HW_012

0.3.0

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

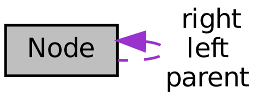
Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Node Struct Reference

Collaboration diagram for Node:



Public Attributes

- int data
- Node ∗ parent
- Node ∗ left
- Node ∗ right
- int color

3.1.1 Detailed Description

Definition at line 12 of file main.cpp.

3.1.2 Member Data Documentation

3.1.2.1 color

```
int Node::color
```

Definition at line 17 of file main.cpp.

3.1.2.2 data

```
int Node::data
```

Definition at line 13 of file main.cpp.

3.1.2.3 left

```
Node* Node::left
```

Definition at line 15 of file main.cpp.

3.1.2.4 parent

```
Node* Node::parent
```

Definition at line 14 of file main.cpp.

3.1.2.5 right

```
Node* Node::right
```

Definition at line 16 of file main.cpp.

The documentation for this struct was generated from the following file:

- /home/scott/CPTR227/HW_012-Red-Black-Tree/src/main.cpp

3.2 RBTree Class Reference

Public Member Functions

- RBTree ()
- void preorder ()
- void inorder ()
- void postorder ()
- NodePtr searchTree (int k)
- NodePtr minimum (NodePtr node)
- NodePtr maximum (NodePtr node)
- NodePtr successor (NodePtr x)
- NodePtr predecessor (NodePtr x)
- void leftRotate (NodePtr x)
- void rightRotate (NodePtr x)
- void insert (int key)
- NodePtr getRoot ()
- void deleteNode (int data)
- void prettyPrint ()

3.2.1 Detailed Description

Definition at line 23 of file main.cpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 RBTree()

```
RBTree::RBTree ( )  [inline]
```

Definition at line 278 of file main.cpp.

```
                {
        TNULL = new Node;
        TNULL->color = 0;
        TNULL->left = nullptr;
        TNULL->right = nullptr;
        root = TNULL;
    }
```

3.2.3 Member Function Documentation

3.2.3.1 deleteNode()

```
void RBTree::deleteNode (
            int data )  [inline]
```

Definition at line 454 of file main.cpp.

```
                                  {
        deleteNodeHelper(this->root, data);
    }
```

3.2.3.2 getRoot()

```
NodePtr RBTree::getRoot ( )  [inline]
```

Definition at line 449 of file main.cpp.

```
                         {
        return this->root;
    }
```

3.2.3.3 inorder()

```
void RBTree::inorder ( )  [inline]
```

Definition at line 294 of file main.cpp.

```
                       {
        inOrderHelper(this->root);
    }
```

3.2.3.4 insert()

```
void RBTree::insert (
            int key )  [inline]
```

Definition at line 403 of file main.cpp.

```
                             {
        // Ordinary Binary Search Insertion
        NodePtr node = new Node;
        node->parent = nullptr;
        node->data = key;
        node->left = TNULL;
        node->right = TNULL;
        node->color = 1; // new node must be red

        NodePtr y = nullptr;
        NodePtr x = this->root;

        while (x != TNULL) {
            y = x;
            if (node->data < x->data) {
                x = x->left;
            } else {
                x = x->right;
            }
        }

        // y is parent of x
        node->parent = y;
        if (y == nullptr) {
            root = node;
        } else if (node->data < y->data) {
            y->left = node;
        } else {
            y->right = node;
        }

        // if new node is a root node, simply return
        if (node->parent == nullptr){
            node->color = 0;
            return;
        }

        // if the grandparent is null, simply return
        if (node->parent->parent == nullptr) {
            return;
        }

        // Fix the tree
        fixInsert(node);
    }
```

3.2.3.5 leftRotate()

```
void RBTree::leftRotate (
            NodePtr x )  [inline]
```

Definition at line 364 of file main.cpp.

```
                                {
        NodePtr y = x->right;
        x->right = y->left;
        if (y->left != TNULL) {
            y->left->parent = x;
        }
        y->parent = x->parent;
        if (x->parent == nullptr) {
            this->root = y;
        } else if (x == x->parent->left) {
            x->parent->left = y;
        } else {
            x->parent->right = y;
        }
        y->left = x;
        x->parent = y;
    }
```

3.2.3.6 maximum()

```
NodePtr RBTree::maximum (
            NodePtr node )  [inline]
```

