**Project 3**

**FILE DIRECTORY**

**DATA STRUCTURE USED: TREES**

**ROLL NOS- 2,5,6**

#include<iostream>

#include<stdio.h>

#include<malloc.h>

#include<cstring>

#include<string>

using namespace std;

#define MAX 50

struct tree

{

string str;

int n;

int flag; // -1 for drive 0 for files and 1 for folders

struct tree\*\* ptr;

};

struct stack

{

tree\* a[MAX];

int top;

}\*s;

typedef struct tree tree;

tree\* create();

void add\_folder(tree\*,string);

tree\* traverse\_folder(tree\*,string,int);

tree\* traverse\_file(tree\*,string,int);

void add\_file(tree\*,string);

void delete\_file(tree\*,string);

void delete\_folder(tree\*,string);

void print\_files(tree\*);

void print\_folders(tree\*);

void print(tree\*);

void push(tree\*);

tree\* pop();

tree\* create()

{

tree\* root=new tree();

cout<<"\nenter drive name\n";

cin>>root->str;

root->flag=-1;

root->n=0;

root->ptr=NULL;

int choice;

do

{

cout<<"\nenter 1 to add files and 2 for folders\n";

cin>>choice;

}while( choice >1 && choice <2);

if(choice == 2)

{

add\_folder(root,"NULL");

}

else if(choice ==1)

{

cout<<"\nenter no of files in "<<root->str;

cin>>root->n;

root->ptr=new tree\*[root->n];

for(int i=0;i<root->n;i++)

{

root->ptr[i]=new tree;

root->ptr[i]->flag=0;

cout<<"\nenter the name of file\n";

cin>>root->ptr[i]->str;

root->ptr[i]->ptr=NULL;

}

}

return root;

}

tree\* traverse\_folder(tree\* root,string folder,int return\_parent\_flag )

{

s->top=-1;

if(root->str==folder)

return root;

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->str == folder )

{

if(return\_parent\_flag==0)

return root->ptr[i];

else

return root;

}

else

{

if(root->ptr[i]!=NULL)

{

push(root->ptr[i]);

}

}

}

while(s->top!=-1)

{

root=pop();

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->str == folder)

{

if(return\_parent\_flag==0)

return root->ptr[i];

else

return root;

}

else

{

if(root->ptr[i]!=NULL)

{

push(root->ptr[i]);

}

}

}

}

return NULL;

}

tree\* traverse\_file(tree\* root,string file,int return\_parent\_flag)

{

s->top=-1;

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->flag==0)

{

if(root->ptr[i]->str == file )

{

if(return\_parent\_flag==0)

return root->ptr[i];

else

return root;

}

}

else

{

if(root->ptr[i]!=NULL)

{

push(root->ptr[i]);

}

}

}

while(s->top!=-1)

{

root=pop();

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->flag==0)

{

if(root->ptr[i]->str == file )

{

if(return\_parent\_flag==0)

return root->ptr[i];

else

return root;

}

}

else

{

if(root->ptr[i]!=NULL)

{

push(root->ptr[i]);

}

}

}

}

return NULL;

}

void add\_folder(tree \*root, string folder)

{

root->n++;

root->ptr=(tree\*\*)realloc(root->ptr,root->n\*sizeof(tree\*));

root->ptr[(root->n)-1]=new tree;

if(folder == "NULL" )

{

cout<<"\nenter the name of folder in "<<root->str<<endl;

cin>>root->ptr[root->n-1]->str;

}

else

{

root->ptr[root->n-1]->str=folder;

}

root->ptr[root->n-1]->flag=1;

int choice;

do

{

cout<<"\nenter 1 to add files and 2 for folders\n";

cin>>choice;

}while(choice >1 && choice <2);

if(choice == 2)

{

add\_folder(root->ptr[root->n-1],"NULL");

