My Project

Generated by Doxygen 1.7.6.1

Thu Dec 5 2013 12:31:11

Contents

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

Declaring a simple struct for the node. Size indicates how many items the node has, depth indicates the vertical position of the node within the tree, key is the array that will contain the inserted values. Finally, parent and child are pointers to those respective nodes.

`#include <node.h>`

Public Attributes

- int size
- int depth
- int key
- node * parent
- node * child [4]

3.1.1 Detailed Description

Declaring a simple struct for the node. Size indicates how many items the node has, depth indicates the vertical position of the node within the tree, key is the array that will contain the inserted values. Finally, parent and child are pointers to those respective nodes.

3.1.2 Member Data Documentation

3.1.2.1 node* node::child[4]

3.1.2.2 int node::depth

3.1.2.3 int node::key

3.1.2.4 node* node::parent

3.1.2.5 int node::size

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

- node.h
- tree.h

3.2 tree Class Reference

Class tree has only one root node initially.

`#include <tree.h>`

Public Member Functions

- tree ()

  The class creator initializes variables size and depth to zero, since nodes are empty.
- ~tree ()

  Standard destructor.
- void insert (int key)

  Insert function: Compares which "slots" are empty in each node. Also compares values to insert in the right position.
- void search (int key)

Public Attributes

- node root

3.2.1 Detailed Description

Class tree has only one root node initially.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 tree::tree ( )

The class creator initializes variables size and depth to zero, since nodes are empty.

3.2.2.2 tree::~tree ( )

Standard destructor.

3.2.3 Member Function Documentation

3.2.3.1 void tree::insert (int *key*)

Insert function: Compares which "slots" are empty in each node. Also compares values to insert in the right position.

If the node is full, we create a new node to insert more data

3.2.3.2 void tree::search (int *key*)

3.2.4 Member Data Documentation

3.2.4.1 node tree::root

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

- tree.h
- tree.cpp

Chapter 4

File Documentation

4.1 main.cpp File Reference

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

Functions

- int main ()

4.1.1 Function Documentation

4.1.1.1 int main ( )

4.2 node.h File Reference

```
#include <iostream> #include <vector>
```

Classes

- struct node

  Declaring a simple struct for the node. Size indicates how many items the node has, depth indicates the vertical position of the node within the tree, key is the array that will contain the inserted values. Finally, parent and child are pointers to those respective nodes.

4.3 tree.cpp File Reference

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

4.4 tree.h File Reference

```
#include <iostream> #include <vector> #include <new>
```

Classes

- struct node

  Declaring a simple struct for the node. Size indicates how many items the node has, depth indicates the vertical position of the node within the tree, key is the array that will contain the inserted values. Finally, parent and child are pointers to those respective nodes.

- class tree

  Class tree has only one root node initially.