CS 474 Homework 4 Documentation

Author:

William Montgomery

University of Illinois at Chicago

CS 474 - Object Oriented Languages and Environments

Spring 2014

This was originally generated by Doxygen as a RTF document. A HTML version of the documentation is included in the docs/html directory.

Source Code

The source code for the project is included in the "hw4" subdirectory.

Running the Project

A Makefile is provided to automate the building of the project. It can be found in the "build" directory. Running make in the "build" directory will build an executable named "hw4".

Running the Project

Once the project is built it can be run from the command line with "./hw4". It prints a menu of options to standard error and then reads in commands from standard input. The menu is printed to standard error so that the program can be run in batch mode, i.e. ./hw4 2> /dev/null.

Added Functionality

This implementation implements all of the required functionality, plus the additional functionality of removing a value from Set 1. This can be accomplished with the following command:

r [string] - Removes a string from Set 1.

Project Description

You are required to implement the all-too-familiar Set Calculator one more time for this project. As with the previous projects, you must implement sets as binary search trees (BSTs). However, this time you are required to use the language C++98 or C++03. Also the sets will contain C-style strings (implemented using the char* type). Another difference is that this set calculator will use a command line interface, rather than a graphical user interface. You will maintain two sets at any point in time, S1 and S2. The calculator supports the usual operations by responding to commands below.

Recall that BSTs are binary trees subject to the following properties. First, a string value is associated with each node. Second, each node can have at most two children, a left child and a right child. Third, given a node x, the values of all nodes in the left subtree of x are less than the value of x, and the values of all nodes in the right subtree of x are greater than the value of x. Use lexicographical ordering to compare strings. (You are encouraged to use the ASCII code of each string character to determine the value of that character.) No duplicate values will be allowed in your trees.

Use a command line interface for entering the commands below. Your command line interface will prompt the user for a command, and then execute the command. Here is a list of commands. Make sure not to cause any memory leaks or dangling pointers in the implementation of these commands.

e - Clear set. This command allows interactive users to delete the current S1 set. The previous value stored in S1 is lost.
s - Switch sets. The sets associated with S1 and S2 are swapped, meaning that S1 will receive the previousS2 set and vice versa.
c - Copy set. Set S1 is deep copied into S2. The previous content of S2 is lost. The content of S1 is not affected. The two sets must not share any data structures, that is, they can be modiï¬□ed independently of each other.
l - List set contents. The string values stored in the two sets are displayed on the standard output stream. The two sets are not modiï¬□ed.
a [string] - Add element. This function allows a user to add a new string to S1. The value is obtained through an appropriate prompt with an interactive user. No action is taken if the string in question is already in the set. The insertion should preserve the **BST** properties of S1.
u - Union. This element takes the set union of S1 and stores the resulting value in S1. The previous content of S1 is lost. S2 is not modiï¬□ed by this operation.
i - Intersection. This command takes the set intersection of S1 and stores the resulting value in S1. The previous content of S1 is lost. S2 is not modiï¬□ed by this operation.
q - Quits the set manager.

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Class Documentation

BST Class Reference

The **BST** class Encapsulates a Binary Search Tree.

`#include <bst.h>`

Public Member Functions

BST ()

__BST::BST__ Default constructor.

~**BST** ()

__BST::~BST__ Destructor.

BST * clear ()

__BST::clear__ Clears the set.

BST * switchSets (**BST** *)

__BST::switchSets__ Switches sets with another __BST__.

BST * copy (**BST** *)

__BST::copy__ Copies another __BST__.

BST * insert (const string)

__BST::insert__ Inserts a new value into the __BST__.

BST * remove (const string)

__BST::remove__ Removes a value from the __BST__.

BST * unionWith (**BST** *)

__BST::unionWith__ Unions with another __BST__.

BST * intersectionWith (**BST** *)

__BST::intersectionWith__.

Protected Attributes

BSTNode * root

Friends

ostream & **operator<<** (ostream &, const **BST** &)

operator << Puts a string representation on the output stream

---

Detailed Description

The **BST** class Encapsulates a Binary Search Tree.

Definition at line 20 of file bst.h.

Constructor & Destructor Documentation

BST::BST ()

BST::BST Default constructor.

Definition at line 12 of file bst.cpp.

