aardvark

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

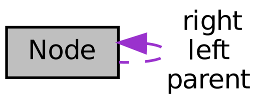
Here is a list of all files with brief descriptions:

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 11 of file main.cpp.

3.1.2 Member Data Documentation

3.1.2.1 color

```
int Node::color
```

Definition at line 16 of file main.cpp.

3.1.2.2 data

```
int Node::data
```

Definition at line 12 of file main.cpp.

3.1.2.3 left

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

Definition at line 14 of file main.cpp.

3.1.2.4 parent

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

Definition at line 13 of file main.cpp.

3.1.2.5 right

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

Definition at line 15 of file main.cpp.

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

- /home/addis/RED-BLACK-TREE-S/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 22 of file main.cpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 RBTree()

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

Definition at line 277 of file main.cpp.

```
277                 {
278         TNULL = new Node;
279         TNULL->color = 0;
280         TNULL->left = nullptr;
281         TNULL->right = nullptr;
282         root = TNULL;
283     }
```

3.2.3 Member Function Documentation

3.2.3.1 deleteNode()

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

Definition at line 453 of file main.cpp.

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

3.2.3.2 getRoot()

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

Definition at line 448 of file main.cpp.

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

3.2.3.3 inorder()

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

Definition at line 293 of file main.cpp.

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

3.2.3.4 insert()

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

Definition at line 402 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 363 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 318 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 310 of file main.cpp.

```
310                                  {
311         while (node->left != TNULL) {
312             node = node->left;
313         }
314         return node;
315     }
```

3.2.3.8 postorder()

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

Definition at line 299 of file main.cpp.

```
299                       {
300         postOrderHelper(this->root);
301     }
```

3.2.3.9 predecessor()

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

Definition at line 345 of file main.cpp.

```
345                                   {
346         // if the left subtree is not null,
347         // the predecessor is the rightmost node in the
348         // left subtree
349         if (x->left != TNULL) {
350             return maximum(x->left);
351         }
352
353         NodePtr y = x->parent;
354         while (y != TNULL && x == y->left) {
355             x = y;
356             y = y->parent;
357         }
358
359         return y;
360     }
```

3.2.3.10 preorder()

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

Definition at line 287 of file main.cpp.

```
287                      {
288         preOrderHelper(this->root);
289     }
```

3.2.3.11 prettyPrint()

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

Definition at line 458 of file main.cpp.

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

3.2.3.12 rightRotate()

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

Definition at line 382 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 305 of file main.cpp.

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

3.2.3.14 successor()

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

Definition at line 326 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:

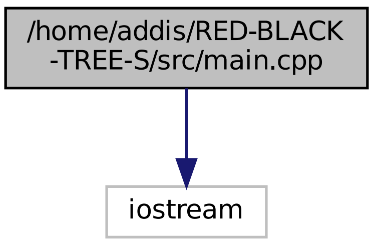
- /home/addis/RED-BLACK-TREE-S/src/main.cpp

Chapter 4

File Documentation

4.1 /home/addis/RED-BLACK-TREE-S/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 19 of file main.cpp.

4.1.2 Function Documentation

4.1.2.1 main()

```
int main ( )
```

Definition at line 466 of file main.cpp.

```
{
    RBTree bst;
    bst.insert(8);
    bst.insert(18);
    bst.insert(5);
    bst.insert(15);
    bst.insert(17);
    bst.insert(25);
    bst.insert(40);
    bst.insert(80);
    bst.deleteNode(25);
    bst.prettyPrint();

    bst.insert(9);
    bst.insert(19);
    bst.insert(6);
    bst.insert(16);
    bst.insert(18);
    bst.insert(26);
    bst.insert(41);
    bst.insert(81);
    bst.deleteNode(26);
    bst.prettyPrint();

    bst.insert(7);
    bst.insert(17);
    bst.insert(4);
    bst.insert(14);
    bst.insert(16);
    bst.insert(24);
    bst.insert(39);
    bst.insert(79);
    bst.deleteNode(24);
    bst.prettyPrint();

    bst.insert(10);
    bst.insert(20);
    bst.insert(7);
    bst.insert(17);
    bst.insert(19);
    bst.insert(27);
    bst.insert(42);
    bst.insert(82);
    bst.deleteNode(27);
    bst.prettyPrint();

    bst.insert(6);
    bst.insert(16);
    bst.insert(3);
    bst.insert(13);
    bst.insert(15);
    bst.insert(23);
    bst.insert(38);
    bst.insert(79);
    bst.deleteNode(23);
    bst.prettyPrint();
    return 0;
}
```

Index