Definition at line 319 of file main.cpp.

```
                                    {
        while (node->right != TNULL) {
            node = node->right;
        }
        return node;
    }
```

3.2.3.7 minimum()

```
NodePtr RBTree::minimum (
            NodePtr node ) [inline]
```

Definition at line 311 of file main.cpp.

```
                                    {
        while (node->left != TNULL) {
            node = node->left;
        }
        return node;
    }
```

3.2.3.8 postorder()

```
void RBTree::postorder ( ) [inline]
```

Definition at line 300 of file main.cpp.

```
                      {
        postOrderHelper(this->root);
    }
```

3.2.3.9 predecessor()

```
NodePtr RBTree::predecessor (
            NodePtr x ) [inline]
```

Definition at line 346 of file main.cpp.

```
                                    {
        // if the left subtree is not null,
        // the predecessor is the rightmost node in the
        // left subtree
        if (x->left != TNULL) {
            return maximum(x->left);
        }

        NodePtr y = x->parent;
        while (y != TNULL && x == y->left) {
            x = y;
            y = y->parent;
        }

        return y;
    }
```

3.2.3.10 preorder()

```
void RBTree::preorder ( ) [inline]
```

Definition at line 288 of file main.cpp.

```
                     {
        preOrderHelper(this->root);
    }
```

3.2.3.11 prettyPrint()

```
void RBTree::prettyPrint ( )  [inline]
```

Definition at line 459 of file main.cpp.

```
                        {
        if (root) {
            printHelper(this->root, "", true);
        }
    }
```

3.2.3.12 rightRotate()

```
void RBTree::rightRotate (
             NodePtr x )  [inline]
```

Definition at line 383 of file main.cpp.

```
                                {
        NodePtr y = x->left;
        x->left = y->right;
        if (y->right != TNULL) {
            y->right->parent = x;
        }
        y->parent = x->parent;
        if (x->parent == nullptr) {
            this->root = y;
        } else if (x == x->parent->right) {
            x->parent->right = y;
        } else {
            x->parent->left = y;
        }
        y->right = x;
        x->parent = y;
    }
```

3.2.3.13 searchTree()

```
NodePtr RBTree::searchTree (
             int k )  [inline]
```

Definition at line 306 of file main.cpp.

```
                              {
        return searchTreeHelper(this->root, k);
    }
```

3.2.3.14 successor()

```
NodePtr RBTree::successor (
            NodePtr x )  [inline]
```

Definition at line 327 of file main.cpp.

```
                                  {
          // if the right subtree is not null,
          // the successor is the leftmost node in the
          // right subtree
          if (x->right != TNULL) {
              return minimum(x->right);
          }

          // else it is the lowest ancestor of x whose
          // left child is also an ancestor of x.
          NodePtr y = x->parent;
          while (y != TNULL && x == y->right) {
              x = y;
              y = y->parent;
          }
          return y;
      }
```

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

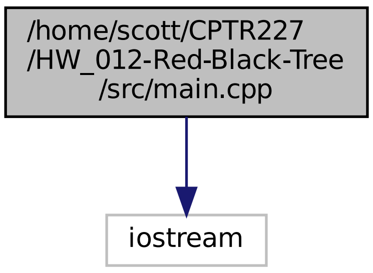
- /home/scott/CPTR227/HW_012-Red-Black-Tree/src/main.cpp

Chapter 4

File Documentation

4.1 /home/scott/CPTR227/HW_012-Red-Black-Tree/src/main.cpp File Reference

```
#include <iostream>
```
Include dependency graph for main.cpp:



Classes

- struct Node
- class RBTree

Typedefs

- typedef Node ∗ NodePtr

Functions

- int main ()

4.1.1 Typedef Documentation

4.1.1.1 NodePtr

```
typedef Node* NodePtr
```

Definition at line 20 of file main.cpp.

4.1.2 Function Documentation

4.1.2.1 main()

```
int main ( )
```

Definition at line 467 of file main.cpp.

```
          {
    RBTree bst;
    bst.insert(1);
    bst.insert(11);
    bst.insert(521312);
    bst.insert(2);
    bst.insert(1302134);
    bst.insert(3);
    bst.insert(402141);
    bst.insert(81240);
    bst.deleteNode(2);
    bst.prettyPrint();
    return 0;
}
```

Index