BST::~BST ()

BST::~BST Destructor.

Definition at line 20 of file bst.cpp.

Member Function Documentation

BST * BST::clear ()

BST::clear Clears the set.

Returns:

this

Definition at line 32 of file bst.cpp.

BST * BST::copy (BST * *anotherBST*)

BST::copy Copies another **BST**.

Parameters:

anotherBST	The other **BST** to copy from

Returns:

this

Remove the old root

Copy the other root if it exists

Definition at line 61 of file bst.cpp.

BST * BST::insert (const string *value*)

BST::insert Inserts a new value into the **BST**.

Parameters:

value	The value

Returns:

this

Definition at line 89 of file bst.cpp.

BST * BST::intersectionWith (BST * *anotherBST*)

BST::intersectionWith.

Parameters:

anotherBST	

Returns:

If root is null or the other **BST** is null

Otherwise

Definition at line 149 of file bst.cpp.

BST * BST::remove (const string *value*)

BST::remove Removes a value from the **BST**.

Parameters:

value	The value

Returns:

this

Definition at line 108 of file bst.cpp.

BST * BST::switchSets (BST * *anotherBST*)

BST::switchSets Switches sets with another **BST**.

Parameters:

anotherBST	The other **BST**

Returns:

this

Definition at line 48 of file bst.cpp.

BST * BST::unionWith (BST * *anotherBST*)

BST::unionWith Unions with another **BST**.

Parameters:

anotherBST	The other **BST**

Returns:

this

If this root is null

If this root is not null

Definition at line 123 of file bst.cpp.

Friends And Related Function Documentation

ostream& operator<< (ostream & *os*, const BST & *bst*)`[friend]`

operator << Puts a string representation on the output stream

Parameters:

os	The output stream
bst	The **BST**

Returns:

The ostream

Definition at line 170 of file bst.cpp.

Member Data Documentation

BSTNode* BST::root`[protected]`

The root node

Definition at line 23 of file bst.h.

The documentation for this class was generated from the following files:

hw4/**bst.h**
hw4/**bst.cpp**

BSTNode Class Reference

The **BSTNode** class Encapsulates a **BST** Node.
`#include <bstnode.h>`

Public Member Functions

BSTNode (const string)
__BSTNode::BSTNode__ Constructor from a string.
BSTNode (const **BSTNode** &)
__BSTNode::BSTNode__ Copy constructor.
~BSTNode ()
__BSTNode::~BSTNode__ Destructor.
const string **min** ()
__BSTNode::min__ Finds the minimum value.
const string **max** ()
__BSTNode::max__ Finds the maximum value.
BSTNode * **insert** (const string)
__BSTNode::insert__ Inserts a value into the subtree.
BSTNode * **remove** (const string)
__BSTNode::remove__ Remove a node from the subtree.
BSTNode * **unionWith** (**BSTNode** *)
__BSTNode::unionWith__ Union with another subtree.
BSTNode * **intersectWith** (**BSTNode** *)
__BSTNode::intersectWith__ Intersects this subtree with another subtree.

Friends

ostream & **operator<<** (ostream &, const **BSTNode** &)
operator << Overloading the << operator to handle __BSTNode__

Detailed Description

The **BSTNode** class Encapsulates a **BST** Node.
Definition at line 18 of file bstnode.h.

Constructor & Destructor Documentation

BSTNode::BSTNode (const string *value*)

BSTNode::BSTNode Constructor from a string.

Parameters:

value	The value to be set

Definition at line 13 of file bstnode.cpp.

BSTNode::BSTNode (const BSTNode & *anotherNode*)

BSTNode::BSTNode Copy constructor.

Parameters:

anotherNode	The node to create from

Initialize the left node

Initialize the right node

Definition at line 24 of file bstnode.cpp.

BSTNode::~BSTNode ()

BSTNode::~BSTNode Destructor.

Definition at line 52 of file bstnode.cpp.

Member Function Documentation

BSTNode * BSTNode::insert (const string *value*)

BSTNode::insert Inserts a value into the subtree.

Parameters:

value	The value to insert

Returns:

this

If the value is less than this value

If the value is greater than this value

Otherwise it is equal and we do not insert

Definition at line 85 of file bstnode.cpp.

