

## BinarySearchTree

1

Generated by Doxygen 1.8.18

<b>1 Class Index</b>	<b>1</b>
<b>1 Class Index</b>	<b>1</b>
1.1 Class List . . . . .	1
<b>2 File Index</b>	<b>1</b>
2.1 File List . . . . .	1
<b>3 Class Documentation</b>	<b>2</b>
3.1 BST Class Reference . . . . .	2
3.1.1 Detailed Description . . . . .	3
3.1.2 Constructor & Destructor Documentation . . . . .	3
3.1.3 Member Function Documentation . . . . .	3
3.1.4 Member Data Documentation . . . . .	7
3.2 BST::node Struct Reference . . . . .	7
3.2.1 Member Data Documentation . . . . .	7
<b>4 File Documentation</b>	<b>8</b>
4.1 BST.cpp File Reference . . . . .	8
4.2 BST.h File Reference . . . . .	8
4.3 driver.cpp File Reference . . . . .	8
4.3.1 Function Documentation . . . . .	8
<b>Index</b>	<b>11</b>

## 1 Class Index

### 1.1 Class List

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

<b>BST</b>	
<b>BST (p. 2) class to traverse nodes and delete or re-organize Class that builds and navigates a binary search tree</b>	<b>2</b>
<b>BST::node</b>	<b>7</b>

## 2 File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<b>BST.cpp</b>	<b>8</b>
<b>BST.h</b>	<b>8</b>
<b>driver.cpp</b>	<b>8</b>

## 3 Class Documentation

### 3.1 BST Class Reference

**BST** (p. 2) class to traverse nodes and delete or re-organize Class that builds and navigates a binary search tree.

```
#include <BST.h>
```

#### Classes

- struct **node**

#### Public Member Functions

- **BST** ()
- void **addLeaf** (char key)  
*Adds a leaf.*
- void **printPreOrder** ()  
*Prints the **BST** (p. 2) in Pre-Order.*
- void **searchKey** (int key)  
*Searches tree for a given value.*
- void **removeNode** (int key)  
*Removes a node with a provided key This is really a helper function for removeNodePrivate.*
- int **findsmallest** ()  
*Helper function for findsmallestPrivate.*
- int **count** ()  
*Counts how many leafs are in tree helper function.*
- void **find** (char keyToFind)  
*Searches for the first match.*

#### Private Member Functions

- void **addLeafPrivate** (int key, **node** \*Ptr)  
*Adds a leaf private.*
- void **printPreOrderPrivate** ( **node** \*Ptr)  
*Prints the values of the **BST** (p. 2)'s private members.*
- void **searchKeyPrivate** (char key, **node** \*Ptr)
- void **removeNodePrivate** (int key, **node** \*parent)  
*Removes a node's private key value.*
- void **removeRootMatch** ()  
*Removes a root match.*
- int **findsmallestPrivate** ( **node** \*Ptr)  
*Finds the smallest node.*
- void **removeMatch** ( **node** \*parent, **node** \*match, bool left)  
*Removes a match.*
- int **countPrivate** ( **node** \*Ptr, int & count)  
*Counts the number of all leafs.*
- **node** \* **createLeaf** (char key)  
*Adds a leaf to the **BST** (p. 2).*
- void **findPrivate** ( **node** \*Ptr, char key)  
*Finds a private.*

## Private Attributes

- `node * root`

### 3.1.1 Detailed Description

**BST** (p. 2) class to traverse nodes and delete or re-organize Class that builds and navigates a binary search tree.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 **BST()** `BST::BST ( )`

### 3.1.3 Member Function Documentation

#### 3.1.3.1 **addLeaf()** `void BST::addLeaf ( char key )`

Adds a leaf.

##### Parameters

in	<i>key</i>	The key/value to add
----	------------	----------------------

#### 3.1.3.2 **addLeafPrivate()** `void BST::addLeafPrivate ( int key, node * Ptr ) [private]`

Adds a leaf private.

##### Parameters

in	<i>key</i>	The key to be added
	<i>Ptr</i>	The node pointer

#### 3.1.3.3 **count()** `int BST::count ( )`

Counts how many leafs are in tree helper function.

**Returns**

returns to helper function to traverse other side of tree

```
3.1.3.4 countPrivate()  int BST::countPrivate (
                        node * Ptr,
                        int & count ) [private]
```

Counts the number of all leafs.

**Parameters**

<i>Ptr</i>	The pointer needed to traverse the tree
<i>count</i>	The amount of trees

**Returns**

Number of leafs.

```
3.1.3.5 createLeaf()  BST::node * BST::createLeaf (
                        char key ) [private]
```

Adds a leaf to the **BST** (p.2).

**Parameters**

in	<i>key</i>	The value being added
----	------------	-----------------------

**Returns**

returns node pointer to last value entered

```
3.1.3.6 find()  void BST::find (
                char key )
```

Searches for the first match.

**Parameters**

in	<i>key</i>	The key to search for
----	------------	-----------------------

**3.1.3.7 findPrivate()** `void BST::findPrivate (`  
    `node * Ptr,`  
    `char key ) [private]`

Finds a private.

#### Parameters

	<i>Ptr</i>	The pointer to traverse the tree
in	<i>key</i>	The key to be found

**3.1.3.8 findsmallest()** `int BST::findsmallest ( )`

Helper function for findsmallestPrivate.