}

int temp;

int oldn=root->ptr[root->n-1]->n;

cout<<"\nenter no of files in "<<root->ptr[root->n-1]->str<<endl;

cin>>temp;

root->ptr[root->n-1]->n+=temp;

root->ptr[root->n-1]->ptr=(tree\*\*)realloc(root->ptr[root->n-1]->ptr ,(root->ptr[root->n-1]->n)\*sizeof(tree\*) );

for(int j=oldn;j<root->ptr[root->n-1]->n;j++)

{

root->ptr[root->n-1]->ptr[j]=new tree;

root->ptr[root->n-1]->ptr[j]->flag=0;

cout<<"\nenter name of new file\n";

cin>>root->ptr[root->n-1]->ptr[j]->str;

root->ptr[root->n-1]->ptr[j]->ptr=NULL;

root->ptr[root->n-1]->ptr[j]->n=0;

}

}

void add\_file(tree \*temp,string file)

{

temp->n++;

temp->ptr=(tree\*\*)realloc(temp->ptr,temp->n\*sizeof(tree\*));

temp->ptr[temp->n-1]=new tree;

if(file == "NULL")

{

cout<<"\nenter the name of file\n";

cin>>temp->ptr[temp->n-1]->str;

}

else

temp->ptr[temp->n-1]->str=file;

temp->ptr[temp->n-1]->n=0;

temp->ptr[temp->n-1]->ptr=NULL;

}

int static l=0;

void delete\_folder(tree\* root,string folder)

{

tree \*t,\*t1;

int i;

if(l==0)

{

for( i=0;i<root->n;i++)

{

if(root->ptr[i]->str==folder)

break;

}

t=root->ptr[i];

for(int j=i+1;j<root->n;j++)

{

root->ptr[j-1]=root->ptr[j];

}

root->n--;

root->ptr=(tree\*\*)realloc(root->ptr,(root->n)\*(sizeof(tree\*)));

l=1;

root=t;

}

for( i=0;i<root->n;i++)

{

if(root->ptr[i]->flag==1)

{

delete\_folder(root->ptr[i],folder);

}

else if(root->ptr[i]->flag==0)

{

t1=root;

delete root->ptr[i];

for(int j=i+1;j<root->n;j++)

{

root->ptr[j-1]=root->ptr[j];

}

root->n--;

root->ptr=(tree\*\*)realloc(root->ptr,(root->n)\*(sizeof(tree\*)));

}

}

delete t1;

}

void delete\_file(tree\* root,string file)

{

int i;

tree\* t;

for( i=0;i<root->n;i++)

{

if(root->ptr[i]->str==file)

break;

}

t=root->ptr[i];

for(int j=i+1;j<root->n;j++)

{

root->ptr[j-1]=root->ptr[j];

}

root->n--;

root->ptr=(tree\*\*)realloc(root->ptr,(root->n)\*(sizeof(tree\*)));

}

void print\_folders(tree\* root)

{

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->flag==1)

{

cout<<root->ptr[i]->str<<endl;

print\_folders(root->ptr[i]);

}

}

}

void print\_files(tree \*root)

{

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->flag==0)

{

cout<<root->ptr[i]->str<<" ";

}

print\_files(root->ptr[i]);

}

}

void print(tree\* root)

{

for(int i=0;i<root->n;i++)

{

if(root->ptr[i]->flag==1)

{

cout<<root->str<<" has "<<root->ptr[i]->str<<" folder"<<endl;

print(root->ptr[i]);

cout<<"\t"<<root->ptr[i]->str<<" has ";

for(int j=0;j<root->ptr[i]->n;j++)

{

if(root->ptr[i]->ptr[j]->flag==0)

cout<<root->ptr[i]->ptr[j]->str<<" ";

}

cout<<" files "<<endl;

}

else if(root->flag==-1)

{

cout<<root->str<<" has "<<root->ptr[i]->str<<" file"<<endl;

}

}

}

void push(tree\* c)

