("define");

temp.push\_back(word\_meaning);

temp.push\_back(meaning);

ans.push\_back(temp);

fin.close();

for(it=ans.begin();it!=ans.end();it++){

for(it2=it->begin();it2!=it->end();it2++){

//cout<<\*it2<<endl;

}

//cout<<endl<<endl;

}

//cout<<"Over in meaning.\n";

return;

found2 = true;

found = true;

}

else{

found2 = true;

}

i++;

}

found2 = false;

}

//cout<<"Your search for "<<word<<" did not match any words in dictionary.\n";

string error = "Your Search for "+word+" did not match any words.";

list<list<string> >::iterator it;

list<string>::iterator it2;

list<string> temp;

temp.push\_back("define");

temp.push\_back("error");

temp.push\_back(error);

ans.push\_back(temp);

}

else

cout<<"Error in opening the dictionary..\n";

}

void file\_handle::type\_file\_search(string type){

ifstream fin;

string dir, filepath ;

//cout<<"In the file search function.\n";

int num;

DIR \*dp,\*dr;

struct stat filestat;

struct dirent \*dirp;

int n=0;

list<string> file\_type;

dir = "webserver";

if((dp = opendir(dir.c\_str())) == NULL) {

cout << "Error(" << errno << ") opening " << dir << endl;

//return errno;

}

else{

list<list<struct details> >::iterator it;

list<struct details>::iterator it2;

while ((dirp = readdir( dp )))

{

filepath = dir + "/" + dirp->d\_name;

string file\_name="";

string type\_of\_file=filetype(dirp->d\_name);

string text2="";

for(int i=0; type\_of\_file[i]!='\0';i++)

if(type\_of\_file[i]>=97 && type\_of\_file[i]<=122) text2 += type\_of\_file[i];

else text2 += char(int(type\_of\_file[i])+32);

if(text2==type){

list<struct details> l;

//cout<<"Name : "<<dirp->d\_name<<endl;

//cout<<dirp->d\_name<<endl;

file\_name = dirp->d\_name;

details d;

d.name = dirp->d\_name;

d.rank = 0;

l.push\_back(d);

int i=0;

string text=""; bool cont=false;

// file name separation

do{

if(filter\_character(file\_name[i])){

// okay to append to update

if(file\_name[i]>=97 && file\_name[i]<=122) text += file\_name[i];

else text += char(int(file\_name[i])+32);

}

else{

// one word is over

details d;

d.name = text;

d.rank = 0;

l.push\_back(d);

text="";

if(file\_name[i]=='.')

cont = true;

}

i++;

}while(!cont);

// file name separation over

type\_file.push\_back(l);

}

}

deque<string>::iterator se;

string text="";

int hit=0;

for(it=type\_file.begin();it!=type\_file.end();it++){

for(se=search.begin();se!=search.end();se++){

for(it2=it->begin();it2!=it->end();it2++){

for(int i=0; it2->name[i]; i++)

if(it2->name[i]>=97 && it2->name[i]<=122) text += it2->name[i];

else text+= char(int(it2->name[i])+32);

if(it2->name == \*se)

hit++;

}

}

it2 = it->end();

--it2;

it2->rank = hit;

hit = 0;

}

// Generate the result by comparing

for(it=type\_file.begin();it!=type\_file.end();it++){

it2 = it->end();

--it2;

// we have it2->rank

if(it2->rank!=0);

else{

it=type\_file.erase(it);

--it;

}

} // now type\_file list won't have the 0 ranked entries

// now sort according to the ranks

type\_file.sort(compare); // working correctly

if(type\_file.empty()){

//cout<<"Your search for "<<type<<" kind of files is not found in the server for your query.\n";

string error = "Your search for "+type+" kind of files is not found.";

list<string> temp;

temp.push\_back("type");

temp.push\_back("error");

temp.push\_back(error);

ans.push\_back(temp);

}

//cout<<"After Comparison...\n"<<endl;

for(it=type\_file.begin();it!=type\_file.end();it++){

// for(it2=it->begin();it2!=it->end();it2++){

//cout<<it2->name<<ends<<it2->rank<<endl;

it2 = it->begin();

list<string> l;

l.push\_back("type");

l.push\_back(type);

l.push\_back(it2->name);

ans.push\_back(l);

