Drzewo Binarne

Generated by Doxygen 1.12.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 BST Class Reference

Implements a Binary Search Tree.

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

Data Structures

- struct Node

  Represents a single node in the BST.

Public Member Functions

- BST ()

  Constructs an empty BST.
- ~BST ()

  Destructor to free all nodes in the tree.
- void insert (int value)

  Inserts a value into the BST.
- bool remove (int value)
- void clear ()
- bool findPath (int value, std::vector< int > &path)
- void printTree (int order)
- void saveToFile (const std::string &filename) const
- void loadFromFile (const std::string &filename)

Private Member Functions

- void insert (Node ∗&node, int value)

  Recursive helper for inserting a value.
- bool remove (Node ∗&node, int value)
- void clear (Node ∗node)
- bool findPath (Node ∗node, int value, std::vector< int > &path)
- void printPreOrder (Node ∗node)
- void printInOrder (Node ∗node)
- void printPostOrder (Node ∗node)
- void saveToFile (Node ∗node, std::ofstream &outFile) const
- void loadFromFile (Node ∗&node, std::ifstream &inFile)

Private Attributes

- Node ∗ root
  Root node of the BST.

3.1.1 Detailed Description

Implements a Binary Search Tree.

This class provides methods for inserting and removing elements, printing the tree in different traversal orders, and saving/loading the tree to/from a binary file.

Definition at line 23 of file BST.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BST()

```
BST::BST ()
```

Constructs an empty BST.

3.1.2.2 ∼BST()

```
BST::∼BST ()
```

Destructor to free all nodes in the tree.

3.1.3 Member Function Documentation

3.1.3.1 clear() [1/2]

```
void BST::clear ()
```

3.1.3.2 clear() [2/2]

```
void BST::clear (
             Node ∗ node)  [private]
```

3.1.3.3 findPath() [1/2]

```
bool BST::findPath (
             int value,
             std::vector< int > & path)
```

3.1.3.4 findPath() [2/2]

```
bool BST::findPath (
             Node * node,
             int value,
             std::vector< int > & path)  [private]
```

3.1.3.5 insert() [1/2]

```
void BST::insert (
             int value)
```

Inserts a value into the BST.

Parameters

value	Value to insert.

Definition at line 12 of file BST.cpp.

3.1.3.6 insert() [2/2]

```
void BST::insert (
             Node *& node,
             int value)  [private]
```

Recursive helper for inserting a value.

Parameters

node	Pointer to the current node.
value	Value to insert.

Definition at line 21 of file BST.cpp.

3.1.3.7 loadFromFile() [1/2]

```
void BST::loadFromFile (
             const std::string & filename)
```

3.1.3.8 loadFromFile() [2/2]

```
void BST::loadFromFile (
             Node *& node,
             std::ifstream & inFile)  [private]
```

3.1.3.9 printInOrder()

```
void BST::printInOrder (
             Node * node)  [private]
```

3.1.3.10 printPostOrder()

```
void BST::printPostOrder (
             Node * node)  [private]
```

3.1.3.11 printPreOrder()

```
void BST::printPreOrder (
             Node * node)  [private]
```

3.1.3.12 printTree()

```
void BST::printTree (
             int order)
```

3.1.3.13 remove() [1/2]

```
bool BST::remove (
             int value)
```

3.1.3.14 remove() [2/2]

```
bool BST::remove (
             Node *& node,
             int value)  [private]
```

3.1.3.15 saveToFile() [1/2]

```
void BST::saveToFile (
             const std::string & filename) const
```

3.1.3.16 saveToFile() [2/2]

```
void BST::saveToFile (
             Node * node,
             std::ofstream & outFile) const  [private]
```

3.1.4 Field Documentation

3.1.4.1 root

```
Node* BST::root  [private]
```

Root node of the BST.

Definition at line 41 of file BST.h.

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

- BST.h
- BST.cpp

3.2 FileManager Class Reference

Handles file operations for the BST.

```
#include <FileManager.h>
```

Public Member Functions

- void saveTreeToBinaryFile (const BST &tree, const std::string &filename)
  Saves the BST to a binary file.
- void loadTreeFromBinaryFile (BST &tree, const std::string &filename)
  Loads the BST from a binary file.

