

Week 3 Assignment 1

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 tree Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 data	5
3.1.2.2 left	5
3.1.2.3 right	5
4 File Documentation	7
4.1 inc/tree.h File Reference	7
4.1.1 Typedef Documentation	8
4.1.1.1 tree_t	8
4.1.2 Function Documentation	8
4.1.2.1 createTreeNode()	8
4.1.2.2 delete()	8
4.1.2.3 findLeftmostNode()	8
4.1.2.4 inorder_traverse()	9
4.1.2.5 insert()	9
4.1.2.6 search()	9
4.2 tree.h	10
4.3 src/createTreeNode.c File Reference	10
4.3.1 Function Documentation	10
4.3.1.1 createTreeNode()	10
4.4 src/deleteNode.c File Reference	11
4.4.1 Function Documentation	11
4.4.1.1 delete()	11
4.5 src/findLeftMode.c File Reference	11
4.5.1 Function Documentation	11
4.5.1.1 findLeftmostNode()	11
4.6 src/inorderTraversal.c File Reference	12
4.6.1 Function Documentation	12
4.6.1.1 inorder_traverse()	12
4.7 src/insertNode.c File Reference	12
4.7.1 Function Documentation	13
4.7.1.1 insert()	13
4.8 src/main.c File Reference	13
4.9 src/searchNode.c File Reference	13

4.9.1 Function Documentation	13
4.9.1.1 search()	13
Index	15

Chapter 1

Class Index

1.1 Class List

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

tree	Structure representing a binary tree node	5
----------------------	---	-------------------

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

inc/ tree.h	7
src/ createTreeNode.c	10
src/ deleteNode.c	11
src/ findLeftMode.c	11
src/ inorderTraversal.c	12
src/ insertNode.c	12
src/ main.c	13
src/ searchNode.c	13

Chapter 3

Class Documentation

3.1 tree Struct Reference

Structure representing a binary tree node.

```
#include <tree.h>
```

Collaboration diagram for tree:

Public Attributes

- int [data](#)
Data stored in the node.
- struct [tree](#) * [left](#)
- struct [tree](#) * [right](#)
Pointers to the left and right child nodes.

3.1.1 Detailed Description

Structure representing a binary tree node.

3.1.2 Member Data Documentation

3.1.2.1 data

```
int tree::data
```

Data stored in the node.

3.1.2.2 left

```
struct tree* tree::left
```

3.1.2.3 right

```
struct tree * tree::right
```

Pointers to the left and right child nodes.

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

- inc/[tree.h](#)

Chapter 4

File Documentation

4.1 inc/tree.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdint.h>
```

Include dependency graph for tree.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [tree](#)
Structure representing a binary tree node.

Typedefs

- typedef struct [tree](#) [tree_t](#)
Structure representing a binary tree node.

Functions

- [tree_t](#) * [createTreeNode](#) (int value)
Creates a new tree node with the given value.
- void [insert](#) ([tree_t](#) **[tree](#), int value)
Inserts a new node with the given value into the binary tree.
- void [inorder_traverse](#) ([tree_t](#) *[tree](#))
Performs an in-order traversal of the tree and prints the node values.
- [tree_t](#) * [findLeftmostNode](#) ([tree_t](#) *[tree](#))
Finds and returns the leftmost node (minimum value) in the tree.
- void [delete](#) ([tree_t](#) **[tree](#), int value)
Deletes a node with the specified value from the tree.
- [tree_t](#) * [search](#) ([tree_t](#) *[tree](#), int value)
Searches for a node with the specified value in the tree.

4.1.1 Typedef Documentation

4.1.1.1 tree_t

```
typedef struct tree tree_t
```

Structure representing a binary tree node.

4.1.2 Function Documentation

4.1.2.1 createTreeNode()

```
tree_t * createTreeNode (  
    int value ) [extern]
```

Creates a new tree node with the given value.

Parameters

<i>value</i>	The value to be stored in the new node.
--------------	---

Returns

tree_t* Pointer to the newly created node.

4.1.2.2 delete()

```
void delete (  
    tree_t ** tree,  
    int value ) [extern]
```

Deletes a node with the specified value from the tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
<i>value</i>	The value of the node to be deleted.

4.1.2.3 findLeftmostNode()

```
tree_t * findLeftmostNode (  
    tree_t * tree ) [extern]
```

Finds and returns the leftmost node (minimum value) in the tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
-------------	----------------------------------

Returns

*tree_t** Pointer to the leftmost node in the tree.

4.1.2.4 inorder_traverse()

```
void inorder_traverse (  
    tree_t * tree )    [extern]
```

Performs an in-order traversal of the tree and prints the node values.

Parameters

<i>tree</i>	Pointer to the root of the tree.
-------------	----------------------------------

4.1.2.5 insert()

```
void insert (  
    tree_t ** tree,  
    int value )    [extern]
```

Inserts a new node with the given value into the binary tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
<i>value</i>	The value to be inserted into the tree.

4.1.2.6 search()

```
tree_t * search (  
    tree_t * tree,  
    int value )    [extern]
```

Searches for a node with the specified value in the tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
<i>value</i>	The value to search for.

Returns

`tree_t*` Pointer to the node if found, NULL if not found.

4.2 tree.h

[Go to the documentation of this file.](#)