//}

//cout<<endl;

}

}

closedir(dp);

}

// Searching the html files and giving the priority

// for Preethi

// all the html files will be in the folder called "webserver"

void file\_handle:: type\_word\_search(){

list<list<struct details> >::iterator it; // iterator for the 2D List

list<details>::iterator it1; // iterator for the 1D List

deque<string>::iterator d; // iterator for the Deque(Which has the list of html files in the webserver folder)

//cout<<"In the word search function : ";

cout<<"Loading...\n";

for(d=v.begin();d!=v.end();d++){

list<details> va;

// reading from the html files and saving as one dimension in the list

va = read\_file(\*d); // use read\_file(\*d); to read the html files in the directory

result.push\_back(va);

}

//cout<<"Tags are taken successfully...\n";

//system("pause");

};

// for vanethi

// for filtering the tag words in the html file

string filter\_tag(string text){

//if(string == "") return "";

char content[10] = {'c','o','n','t','e','n','t','=','\"'};

char restrict[50] = {',','.',':',';','[',']','{','}','(',')','<','>','?','/','\"','\'','!','@','#','$','%','^','&','\*','(',')','-','=','+','\_','/','\*','|'};

// remove commas,full stops like that

bool cont=true;

int i;

for(i=0; (i<9) && cont; i++){

if(text[i] == content[i]);

else cont=false;

}

if(cont){

//i--;

string ret\_text="";

for(int j=0; text[i]!='\0'; j++){

if(text[i]!=' ') ret\_text += text[i];

i++;

//cout<<text[i]<<ends<<ret\_text[j]<<endl;

}

//cout<<"ret\_text is "<<ret\_text<<endl;

return ret\_text;

}

else

for(i=0;text[i]!='\0';i++){

for(int j=0; restrict[j]!='\0';j++){

if(text[i] == restrict[j]){

text[i]='\0';

}

}

}

return text;

}

bool check(string text){

for(int i=0;i<100 && text[i]!='\0'; i++){

if(text[i]=='\"')

return false;

}

return true;

}

bool comp(struct details l1, struct details l2){

if(l1.rank > l2.rank) return true;

return false;

}

// for Preethi

list<struct details> file\_handle::read\_file(string file\_name){

list<struct details> l;

string dir, filepath ;

int num;

DIR \*dp,\*dr;

struct stat filestat;

struct dirent \*dirp;

int n=0;

dir = "webserver";

filepath = dir +"\\"+ file\_name;

if((dp = opendir(dir.c\_str())) == NULL) {

cout << "Error(" << errno << ") opening " << dir << endl;

//return errno;

}

else

while ((dirp = readdir( dp )))

{

if(dirp->d\_name == file\_name)

{

//cout<<"File is there "<<file\_name<<"\n";

char c,C;

char name[100]={'\0'},path[200]={'\0'};

for(int i=0;file\_name[i]!='\0';i++){

name[i] = file\_name[i];

}

strcat(path,"webserver/");

strcat(path,name);

ifstream fin(path);

string s;

bool reached\_meta=false,reached\_content = false;

bool finished\_reading = false;

//if(fin) cout<<"success in open\n";

//else cout<<"Fail in open\n";

int i=0;

string text=""; bool cont=false;

do{

if(filter\_character(file\_name[i])){

// okay to append to update

if(file\_name[i]>=97 && file\_name[i]<=122) text += file\_name[i];

else text += char(int(file\_name[i])+32);

}

else{

// one word is over

details d;

d.name = text;

d.rank = 0;

l.push\_back(d);

text="";

if(file\_name[i]=='.')

cont = true;

}

i++;

}while(!cont);

// check for name="Description"

string tag\_name="name=\"description\"", content="content=\"";

int for\_name=0, for\_des = 0, for\_con=0;

bool reached\_name = false, reached\_des = false;

while(!reached\_content && !fin.eof()){

fin>>c;

if(int(c)>=97 && int(c)<=122)