3.2.1 Detailed Description

Handles file operations for the BST.

This class provides methods to save a binary search tree to a binary file and load a binary search tree from a binary file.

Definition at line 18 of file FileManager.h.

3.2.2 Member Function Documentation

3.2.2.1 loadTreeFromBinaryFile()

```
void FileManager::loadTreeFromBinaryFile (
             BST & tree,
             const std::string & filename)
```

Loads the BST from a binary file.

Loads the BST from a binary file.

Parameters

tree	Reference to the BST object to populate.
filename	Name of the file to load the tree from.

Parameters

tree	Reference to the BST object to populate.
filename	Name of the file to load the tree from.

Definition at line 19 of file FileManager.cpp.

3.2.2.2 saveTreeToBinaryFile()

```
void FileManager::saveTreeToBinaryFile (
             const BST & tree,
             const std::string & filename)
```

Saves the BST to a binary file.

Saves the BST to a binary file.

Parameters

tree	Reference to the BST object.
filename	Name of the file to save the tree.

Parameters

tree	Reference to the BST object.
filename	Name of the file to save the tree.

Definition at line 12 of file FileManager.cpp.

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

- FileManager.h
- FileManager.cpp

3.3 BST::Node Struct Reference

Represents a single node in the BST.

Public Member Functions

- Node (int value)

  Constructs a node with the given value.

Data Fields

- int data

  Value stored in the node.
- Node ∗ left

  Pointer to the left child.
- Node ∗ right

  Pointer to the right child.

3.3.1 Detailed Description

Represents a single node in the BST.

Definition at line 29 of file BST.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Node()

```
BST::Node::Node (
                int value)  [inline]
```

Constructs a node with the given value.

Parameters

value	Value to store in the node.

Definition at line 38 of file BST.h.

3.3.3 Field Documentation

3.3.3.1 data

```
int BST::Node::data
```

Value stored in the node.

Definition at line 30 of file BST.h.

3.3.3.2 left

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

Pointer to the left child.

Definition at line 31 of file BST.h.

3.3.3.3 right

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

Pointer to the right child.

Definition at line 32 of file BST.h.

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

- BST.h

Chapter 4

File Documentation

4.1 BST.cpp File Reference

Implementation of the BST class.

```
#include "BST.h"
```

4.1.1 Detailed Description

Implementation of the BST class.

Definition in file BST.cpp.

4.2 BST.cpp

Go to the documentation of this file.

```

#include "BST.h"

void BST::insert(int value) {
    insert(root, value);
}

void BST::insert(Node*& node, int value) {
    if (node == nullptr) {
        node = new Node(value);
    }
    else if (value < node->data) {
        insert(node->left, value);
    }
    else {
        insert(node->right, value);
    }
}
```

4.3 BST.h File Reference

Defines the Binary Search Tree (BST) class.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
```

Data Structures

- class BST

  Implements a Binary Search Tree.
- struct BST::Node

  Represents a single node in the BST.

Macros

- #define BST_H

4.3.1 Detailed Description

Defines the Binary Search Tree (BST) class.

Definition in file BST.h.

4.3.2 Macro Definition Documentation

4.3.2.1 BST_H

```
#define BST_H
```

Definition at line 8 of file BST.h.

4.4 BST.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007 #ifndef BST_H
00008 #define BST_H
00009
00010 #include <iostream>
00011 #include <fstream>
00012 #include <vector>
00013 #include <string>
00014
00023 class BST {
00024 private:
00029     struct Node {
00030         int data;
00031         Node* left;
00032         Node* right;
00033
00038         Node(int value) : data(value), left(nullptr), right(nullptr) {}
00039     };
00040
00041     Node* root;
00042
00043     void insert(Node*& node, int value);
00044     bool remove(Node*& node, int value);
00045     void clear(Node* node);
00046     bool findPath(Node* node, int value, std::vector<int>& path);
00047     void printPreOrder(Node* node);
00048     void printInOrder(Node* node);
00049     void printPostOrder(Node* node);
00050     void saveToFile(Node* node, std::ofstream& outFile) const;
00051     void loadFromFile(Node*& node, std::ifstream& inFile);
00052
00053 public:
    BST();

    ~BST();

    void insert(int value);
    bool remove(int value);
    void clear();
    bool findPath(int value, std::vector<int>& path);
    void printTree(int order);

    void saveToFile(const std::string& filename) const;
    void loadFromFile(const std::string& filename);
};

#endif
```

4.5 FileManager.cpp File Reference

Implementation of file management functions for the BST.

```
#include "FileManager.h"
#include <fstream>
```

4.5.1 Detailed Description

Implementation of file management functions for the BST.