{

if(s->top==MAX-1)

cout<<"Stack overflow";

else

{

s->top++;

s->a[s->top]=c;

}

}

tree\* pop()

{

tree\* c;

if(s->top==-1)

printf("Stack underflow");

else

{c=s->a[s->top];

s->top--;}

return c;

}

int main()

{

s=(struct stack\*)malloc(sizeof(struct stack));

s->top=-1;

string file,folder;

string destination\_folder;

tree\* root=create();

tree \*folderp,\*filep;

int ch;

while(ch!=10)

{

cout<<"\n\nenter the number to perform the corresponding operations\n1.search folder 2.search file 3.add folder 4.add file 5.delete folder 6.delete file 7.print folders 8.print files 9.print whole directory\n";

cin>>ch;

switch(ch)

{

case 1:int c;

do

{

cout<<"\nenter folder name to be searched\n";

cin>>folder;

folderp=traverse\_folder(root,folder,0);

if(folderp==NULL)

{

cout<<"folder not found\n";

cout<<"enter 1 to continue and 0 to exit\n";

cin>>c;

}

else

{

cout<<folderp->str<<" found"<<endl;

c=0;

}

}while(c==1);

break;

case 2:int c1;

do

{

cout<<"\nenter file name to be searched\n";

cin>>file;

filep=traverse\_file(root,file,0);

if(filep==NULL)

{

cout<<"file not found\n";

cout<<"enter 1 to continue and 0 to exit\n";

cin>>c1;

}

else

{

cout<<filep->str<<" found"<<endl;

c1=0;

}

}while(c1 == 1);

break;

case 3:

do

{

cout<<"\nenter in which folder/drive do you want to add folder\n";

cin>>destination\_folder;

folderp=traverse\_folder(root,destination\_folder,0);

if(folderp==NULL)

{

cout<<"\nenter valid folder name\n";

cout<<"enter 1 to continue and 0 to exit\n";

cin>>c1;

}

else

{

cout<<"\nenter the name of new folder\n";

cin>>folder;

add\_folder(folderp,folder);

cout<<"\nnew folder has been added successfully!\n";

c=0;

}

}while(c==1);

break;

case 4:do

{

cout<<"\nenter folder/drive name where new file is to be added\n";

cin>>destination\_folder;

folderp=traverse\_folder(root,destination\_folder,0);

if(folderp==NULL)

{

cout<<"\nenter valid folder name\n";

cout<<"enter 1 to continue and 0 to exit\n";

cin>>c1;

}

else

{

cout<<"enter the name of new file\n";

cin>>file;

add\_file(folderp,file);

cout<<"new file has been added successfully!\n";

}

}while(c1==1);

break;

case 5:

do

{

l=0;

cout<<"\nenter the name of the folder to be deleted\n";

cin>>folder;

folderp=traverse\_folder(root,folder,1);

if(folderp==NULL)

{

cout<<"\nenter valid folder name\n";

cout<<"enter 1 to continue and 0 to exit\n";

cin>>c;

}

else

{

delete\_folder(folderp,folder);

cout<<"folder has been deleted successfully!\n";

}

}while(c==1);

break;

case 6:do

{

cout<<"\nenter the name of the file to be deleted\n";

cin>>file;

folderp=traverse\_folder(root,file,1);

if(folderp==NULL)

{

cout<<"\nenter valid folder name\n";

cout<<"enter 1 to continue and 0 to exit\n";

cin>>c;

}

else

{

delete\_file(folderp,file);

cout<<"file has been deleted successfully!\n";

}

}while(c==1);

break;

case 7:cout<<"\nlists of folders in "<<root->str<<" is\n";

print\_folders(root);

break;

case 8:

cout<<"\nlists of files in directory are\n";

print\_files(root);

break;

case 9:cout<<"\nentire file is \n";

print(root);break;

default:ch=10;

}

}

return 0;

}











