DrzewoBST

Wygenerowano za pomocą Doxygen 1.12.0

1 Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

BST
Klasa reprezentująca drzewo BST (Binary Search Tree) 1

2 Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików wraz z ich krótkimi opisami:

ConsoleApplication1.cpp 7

3 Dokumentacja klas

3.1 Dokumentacja klasy BST

Klasa reprezentująca drzewo BST (Binary Search Tree).

Metody publiczne

- BST ()

  Konstruktor inicjalizujący puste drzewo.
- Node * getRoot ()

  Zwraca wskaźnik do korzenia drzewa.
- void setRoot (Node *_root)

  Ustawia wskaźnik do korzenia drzewa.
- void addElement (int value)

  Dodaje element do drzewa.
- void deleteElement (int value)

  Usuwa element o zadanej wartości.
- void deleteTree ()

  Usuwa całe drzewo, usuwając wszystkie jego węzły.
- void findPath (int value)

  Znajduje i wyświetla ścieżkę do elementu o zadanej wartości.
- void displayTree (const string &order)

  Wyświetla drzewo w wybranym porządku.
- void saveToFile (const string &filename)

  Zapisuje drzewo do pliku tekstowego.

Metody prywatne

- Node * addElement (Node *node, int value)

  Prywatna metoda dodająca element do drzewa.
- Node * findMin (Node *node)

  Prywatna metoda znajdująca węzeł o najmniejszej wartości.
- Node * deleteElement (Node *node, int value)

  Prywatna metoda usuwająca element z drzewa.
- void inOrder (Node *node)

  Prywatna metoda wyświetlająca drzewo w porządku inorder.
- void preOrder (Node *node)

  Prywatna metoda wyświetlająca drzewo w porządku preorder.
- void postOrder (Node *node)

  Prywatna metoda wyświetlająca drzewo w porządku postorder.
- void saveToFile (Node *node, ofstream &file)

  Prywatna metoda zapisująca drzewo do pliku.

Atrybuty prywatne

- Node * root

3.1.1 Opis szczegółowy

Klasa reprezentująca drzewo BST (Binary Search Tree).

Definicja w linii 17 pliku ConsoleApplication1.cpp.

3.1.2 Dokumentacja konstruktora i destruktora

BST()

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

Konstruktor inicjalizujący puste drzewo.

3.1.3 Dokumentacja funkcji składowych

addElement() [1/2]

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

Dodaje element do drzewa.

Parametry

value	Wartość do dodania.

addElement() [2/2]

```
Node * BST::addElement (
            Node * node,
            int value)  [private]
```

Prywatna metoda dodająca element do drzewa.

Parametry

node	Wskaźnik do bieżącego węzła.
value	Wartość do dodania do drzewa.

Zwraca

Wskaźnik do nowego lub zmodyfikowanego węzła.

deleteElement() [1/2]

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

Usuwa element o zadanej wartości.

Parametry

value	Wartość do usunięcia.

deleteElement() [2/2]

```
Node * BST::deleteElement (
            Node * node,
            int value)  [private]
```

Prywatna metoda usuwająca element z drzewa.

Parametry

node	Wskaźnik do bieżącego węzła.
value	Wartość do usunięcia.

Zwraca

Wskaźnik do nowego lub zmodyfikowanego węzła.

deleteTree()

```
void BST::deleteTree ()
```

Usuwa całe drzewo, usuwając wszystkie jego węzły.

displayTree()

```
void BST::displayTree (
            const string & order)
```

Wyświetla drzewo w wybranym porządku.

Parametry

order	Kolejność wyświetlania ("inorder", "preorder", "postorder").

findMin()

```
Node * BST::findMin (
            Node * node)  [private]
```

Prywatna metoda znajdująca węzeł o najmniejszej wartości.

Parametry

node	Wskaźnik do bieżącego węzła.

Zwraca

Wskaźnik do węzła o najmniejszej wartości.

findPath()

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

Znajduje i wyświetla ścieżkę do elementu o zadanej wartości.