C=char(int(c)-32);

else C=char(int(c)+32);

if(!reached\_name && (c=='n' || c=='N'))

for(int i=0;i<=17 && !fin.eof() ;i++){

if(int(c)>=97 && int(c)<=122)

C=char(int(c)-32);

else C=char(int(c)+32);

if(c==tag\_name[i] || C==tag\_name[i]);

else i=0;

if(i==17){

reached\_name = true;

//system("pause");

}

fin>>c;

}

if(!reached\_content && reached\_name){

//system("pause");

if(c==content[for\_con] || C==content[for\_con]){

if(c=='\"')

reached\_content=true;

for\_con++;

}

}// reached <meta name="description" content="

} // while ends here

// and we reached <meta name="description" content="

// extraction starts here

char punctuations[100] = {' ',',','.',':'};

while(reached\_content && !finished\_reading && !fin.eof() ){

fin>>text;

string text2="";

if(check(text)){

text = filter\_tag(text);

if(filter\_term(text)){

details d;

for(int i=0; text[i]; i++)

if(text[i]>=97 && text[i]<=122) text2 += text[i];

else text2 += char(int(text[i])+32);

d.name = text2;

d.rank = 0;

l.push\_back(d);

text2 = "";

}

//cout<<"One of the tag is "<<text<<endl;

}

else{

// last word

string last="";

text2="";

for(int i=0;i<100 && text[i]!='\"'; i++)

last += text[i];

details d;

for(int i=0; last[i]; i++)

if(last[i]>=97 && last[i]<=122) text2 += last[i];

else text2 += char(int(last[i])+32);

d.name = text2;

d.rank = 0;

l.push\_back(d);

finished\_reading = true;

}

//system("pause");

}

// close the file

fin.close();

}

}

closedir(dp);

return l;

}

void file\_handle::compare\_and\_generate(){

list<list<struct details> >::iterator it;

list<struct details>::iterator it2;

deque<string>::iterator se;

int hit=0; string text="";

for(it=result.begin();it!=result.end();it++){

for(se=search.begin();se!=search.end();se++){

for(it2=it->begin();it2!=it->end();it2++){

if(it2->name == \*se)

hit++;

}

}

it2 = it->end();

--it2;

it2->rank = hit;

hit = 0;

}

list<struct details> temp;

for(it=result.begin(),se=v.begin();it!=result.end() && se!=v.end();it++,se++){

it2 = it->end();

--it2;

if(it2->rank){

//cout<<\*se<<ends<<it2->rank<<endl;

details d;

d.name = \*se;

d.rank = it2->rank;

temp.push\_back(d);

}

}

//cout<<"After sorting";

temp.sort(comp);

if(temp.empty()){

//cout<<"Your query did not match any documents.\n";

string error = "Your query did not match any documents.";

list<string> temp;

temp.push\_back("query");

temp.push\_back("error");

temp.push\_back(error);

ans.push\_back(temp);

}

for(it2=temp.begin();it2!=temp.end();it2++){

//cout<<it2->name<<ends<<it2->rank<<endl;

list<string> t;

t.push\_back("query");

t.push\_back(it2->name);

ans.push\_back(t);

}

cout<<"Result for your query : \n";

//final\_result();

}

void file\_handle::final\_result(){

// print

list<list<string> >::iterator it;

list<string>::iterator it2;

for(it=ans.begin();it!=ans.end();it++){

//for(it2 = it->begin();it2!=it->end();it2++){

it2 = it->begin();

if(\*it2 == "define"){

it2++;

if(\*it2 == "error"){

it2++;

cout<<\*it2<<endl;

}

else{

cout<<"Definition : \n\t"<<\*it2<<endl;

}

}

else if(\*it2 == "type"){

it2++;

if(\*it2 == "error"){

it2++;

cout<<\*it2<<endl;

}

else{

cout<<"["<<\*it2<<"] ";

it2++;

cout<<\*it2<<endl;

}

}

else{

it2++;

if(\*it2 == "error"){

it2++;

cout<<\*it2<<endl;

}

else{

cout<<\*it2<<endl;

}

}

//}

cout<<endl;

}

}