BSTNode * BSTNode::intersectWith (BSTNode * *anotherNode*)

BSTNode::intersectWith Intersects this subtree with another subtree.

Parameters:

anotherNode	The root of the other subtree

Returns:

The new root of this subtree

Definition at line 231 of file bstnode.cpp.

const string BSTNode::max ()

BSTNode::max Finds the maximum value.

Returns:

The maximum value

Definition at line 75 of file bstnode.cpp.

const string BSTNode::min ()

BSTNode::min Finds the minimum value.

Returns:

The minimum value

Definition at line 66 of file bstnode.cpp.

BSTNode * BSTNode::remove (const string *value*)

BSTNode::remove Remove a node from the subtree.

Parameters:

value	The value to remove

Returns:

this

The value is less than this value and there is a left node

The value is a greater than this value and there is a right node

This is the value

Leaf node

Only has right child

Only has left child

Has two children Set this to the max of the left and remove that value from the left

Definition at line 122 of file bstnode.cpp.

BSTNode * BSTNode::unionWith (BSTNode * *anotherNode*)

BSTNode::unionWith Union with another subtree.

Parameters:

anotherNode	The root of the other subtree

Returns:

this

Definition at line 184 of file bstnode.cpp.

Friends And Related Function Documentation

ostream& operator<< (ostream & *os*, const BSTNode & *node*)`[friend]`

operator << Overloading the << operator to handle **BSTNode**

Parameters:

os	The output stream
node	The node

Returns:

The output stream

Definition at line 253 of file bstnode.cpp.

The documentation for this class was generated from the following files:

hw4/**bstnode.h**
hw4/**bstnode.cpp**

File Documentation

hw4/bst.cpp File Reference

`#include "bst.h"`

Functions

ostream & **operator<<** (ostream &os, const **BST** &bst)

operator << Puts a string representation on the output stream

---

Detailed Description

The implementation file for **BST**

Author:

William Montgomery

Date:

April 2014

Definition in file **bst.cpp**.

Function Documentation

ostream& operator<< (ostream & *os*, const BST & *bst*)

operator << Puts a string representation on the output stream

Parameters:

os	The output stream
bst	The **BST**

Returns:

The ostream

Definition at line 170 of file bst.cpp.

hw4/bst.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include <string>
#include "bstnode.h"
```

Classes

class **BST**

*The **BST** class Encapsulates a Binary Search Tree.*

Detailed Description

The header file for **BST**

Author:

William Montgomery

Date:

April 2014

Definition in file **bst.h**.

hw4/bstnode.cpp File Reference

`#include "bstnode.h"`

Functions

ostream & **operator<<** (ostream &os, const **BSTNode** &node)

*operator << Overloading the << operator to handle **BSTNode***

Detailed Description

The implementation file for **BSTNode**

Author:

William Montgomery

Date:

April 2014

Definition in file **bstnode.cpp**.

Function Documentation

ostream& operator<< (ostream & *os*, const BSTNode & *node*)

operator << Overloading the << operator to handle **BSTNode**

Parameters:

os	The output stream
node	The node

Returns:

The output stream

Definition at line 253 of file bstnode.cpp.

hw4/bstnode.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include <string>
```

Classes

class **BSTNode**

*The **BSTNode** class Encapsulates a **BST** Node.*

Detailed Description

The header file for **BSTNode**

Author:

William Montgomery

Date:

April 2014

Definition in file **bstnode.h**.

hw4/main.cpp File Reference

```
#include <iostream>
#include "bst.h"
```

Functions

void **printMenu** ()

printMenu Prints out a menu of options

bool **performAction** (**BST** &set1, **BST** &set2)

performAction Reads in user input and then performs the action

int **main** (void)

main The main function

Detailed Description

The driver for the program

Author:

William Montgomery

Date:

April 2014

Definition in file **main.cpp**.

Function Documentation

int main (void)

main The main function

Returns:

0, unless there is an error

Definition at line 89 of file main.cpp.

bool performAction (BST & *set1*, BST & *set2*)

performAction Reads in user input and then performs the action

Parameters:

set1	The first set
set2	The second set

Returns:

True if there are more actions to process

Definition at line 38 of file main.cpp.

void printMenu ()

printMenu Prints out a menu of options

Definition at line 16 of file main.cpp.

hw4/mainpage.dox File Reference

Index
INDEX