```

00001 #ifndef TREE_H
00002 #define TREE_H
00003
00004 #include<stdio.h>
00005 #include<stdlib.h>
00006 #include<stdbool.h>
00007 #include<stdint.h>
00008
00012 typedef struct tree
00013 {
00014     int data;
00015     struct tree *left, *right;
00016 } tree_t;
00017
00018 extern tree_t * createTreeNode(int value);
00019 extern void insert(tree_t **tree, int value);
00020 extern void inorder_traverse(tree_t *tree);
00021 extern tree_t* findLeftmostNode(tree_t *tree);
00022 extern void delete(tree_t **tree, int value);
00023 extern tree_t* search(tree_t *tree, int value);
00024
00025 #endif

```

4.3 src/createTreeNode.c File Reference

```
#include "tree.h"
```

Include dependency graph for createTreeNode.c:

Functions

- [tree_t* createTreeNode](#) (int value)
Creates a new tree node with the given value.

4.3.1 Function Documentation

4.3.1.1 createTreeNode()

```
tree_t * createTreeNode (
    int value )
```

Creates a new tree node with the given value.

Parameters

<i>value</i>	The value to be stored in the new node.
--------------	---

Returns

`tree_t*` Pointer to the newly created node.

4.4 src/deleteNode.c File Reference

```
#include "tree.h"
```

Include dependency graph for deleteNode.c:

Functions

- void `delete` (`tree_t **tree`, int value)
Deletes a node with the specified value from the tree.

4.4.1 Function Documentation

4.4.1.1 delete()

```
void delete (  
    tree_t ** tree,  
    int value )
```

Deletes a node with the specified value from the tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
<i>value</i>	The value of the node to be deleted.

4.5 src/findLeftMode.c File Reference

```
#include "tree.h"
```

Include dependency graph for findLeftMode.c:

Functions

- `tree_t *` `findLeftmostNode` (`tree_t *tree`)
Finds and returns the leftmost node (minimum value) in the tree.

4.5.1 Function Documentation

4.5.1.1 findLeftmostNode()

```
tree_t * findLeftmostNode (  
    tree_t * tree )
```

Finds and returns the leftmost node (minimum value) in the tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
-------------	----------------------------------

Returns

*tree_t** Pointer to the leftmost node in the tree.

4.6 src/inorderTraversal.c File Reference

```
#include "tree.h"
```

Include dependency graph for inorderTraversal.c:

Functions

- void [inorder_traverse](#) ([tree_t](#) **tree*)
Performs an in-order traversal of the tree and prints the node values.

4.6.1 Function Documentation

4.6.1.1 [inorder_traverse\(\)](#)

```
void inorder_traverse (  
    tree\_t * tree )
```

Performs an in-order traversal of the tree and prints the node values.

Parameters

<i>tree</i>	Pointer to the root of the tree.
-------------	----------------------------------

4.7 src/insertNode.c File Reference

```
#include "tree.h"
```

Include dependency graph for insertNode.c:

Functions

- void [insert](#) ([tree_t](#) ***tree*, int value)
Inserts a new node with the given value into the binary tree.

4.7.1 Function Documentation

4.7.1.1 insert()

```
void insert (
    tree_t ** tree,
    int value )
```

Inserts a new node with the given value into the binary tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
<i>value</i>	The value to be inserted into the tree.

4.8 src/main.c File Reference

```
#include "tree.h"
Include dependency graph for main.c:
```

4.9 src/searchNode.c File Reference

```
#include "tree.h"
Include dependency graph for searchNode.c:
```

Functions

- `tree_t * search (tree_t *tree, int value)`
Searches for a node with the specified value in the tree.

4.9.1 Function Documentation

4.9.1.1 search()

```
tree_t * search (
    tree_t * tree,
    int value )
```

Searches for a node with the specified value in the tree.

Parameters

<i>tree</i>	Pointer to the root of the tree.
<i>value</i>	The value to search for.

Returns

tree_t* Pointer to the node if found, NULL if not found.

Index

- createTreeNode
 - createTreeNode.c, [10](#)
 - tree.h, [8](#)
- createTreeNode.c
 - createTreeNode, [10](#)
- data
 - tree, [5](#)
- delete
 - deleteNode.c, [11](#)
 - tree.h, [8](#)
- deleteNode.c
 - delete, [11](#)
- findLeftMode.c
 - findLeftmostNode, [11](#)
- findLeftmostNode
 - findLeftMode.c, [11](#)
 - tree.h, [8](#)
- inc/tree.h, [7](#), [10](#)
- inorder_traverse
 - inorderTraversal.c, [12](#)
 - tree.h, [9](#)
- inorderTraversal.c
 - inorder_traverse, [12](#)
- insert
 - insertNode.c, [13](#)
 - tree.h, [9](#)
- insertNode.c
 - insert, [13](#)
- left
 - tree, [5](#)
- right
 - tree, [5](#)
- search
 - searchNode.c, [13](#)
 - tree.h, [9](#)
- searchNode.c
 - search, [13](#)
- src/createTreeNode.c, [10](#)
- src/deleteNode.c, [11](#)
- src/findLeftMode.c, [11](#)
- src/inorderTraversal.c, [12](#)
- src/insertNode.c, [12](#)
- src/main.c, [13](#)
- src/searchNode.c, [13](#)
- tree, [5](#)
 - data, [5](#)
 - left, [5](#)
 - right, [5](#)
- tree.h
 - createTreeNode, [8](#)
 - delete, [8](#)
 - findLeftmostNode, [8](#)
 - inorder_traverse, [9](#)
 - insert, [9](#)
 - search, [9](#)
 - tree_t, [8](#)
- tree_t
 - tree.h, [8](#)