PROJECT 10

**IMPLEMENTING AN ORDERED LIST WITH A BINARY SEARCH TREE**

Course: CSCI 301

Name: Ziqi Zhao

Section Number: 1

Project Number: 10

Due Date: Wednesday, April 24

1. **Design Document**

**Introduction:**

This program is using a binary search tree to implement an ordered list. A binary search tree (BST) is a binary tree in which the key value at each node is greater than all keys in its left subtree and less than all keys in its right subtree. In this project, the class can implement operations like insertion, removal, present, write out and so on. I create a class named BST, it’s a pointer-based binary search tree.

**Data Structure:**

This program used a class named BST to implement a pointer-based binary search tree into an ordered list. Struct the Node in the project.

**Function:**

BST( ); Constructor function, initialized empty.

~BST( ); Destructor, remove node.

bool empty( ); Boolean function, return true if the tree is empty.

void insert(int) function can insert elements and order the list in ascending.It calls the private function void r\_insert(Node \*& p, int val).

void remove(int) function is used to remove a value from a list. It calls the private function

bool present(int ); Boolean function, use the id as key value to judge whether the node existed.

int length( ) can return the length of a list. It calls the private function.

void menu( ); Menu function, show the explanation and command in the terminal.

**The main program:** The main function use “do”, “while”, “case” and “switch” to read command which the user can type in the prompt. I create a BST tree, character ‘command’, as parameter to call the functions.

1. **Code**

//================================================================

// Name : BST.cpp

// Author : Ziqi Zhao

// Course : CSCI 301

// Section : 1

// Project : 10

// Due Date : Wednesday, April 24

// Description : This is a program used a pointer-based binary search

// tree (BST) to implement an ordered list. The user

// enter command character and do operations with these items-insert

// new items, remove items and print the items.

//================================================================

# include <iostream>

# include <cstdlib>

using namespace std;

/\*

\* Node Declaration

\*/

struct node

{

int data;

node \*left;

node \*right;

};

/\*

\* Class Declaration

\*/

class BST

{

public:

// pointer to store the root of the tree.

node\* root;

bool find(node\*, int);

node\* insert(node\*, int);

node\* remove(node\*, int);

int length(node\*);

void present(node\*);

BST()

{

root = NULL;

}

~BST()

{

root = NULL;

}

};

\* Main Contains Menu

int main()

{

char choice = 'h';

int num;

BST bst;

node \*temp;

bool quit = false;

while (1)

{

switch(choice)

{

case 'e':

{

// calling the destructor and then constructor again

bst.~BST();

bst = BST();

break;

}

case 'i':

{

cin>>num;

// taking input of the number and calling the insert function

bst.root = bst.insert(bst.root, num);

break;

}

case 'r':

{

cin>>num;

// calling the remove funciton to removeete after taking the num in

bst.root = bst.remove(bst.root, num);

break;

}

case 'l':

{

int len = bst.length(bst.root);

// finding the length and then print the found value

cout<<"The list contains "<<len<< " elements"<<endl;;

break;

}

case 'p':

{

cin>>num;

// calling the find method to check if the value is present or not

bool isPresent = bst.find(bst.root, num);

if(isPresent){

cout<<"The value "<<num<<" is present in the list"<<endl;

}else{

cout<<"The value "<<num<<" is not present in the list"<<endl;

}

break;

}

case 'w':

{

cout<<"The list: ";

// printing the list sequentially using the present method of the BST

bst.present(bst.root);

cout<<endl;

break;

}

case 'h':

{

cout<<"e -- Re-initialize the list to be empty."<<endl;

cout<<"i v -- Insert the value v into the list."<<endl;

cout<<"r v -- Remove the value v from the list."<<endl;

cout<<"l -- Report the length of the list."<<endl;

cout<<"p v -- Is the value v present in the list?"<<endl;

cout<<"w -- Write out the list."<<endl;

cout<<"h -- See this menu."<<endl;

cout<<"q -- Quit."<<endl;

break;

}

case 'q':

// quiting the service.

quit = true;

break;

default:

cout<<"Wrong choice"<<endl;

break;

}

if(quit){

break;

}

cin>>choice;

}

}

/\*

\* Find Element in the Tree

\*/

bool BST::find(node\* r, int item)

{

if(r == NULL){

// we are at leaf node

return false;

}else if(r->data == item){

// if we found the item

return true;

}else{

// otherwise check in left and right part of the tree.

return this->find(r->left, item) || this->find(r->right, item);

}

}

\* Inserting Element into the Tree

node\* BST::insert(node \*r, int item)

{

// if we are at root node we need to insert our node there.

if(r == NULL){

node\* temp = new node();

temp->data = item;

temp->left = NULL;

temp->right = NULL;

return temp;

}

// if data item is less than root then we need to go to insert in the left

else if(r->data > item){

r->left = this->insert(r->left, item);

return r;

}

// if new data item is greater than we need to insert it in right of root

else if(r->data < item){

r->right = this->insert(r->right, item);

return r;

}

// if new data item is equal to the current root node then just return

// do not insert.

else{

return r;

}

}

node\* BST::remove(node\* root, int data)

{

// base case

if (root == NULL) return root;

// data to be removeeted is smaller than current root node.

