Drzewo_BST

Wygenerowano za pomocą Doxygen 1.12.0

Rozdział 1

Indeks klas

1.1 Lista klas

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

Rozdział 2

Indeks plików

2.1 Lista plików

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

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja klasy BST

Implementacja drzewa BST (Binary Search Tree).

`#include <BST.h>`

Metody publiczne

- BST ()

  Konstruktor klasy BST.
- ~BST ()

  Destruktor klasy BST.
- void insert (int value)

  Wstawia element do drzewa.
- void remove (int value)

  Usuwa element z drzewa.
- void removeTree ()

  Usuwa ca³e drzewo.
- void showInorder () const

  Wywietla drzewo w porz¹dku inorder.
- void showPreorder () const

  Wywietla drzewo w porz¹dku preorder.
- void showPostorder () const

  Wywietla drzewo w porz¹dku postorder.
- void displayTree () const

  Wywietla drzewo w formie graficznej (strukturalnej).
- void saveToFile (const string &filename, int orderType) const

  Zapisuje drzewo do pliku w okrelonym porz¹dku.

3.1.1 Opis szczegÃ³Åowy

Implementacja drzewa BST (Binary Search Tree).

Klasa implementuj¹ca drzewo binarne wyszukiwania (Binary Search Tree).

Ta klasa implementuje drzewo binarne wyszukiwania, oferuj¹c operacje takie jak: wstawianie, usuwanie, oraz wywietlanie drzewa w ró¿nych porz¹dkach (inorder, preorder, postorder).

Klasa ta implementuje drzewo binarne wyszukiwania i oferuje metody do wstawiania, usuwania, wywietlania i zapisywania elementów drzewa w ró¿nych porz¹dkach.

3.1.2 Dokumentacja konstruktora i destruktora

3.1.2.1 BST()

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

Konstruktor klasy BST.

Inicjalizuje drzewo jako puste (root = nullptr).

3.1.2.2 ∼BST()

```
BST::∼BST ()
```

Destruktor klasy BST.

Usuwa ca³e drzewo poprzez wywo³anie metody removeTree.

Usuwa ca³e drzewo binarne poprzez wywo³anie metody removeTree.

3.1.3 Dokumentacja funkcji skÅadowych

3.1.3.1 displayTree()

```
void BST::displayTree () const
```

Wywietla drzewo w formie graficznej (strukturalnej).

Wywietla drzewo w formie graficznej.

Funkcja ta wywo³uje rekurencyjn¹ wersjê displayTree, aby wywietliæ strukturê ca³ego drzewa.

3.1.3.2 insert()

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

Wstawia element do drzewa.

Funkcja wywo³uje rekurencyjn¹ wersjê insert, przekazuj¹c root jako pocz¹tkowy wêze³.

Parametry

value	Wartoæ do wstawienia.

3.1.3.3 remove()

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

Usuwa element z drzewa.

Funkcja wywo^{3}uje rekurencyjn¹ wersjê remove, przekazuj¹c root jako pocz¹tkowy wêze^3.

Parametry

value	Wartoæ do usuniêcia.

3.1.3.4 removeTree()

```
void BST::removeTree ()
```

Usuwa ca^{3}e drzewo.

Funkcja wywo^{3}uje rekurencyjn¹ wersjê removeTree, aby usun¹æ wszystkie wêz^{3}y drzewa.

Funkcja wywo^{3}uje rekurencyjn¹ wersjê removeTree, przekazuj¹c root jako pocz¹tkowy wêze^3.

3.1.3.5 saveToFile()

```
void BST::saveToFile (
            const string & filename,
            int orderType) const
```

Zapisuje drzewo do pliku w okrelonym porz¹dku.

Funkcja zapisuje elementy drzewa do pliku w jednym z trzech porz¹dków: inorder, preorder, postorder.

Parametry

filename	Nazwa pliku, do którego zostanie zapisane drzewo.
orderType	Typ porz¹dku (1 - inorder, 2 - preorder, 3 - postorder).

3.1.3.6 showInorder()

```
void BST::showInorder () const
```

Wywietla drzewo w porz¹dku inorder.

Funkcja ta wypisuje elementy drzewa w porz¹dku inorder (lewy, korzeñ, prawy).

3.1.3.7 showPostorder()

```
void BST::showPostorder () const
```

Wywietla drzewo w porz¹dku postorder.

Funkcja ta wypisuje elementy drzewa w porz¹dku postorder (lewy, prawy, korzeñ).

3.1.3.8 showPreorder()

```
void BST::showPreorder () const
```

Wywietla drzewo w porz¹dku preorder.

Funkcja ta wypisuje elementy drzewa w porz¹dku preorder (korzeñ, lewy, prawy).

Dokumentacja dla tej klasy zostaÅa wygenerowana z plikÃ^{3}w:

- C:/gry/Drzewo__BST/Drzewo__BST/BST.h
- C:/gry/Drzewo__BST/Drzewo__BST/BST.cpp

Rozdział 4

Dokumentacja plików

4.1 BST.h

```
00001 #ifndef BST_H
00002 #define BST_H
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <fstream>
00007 #include <string>
00008
00009 using namespace std;
00010
00018 class BST {
00019 private:
00026     struct Node {
00027         int data;
00028         Node* left;
00029         Node* right;
00030
00038         Node(int value) : data(value), left(nullptr), right(nullptr) {}
00039     };
00040
00041     Node* root;
00042
00051     void insert(Node*& node, int value);
00052
00062     Node* remove(Node* node, int value);
00063
00071     void removeTree(Node*& node);
00072
00082     void inorder(Node* node, vector<int>& result) const;
00083
00093     void preorder(Node* node, vector<int>& result) const;
00094
00104     void postorder(Node* node, vector<int>& result) const;
00105
00114     void displayTree(Node* node, int indent = 0) const;
00115
00116 public:
00122     BST();
00123
00129     ~BST();
00130
00138     void insert(int value);
00139
00147     void remove(int value);
00148
00154     void removeTree();
00155
00161     void showInorder() const;
00162
00168     void showPreorder() const;
00169
00175     void showPostorder() const;
00176
00182     void displayTree() const;
00183
00193     void saveToFile(const string& filename, int orderType) const;
00194 };
00195
00196 #endif
```

Skorowidz