#### Returns

returns to associated function to recurse tree again

**3.1.3.9 findsmallestPrivate()** `int BST::findsmallestPrivate (`  
    `node * Ptr ) [private]`

Finds the smallest node.

#### Parameters

<i>Ptr</i>	The pointer used to traverse the tree
------------	---------------------------------------

#### Returns

return's node with smallest key

**3.1.3.10 printPreOrder()** `void BST::printPreOrder ( )`

Prints the **BST** (p. 2) in Pre-Order.

**3.1.3.11 printPreOrderPrivate()** `void BST::printPreOrderPrivate (`  
    `node * Ptr ) [private]`

Prints the values of the **BST** (p. 2)'s private members.

**Parameters**

<i>Ptr</i>	The pointer needed to recurse through tree
------------	--

**3.1.3.12 removeMatch()** void BST::removeMatch (   
    **node** \* *parent*,  
    **node** \* *match*,  
    bool *left* ) [private]

Removes a match.

**Parameters**

	<i>parent</i>	The parent
	<i>match</i>	The match
in	<i>left</i>	The left

**3.1.3.13 removeNode()** void BST::removeNode (   
    int *key* )

Removes a node with a provided key This is really a helper function for removeNodePrivate.

**Parameters**

in	<i>key</i>	The key to be removed.
----	------------	------------------------

**3.1.3.14 removeNodePrivate()** void BST::removeNodePrivate (   
    int *key*,  
    **node** \* *parent* ) [private]

Removes a node's private key value.

**Parameters**

in	<i>key</i>	The key to be removed
	<i>parent</i>	The parent above the removed key for reassigning child node

**3.1.3.15 removeRootMatch()** void BST::removeRootMatch ( ) [private]

Removes a root match.

**3.1.3.16 searchKey()** `void BST::searchKey (`  
`int key )`

Searches tree for a given value.

Parameters

in	key	The key to find
----	-----	-----------------

**3.1.3.17 searchKeyPrivate()** `void BST::searchKeyPrivate (`  
`char key,`  
`node * Ptr ) [private]`

### 3.1.4 Member Data Documentation

**3.1.4.1 root** `node* BST::root [private]`

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

- **BST.h**
- **BST.cpp**

## 3.2 BST::node Struct Reference

### Public Attributes

- `char key`
- `node * left = NULL`
- `node * right = NULL`

### 3.2.1 Member Data Documentation

**3.2.1.1 key** `char BST::node::key`

**3.2.1.2 left** `node* BST::node::left = NULL`



**3.2.1.3 right** `node* BST::node::right = NULL`

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

- **BST.h**

## 4 File Documentation

### 4.1 BST.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include "BST.h"
```

### 4.2 BST.h File Reference

#### Classes

- class **BST**  
*BST (p. 2) class to traverse nodes and delete or re-organize Class that builds and navigates a binary search tree.*
- struct **BST::node**

### 4.3 driver.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include "BST.h"
```

#### Functions

- int **menu** ()  
*Menu for user to interact.*
- void **clearScreen** ()  
*clears screen for next interface to appear*
- void **pauseScreen** ()  
*Keeps info on screen long enough for user to read.*
- int **main** ()  
*Driver main function Creates a UI for the user to interact with.*

#### 4.3.1 Function Documentation

**4.3.1.1 clearScreen()** `void clearScreen ( )`

clears screen for next interface to appear

**4.3.1.2 main()** `int main ( )`

Driver main function Creates a UI for the user to interact with.

**4.3.1.3 menu()** `int menu ( )`

Menu for user to interact.

**Returns**

returns a value to switch case to operate off of

**4.3.1.4 pauseScreen()** `void pauseScreen ( )`

Keeps info on screen long enough for user to read.



## Index

- addLeaf
  - BST, 3
- addLeafPrivate
  - BST, 3
- BST, 2
  - addLeaf, 3
  - addLeafPrivate, 3
  - BST, 3
  - count, 3
  - countPrivate, 4
  - createLeaf, 4
  - find, 4
  - findPrivate, 4
  - findsmallest, 5
  - findsmallestPrivate, 5
  - printPreOrder, 5
  - printPreOrderPrivate, 5
  - removeMatch, 6
  - removeNode, 6
  - removeNodePrivate, 6
  - removeRootMatch, 6
  - root, 7
  - searchKey, 6
  - searchKeyPrivate, 7
- BST.cpp, 8
- BST.h, 8
- BST::node, 7
  - key, 7
  - left, 7
  - right, 7
- clearScreen
  - driver.cpp, 8
- count
  - BST, 3
- countPrivate
  - BST, 4
- createLeaf
  - BST, 4
- driver.cpp, 8
  - clearScreen, 8
  - main, 9
  - menu, 9
  - pauseScreen, 9
- find
  - BST, 4
- findPrivate
  - BST, 4
- findsmallest
  - BST, 5
- findsmallestPrivate
  - BST, 5
- key
  - BST::node, 7
- left
  - BST::node, 7
- main
  - driver.cpp, 9
- menu
  - driver.cpp, 9
- pauseScreen
  - driver.cpp, 9
- printPreOrder
  - BST, 5
- printPreOrderPrivate
  - BST, 5
- removeMatch
  - BST, 6
- removeNode
  - BST, 6
- removeNodePrivate
  - BST, 6
- removeRootMatch
  - BST, 6
- right
  - BST::node, 7
- root
  - BST, 7
- searchKey
  - BST, 6
- searchKeyPrivate
  - BST, 7