Parametry

value	Wartość do znalezienia.

getRoot()

```
Node * BST::getRoot ()
```

Zwraca wskaźnik do korzenia drzewa.

Zwraca

Wskaźnik do korzenia.

inOrder()

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

Prywatna metoda wyświetlająca drzewo w porządku inorder.

Parametry

node	Wskaźnik do bieżącego węzła.

postOrder()

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

Prywatna metoda wyświetlająca drzewo w porządku postorder.

Parametry

node	Wskaźnik do bieżącego węzła.

preOrder()

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

Prywatna metoda wyświetlająca drzewo w porządku preorder.

Parametry

node	Wskaźnik do bieżącego węzła.

saveToFile() [1/2]

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

Zapisuje drzewo do pliku tekstowego.

Parametry

filename	Nazwa pliku, do którego zostanie zapisane drzewo.

saveToFile() [2/2]

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

Prywatna metoda zapisująca drzewo do pliku.

Parametry

node	Wskaźnik do bieżącego węzła.
file	Strumień wyjściowy pliku.

setRoot()

```
void BST::setRoot (
            Node * _root)
```

Ustawia wskaźnik do korzenia drzewa.

Parametry

_root	Nowy wskaźnik do korzenia.

3.1.4 Dokumentacja atrybutów składowych

root

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

Wskaźnik do korzenia drzewa

Definicja w linii 19 pliku ConsoleApplication1.cpp.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ConsoleApplication1.cpp

4 Dokumentacja plików

4.1 Dokumentacja pliku ConsoleApplication1.cpp

```
#include <iostream>
#include <fstream>
#include "Node.hpp"
#include "BSTFileHandler.hpp"
```

Komponenty

- class BST

  Klasa reprezentująca drzewo BST (Binary Search Tree).

Funkcje

- int main ()

  Funkcja główna programu do obsługi drzewa BST.

4.1.1 Dokumentacja funkcji

main()

```
int main ()
```

Funkcja główna programu do obsługi drzewa BST.

Zwraca

0 jeśli program zakończy się poprawnie.

Definicja w linii 127 pliku ConsoleApplication1.cpp.

```
00127           {
00128     BST tree;
00129     int option, value;
00130     string order, filename;
00131 
00132     do {
00133         // Menu główne programu
00134         cout « "\nMenu:\n";
00135         cout « "1. Add element\n";
00136         cout « "2. Delete element\n";
00137         cout « "3. Delete entire tree\n";
00138         cout « "4. Find path to element\n";
00139         cout « "5. Display tree\n";
00140         cout « "6. Save tree to binary file\n";
00141         cout « "7. Load tree from binary file\n";
00142         cout « "8. Exit\n";
00143         cout « "Choose an option: ";
00144         cin » option;
00145 
00146         switch (option) {
00147         case 1:
00148             cout « "Enter value to add: ";
00149             cin » value;
00150             tree.addElement(value); // Dodawanie elementu do drzewa
00151             break;
00152         case 2:
00153             cout « "Enter value to delete: ";
00154             cin » value;
00155             tree.deleteElement(value); // Usuwanie elementu z drzewa
00156             break;
00157         case 3:
00158             tree.deleteTree(); // Usuwanie całego drzewa
00159             break;
00160         case 4:
00161             cout « "Enter value to find path: ";
00162             cin » value;
00163             tree.findPath(value); // Znajdowanie ścieżki do elementu
00164             break;
00165         case 5:
00166             cout « "Enter display order (inorder, preorder, postorder): ";
00167             cin » order;
00168             tree.displayTree(order); // Wyświetlanie drzewa w wybranym porządku
00169             break;
00170         case 6:
00171             cout « "Enter filename to save tree: ";
00172             cin » filename;
00173             BSTFileHandler::saveToBinaryFile(tree.getRoot(), filename); // Zapisywanie drzewa do pliku
      binarnego
00174             break;
00175         case 7:
00176             cout « "Enter filename to load tree: ";
00177             cin » filename;
00178             tree.setRoot(BSTFileHandler::loadFromBinaryFile(filename)); // Wczytywanie drzewa z pliku
      binarnego
00179             break;
00180         case 8:
00181             cout « "Exiting program." « endl; // Wyjście z programu
00182             break;
00183         default:
00184             cout « "Invalid option. Try again." « endl; // Obsługa niepoprawnego wyboru
00185         }
00186     } while (option != 8);
00187 
00188     return 0;
00189 }
```

