/\*

***Name* :-** Abrar Mahedavi

***Roll No.* :-** 21035

***Title* :-** Wrtite a Program using object oriented programming using c++ to create a binarytree if inorder and preorder or inorder and post order any traversal is given.

/\*

***Program :***

#include<iostream>

#include<string.h>

using namespace std;

class node //Declaration of Class

{

public:

node\* Left;

node\* Right;

char data;

};

class tree //Declaration of Class

{

private:

node\* root;

public:

char In[20];

char Pre[20];

char Post[20];

void Inorder(); //Prototype for Inorder function

void Inorder1(node\* p); //Prototype for Inorder1 function

void preorder(); //Prototype for preorder function

void inorder(); //Prototype for inorder function

void postorder(); //Prototype for postorder function

void Preorder(); //Prototype for Preorder function

void Postorder(); //Prototype for Postorder function

void Create(); //Prototype for Create function

void Create1(); //Prototype for Create1 function

void Preorder1(node\*); //Prototype for Preorder1 function

void Postorder1(node\*); //Prototype for Preorder1 function

node\* createpreorder(char\*,char\*);

node\* createpostorder(char\*,char\*);

void divide(char\*,char\*,char\*,char\*,char\*,char\*);

void divide1(char\*,char\*,char\*,char\*,char\*,char\*);

int found(char,char[]);

};

void tree::preorder() //Function Definition for preorder

{

cout<<"Enter The Preorder"<<endl;

cin>>Pre;

cout<<"The Entered Preorder is "<<Pre<<endl;

}

void tree::inorder() //Function Definition for inorder

{

cout<<"Enter The Inorder"<<endl;

cin>>In;

cout<<"The Entered Inorder is "<<In<<endl;

}

void tree::postorder() //Function Definition for postorder

{

cout<<"Enter The Postorder"<<endl;

cin>>Post;

cout<<"The Entered Postorder is "<<Post<<endl;

}

void tree ::Preorder() //Function Definition for Preorder

{

Preorder1(root); //Calling of Preorder1 function

}

node\* tree::createpreorder(char\* In,char\* Pre)

{

char In1[20];

char In2[20];

char Pre1[20];

char Pre2[20];

node \*p;

if(strlen(Pre)==0)

{

return NULL;

}

p=new node ;

p->data=Pre[0];

divide(In,Pre,In1,Pre1,In2,Pre2); //Calling of divide function

p->Left=createpreorder(In1,Pre1); //Calling of createpreorder function

p->Right=createpreorder(In2,Pre2); //Calling of createpreorder function

return p;

}

void tree::Preorder1(node\* p)

{

if(p!=NULL)

{

cout<<p->data;

Preorder1(p->Left); //Calling of Preorder1 function

Preorder1(p->Right); //Calling of Preorder1 function

}

}

void tree::Create()

{

root=createpreorder(In,Pre); //Calling of createpreorder function

}

void tree ::divide(char \*In,char \*Pre,char \*In1,char \*Pre1,char \*In2,char \*Pre2)

{

int i=0;

int j=0;

int l=0;

int m=0;

int k=1;

for(i=0;In[i]!=Pre[0];i++)

{

In1[i]=In[i];

}

In1[i]='\0';

i++;

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

{

In2[j]=In[i];

i++;

}

In2[j]='\0';

for(k=1;Pre[k]!='\0';k++)

{

if(found(Pre[k],In1))

{

Pre1[l++]=Pre[k];

}

else

{

Pre2[m++]=Pre[k];

}

}

Pre1[l]='\0';

Pre2[m]='\0';

}

int tree::found(char ch, char a[20])

{

int i;

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

{

if(ch==a[i])

{

return 1;

}

}

return 0;

}

void tree::Inorder()

{

Inorder1(root);

}

void tree::Inorder1(node\* p)

{

if(p!=NULL)

{

Inorder1(p->Left); //Calling of Inrder function

cout<<p->data;

Inorder1(p->Right); //Calling of Inorder function

}

}

void tree ::Postorder()

{

Postorder1(root); //Calling of Postorder function

}

void tree::Postorder1(node\* p)

{

if(p!=NULL)