Definition in file FileManager.cpp.

4.6 FileManager.cpp

Go to the documentation of this file.

```

#include "FileManager.h"
#include <fstream>

void FileManager::saveTreeToBinaryFile(const BST& tree, const std::string& filename) {
    tree.saveToFile(filename);
}

void FileManager::loadTreeFromBinaryFile(BST& tree, const std::string& filename) {
    tree.loadFromFile(filename);
}
```

4.7 FileManager.h File Reference

Provides file management operations for saving/loading BST.

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

Data Structures

- class FileManager

  Handles file operations for the BST.

4.7.1 Detailed Description

Provides file management operations for saving/loading BST.

Definition in file FileManager.h.

4.8 FileManager.h

Go to the documentation of this file.

```
#pragma once

#include "BST.h"
#include <string>

class FileManager {
public:
    void saveTreeToBinaryFile(const BST& tree, const std::string& filename);

    void loadTreeFromBinaryFile(BST& tree, const std::string& filename);
};
```

4.9 main.cpp File Reference

Entry point for the binary search tree (BST) program.

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

Functions

- void menu ()
  Displays the menu options to the user.
- int main ()
  Main function to interact with the user and manage the BST.

4.9.1 Detailed Description

Entry point for the binary search tree (BST) program.

This file contains the main logic for interacting with the binary search tree through a console-based menu. It uses the `BST` class for tree operations and `FileManager` class for file handling.

Definition in file main.cpp.

4.9.2 Function Documentation

4.9.2.1 main()

```
int main ()
```

Main function to interact with the user and manage the BST.

The function presents a menu-driven interface to perform operations like inserting and removing elements, printing the tree, and saving/loading the tree to/from a binary file.

Returns

int Exit status.

Definition at line 35 of file main.cpp.

4.9.2.2 menu()

```
void menu ()
```

Displays the menu options to the user.

Definition at line 17 of file main.cpp.

4.10 main.cpp

Go to the documentation of this file.

```
00001
00010 #include <iostream>
00011 #include "BST.h"
00012 #include "FileManager.h"
00013
00017 void menu() {
00018     std::cout « "1. Insert Element\n";
00019     std::cout « "2. Remove Element\n";
00020     std::cout « "3. Print Tree\n";
00021     std::cout « "4. Save Tree to File\n";
00022     std::cout « "5. Load Tree from File\n";
00023     std::cout « "6. Exit\n";
00024 }
00025
00035 int main() {
00036     BST tree;
00037     int choice;
00038
00039     while (true) {
00040         menu();
00041         std::cin » choice;
00042
00043         if (choice == 1) {
00044             int value;
00045             std::cout « "Enter value to insert: ";
00046             std::cin » value;
00047             tree.insert(value);
00048         }
00049         else if (choice == 2) {
00050             int value;
00051             std::cout « "Enter value to remove: ";
00052             std::cin » value;
00053             tree.remove(value);
00054         }
00055         else if (choice == 3) {
00056             std::cout « "Choose traversal order (1- Preorder, 2- Inorder, 3- Postorder): ";
00057             int order;
```

```
            std::cin » order;
            tree.printTree(order);
        }
        else if (choice == 4) {
            std::string filename;
            std::cout « "Enter filename to save: ";
            std::cin » filename;
            tree.saveToFile(filename);
        }
        else if (choice == 5) {
            std::string filename;
            std::cout « "Enter filename to load: ";
            std::cin » filename;
            tree.loadFromFile(filename);
        }
        else if (choice == 6) {
            break;
        }
    }

    return 0;
}
```

Index