4.2 ConsoleApplication1.cpp

Idź do dokumentacji tego pliku.

```
00001
00006 #include <iostream>
00007 #include <fstream>
00008 #include "Node.hpp"
00009 #include "BSTFileHandler.hpp" // Zakładamy, że ta klasa obsługuje operacje binarne
00010
00011 using namespace std;
00012
00017 class BST {
00018 private:
00019     Node* root;
00027     Node* addElement(Node* node, int value);
00028
00034     Node* findMin(Node* node);
00035
00042     Node* deleteElement(Node* node, int value);
00043
00048     void inOrder(Node* node);
00049
00054     void preOrder(Node* node);
00055
00060     void postOrder(Node* node);
00061
00067     void saveToFile(Node* node, ofstream& file);
00068
00069 public:
00073     BST();
00074
00079     Node* getRoot();
00080
00085     void setRoot(Node* _root);
00086
00091     void addElement(int value);
00092
00097     void deleteElement(int value);
00098
00102     void deleteTree();
00103
00108     void findPath(int value);
00109
00114     void displayTree(const string& order);
00115
00120     void saveToFile(const string& filename);
00121 };
00122
00127 int main() {
00128     BST tree;
00129     int option, value;
00130     string order, filename;
00131
00132     do {
00133         // Menu główne programu
00134         cout « "\nMenu:\n";
00135         cout « "1. Add element\n";
00136         cout « "2. Delete element\n";
00137         cout « "3. Delete entire tree\n";
00138         cout « "4. Find path to element\n";
00139         cout « "5. Display tree\n";
00140         cout « "6. Save tree to binary file\n";
00141         cout « "7. Load tree from binary file\n";
00142         cout « "8. Exit\n";
00143         cout « "Choose an option: ";
00144         cin » option;
00145
00146         switch (option) {
00147         case 1:
00148             cout « "Enter value to add: ";
00149             cin » value;
00150             tree.addElement(value); // Dodawanie elementu do drzewa
00151             break;
00152         case 2:
00153             cout « "Enter value to delete: ";
00154             cin » value;
00155             tree.deleteElement(value); // Usuwanie elementu z drzewa
00156             break;
00157         case 3:
00158             tree.deleteTree(); // Usuwanie całego drzewa
00159             break;
00160         case 4:
00161             cout « "Enter value to find path: ";
00162             cin » value;
00163             tree.findPath(value); // Znajdowanie ścieżki do elementu
00164             break;
```

```
00165         case 5:
00166             cout « "Enter display order (inorder, preorder, postorder): ";
00167             cin » order;
00168             tree.displayTree(order); // Wyświetlanie drzewa w wybranym porządku
00169             break;
00170         case 6:
00171             cout « "Enter filename to save tree: ";
00172             cin » filename;
00173             BSTFileHandler::saveToBinaryFile(tree.getRoot(), filename); // Zapisywanie drzewa do pliku
      binarnego
00174             break;
00175         case 7:
00176             cout « "Enter filename to load tree: ";
00177             cin » filename;
00178             tree.setRoot(BSTFileHandler::loadFromBinaryFile(filename)); // Wczytywanie drzewa z pliku
      binarnego
00179             break;
00180         case 8:
00181             cout « "Exiting program." « endl; // Wyjście z programu
00182             break;
00183         default:
00184             cout « "Invalid option. Try again." « endl; // Obsługa niepoprawnego wyboru
00185         }
00186     } while (option != 8);
00187 
00188     return 0;
00189 }
```

Skorowidz