// then we will find our value in the left

if (data < root->data)

root->left = remove(root->left, data);

// data to be removed is greater than current root node.

// then we will find our value in the right

else if (data > root->data)

root->right = remove(root->right, data);

else

{

// node with only one child or no child

//case 1

if (root->left == NULL)

{

struct node \*temp = root->right;

free(root);

return temp;

}

//case 2

else if (root->right == NULL)

{

struct node \*temp = root->left;

free(root);

return temp;

}

// Copy the present successor's content to this node

root->data = temp->data;

// Remove the present successor

root->right = remove(root->right, temp->data);

}

return root;

}

// length of the BST

int BST:: length(node\* root){

if(root==NULL){

return 0;

}else{

return 1 + this->length(root->left) + this->length(root->right);

}

}

// present traversal of the tree.

void BST::present(node\* root){

if(root==NULL) return;

this->present(root->left);

cout<<root->data<<" ";

this->present(root->right);

}

1. **User Document**

This program named **BST.cpp** is use binary search tree to implement ordered list The List is initially empty, the user can insert some values, remove values if it has, judge whether it empty and a specific value in this list, etc. The menu looks like this:

The program’s name is BST.cpp

To compile the program, simply enter:

g++ BST.cpp

To run it, enter:

a.out

Please enter your input after the program prompts the command line:

This program responds to commands the user enters to

manipulate an ordered list of integers, which is

initially empty. In the following commands, v is any

integer.

e -- Re-initialize the list to be empty.

i v -- Insert the value v into the list.

r v -- Remove the value v from the list.

l -- Report the length of the list.

p v -- Is the value v present in the list?

w -- Write out the list.

h -- See this menu.

q -- Quit.

1. **Tests**

csci2>a.out

This program responds to commands the user enters to

manipulate an ordered list of integers, which is

initially empty. In the following commands, v is any

integer.

e -- Re-initialize the list to be empty.

i v -- Insert the value v into the list.

r v -- Remove the value v from the list.

l -- Report the length of the list.

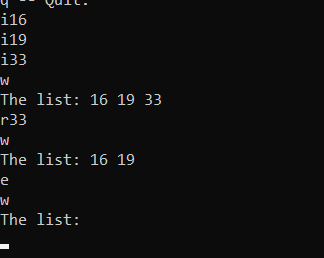
p v -- Is the value v present in the list?

w -- Write out the list.

h -- See this menu.

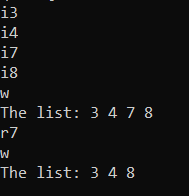
q -- Quit.

1) // This test shows Re-initialize the list to be empty. If not, write out the list.

csci2>a.out

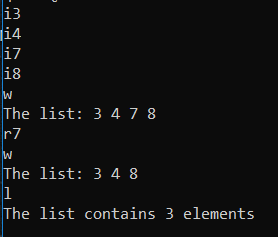
2) // This test shows Insert and remove value from the list.

csci2>a.out



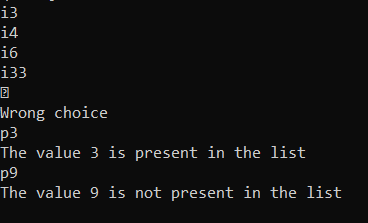
3) // This test shows report the length of the linked list,

csci2>a.out



4) // This test shows judge present value v whether in the list.

csci2>a.out



1. **Summary**

From this program, I learn how to use a binary search tree (BST) implement an ordered list. A BST is a binary tree in which the value at each node is greater than all the values in its left subtree and less than all the values in its right subtree. It’s a special ADT, the member functions like insert, present have the bog O time-O(logn). Sometimes this solution will reduce the time complexity. Another feature is that in this kind ADT, the functions are almost recursive. Recursive functions operate on pointer-based class objects must have two functions: a public one and a recursive private one that the public one calls. Usually, the base case is that tree is empty and smaller problem is the left and right subtrees.