{

Postorder1(p->Left); //Calling of Postorder function

Postorder1(p->Right); //Calling of Postorder function

cout<<p->data;

}

}

void tree::Create1()

{

root=createpostorder(In,Post); //Calling of createpostorder function

}

void tree ::divide1(char \*In,char \*Post,char \*In1,char \*Post1,char \*In2,char \*Post2)

{

int i=0;

int j=0;

int l=0;

int m=0;

int k=0;

int r;

r=strlen(Post);

for(i=0;In[i]!=Post[r-1];i++)

{

In1[i]=In[i];

}

In1[i]='\0';

i++;

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

{

In2[j]=In[i];

j++;

}

In2[j]='\0';

//for(k=0;Post[k]!=Post[r-1];k++)

for(k=0;k<(r-1);k++)

{

if(found(Post[k],In1)) //Calling of found function

{

Post1[l++]=Post[k];

}

else

{

Post2[m++]=Post[k];

}

}

Post1[l]='\0';

Post2[m]='\0';

}

node\* tree::createpostorder(char\* In,char\* Post)

{

char In1[20];

char In2[20];

char Post1[20];

char Post2[20];

node \*p;

int l;

l=strlen(Post);

if(l==0)

{

return NULL;

}

p=new node ;

p->data=Post[l-1];

divide1(In,Post,In1,Post1,In2,Post2); //Calling of divide function

p->Left=createpostorder(In1,Post1); //Calling of createpostorder function

p->Right=createpostorder(In2,Post2); //Calling of createpostorder function

return p;

}

int main()

{

tree t;

int ch;

char ch1;

do

{

cout<<"Which operation do u want to perform \n\n\t";

cout<<"\t\t1. Inorder and Preorder :\n\t";

cout<<"\t\t2. Inorder and Postorder :\n\t";

cout<<"\t\t3. Exit :\n\t";

cin>>ch;

switch(ch)

{

case 1:

t.inorder();

t.preorder();

t.Create();

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

cout<<"\nInorder is: ";

t.Inorder();

cout<<"\nPreorder is: ";

t.Preorder();

cout<<"\nPostorder is: ";

t.Postorder();

cout<<endl;

break;

case 2:

t.inorder();

t.postorder();

t.Create1();

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

cout<<"\nInorder is: ";

t.Inorder();

cout<<"\nPostorder is: ";

t.Postorder();

cout<<"\nPreorder is: ";

t.Preorder();

cout<<endl;

break;

case 3:

cout<<"------------------------THANK YOU----------------------------\n\n";

break;

default:

cout<<"\n\tEntered wrong choice\n";

}

cout<<"Do you want to continue Y/N :";

cin>>ch1;

}while(ch1=='Y'||ch1=='y');

return 0;

}

***Output :***

compeng-sl2-08@compeng-sl2-08:~/Abrar$ g++ Tree.cpp

compeng-sl2-08@compeng-sl2-08:~/Abrar$ ./a.out

Which operation do u want to perform

1. Inorder and Preorder :

2. Inorder and Postorder :

3. Exit :

1

Enter The Inorder

BCAEDGHFI

The Entered Inorder is BCAEDGHFI

Enter The Preorder

ABCDEFGHI

The Entered Preorder is ABCDEFGHI

-----OUTPUT------

Inorder is: BCAEDGHFI

Preorder is: ABCDEFGHI

Postorder is: CBEHGIFDA

Do you want to continue Y/N :Y

Which operation do u want to perform

1. Inorder and Preorder :

2. Inorder and Postorder :

3. Exit :

2

Enter The Inorder

DBFEAGCLJHK

The Entered Inorder is DBFEAGCLJHK

Enter The Postorder

DFEBGLJKHCA

The Entered Postorder is DFEBGLJKHCA

-----OUTPUT------

Inorder is: DBFEAGCLJHK

Postorder is: DFEBGLJKHCA

Preorder is: ABDEFCGHJLK

Do you want to continue Y/N y

Which operation do u want to perform

1. Inorder and Preorder :

2. Inorder and Postorder :

3. Exit :

3

------------------------THANK YOU----------------------------

Do you want to continue Y/N :N