Drzewo BST C++

Wygenerowano za pomocą Doxygen 1.15.0

Rozdział 1

Implementacja drzewa BST w języku C++

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

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

Rozdział 4

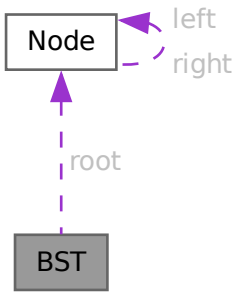
Dokumentacja klas

4.1 Dokumentacja klasy BST

Implementuje binarne drzewo poszukiwan (BST).

`#include <BST.h>`

Diagram współpracy dla BST:



Metody publiczne

- BST ()

  Konstruktor domyslny.
- ~BST ()

  Destruktor.
- void clear ()

  Usuwa wszystkie wezly z drzewa.
- bool isEmpty ()

  Sprawdza, czy drzewo jest puste.
- void add (int value)

Dodaje nowa wartosc do drzewa.

- void remove (int value)

  Usuwa wezel o podanej wartosci.
- void displayInorder ()

  Wyswietla drzewo w kolejnosci in-order (LKP).
- void displayPreorder ()

  Wyswietla drzewo w kolejnosci pre-order (KLP).
- void displayPostorder ()

  Wyswietla drzewo w kolejnosci post-order (LPK).
- void displayGraphical ()

  Wyswietla graficzna (tekstowa) reprezentacje drzewa.
- void saveToFile (const string &filename)

  Zapisuje drzewo binarnie do pliku (pre-order).
- void loadFromFile (const string &filename)

  Wczytuje drzewo z pliku binarnego.

Metody prywatne

- Node ∗ removeRecursive (Node ∗node, int value)

  Prywatny pomocnik rekursywny do usuwania.
- Node ∗ findMin (Node ∗node)

  Prywatny pomocnik do znajdowania minimum w poddrzewie.
- Node ∗ addRecursive (Node ∗node, int value)

  Prywatny pomocnik rekursywny do dodawania.
- void saveRecursive (Node ∗node, ofstream &outFile)

  Prywatny pomocnik rekursywny do zapisu do pliku (pre-order).
- void clearRecursive (Node ∗node)

  Prywatny pomocnik rekursywny do czyszczenia drzewa.
- void inorderRecursive (Node ∗node)

  Prywatny pomocnik rekursywny do wyswietlania in-order.
- void preorderRecursive (Node ∗node)

  Prywatny pomocnik rekursywny do wyswietlania pre-order.
- void postorderRecursive (Node ∗node)

  Prywatny pomocnik rekursywny do wyswietlania post-order.
- void displayGraphicalRecursive (Node ∗node, const string &prefix, bool isLeft, bool isRoot)

  Prywatny pomocnik rekursywny do wyswietlania graficznego.

Atrybuty prywatne

- Node ∗ root

  Wskaznik na korzen drzewa.

4.1.1 Opis szczegółowy

Implementuje binarne drzewo poszukiwan (BST).

- Zapewnia operacje dodawania, usuwania, wyswietlania oraz zapisu/odczytu z pliku.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 BST()

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

Konstruktor domyslny.

- Inicjalizuje puste drzewo, ustawiajac korzen (root) na nullptr.

4.1.2.2 ∼BST()

```
BST::∼BST ()
```

Destruktor.

Nota

- W biezacej implementacji jest pusty. Poprawny destruktor powinien wywolac clear() do zwolnienia pamieci. np. BST::∼BST() { clear(); }

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 add()

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

Dodaje nowa wartosc do drzewa.

Publiczna metoda dodajaca nowa wartosc do drzewa.

*

Parametry

value	Wartosc do dodania.

- Rozpoczyna rekursywne dodawanie od korzenia.
- **Parametry**

value	Wartosc do dodania.

4.1.3.2 addRecursive()

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

Prywatny pomocnik rekursywny do dodawania.

Rekursywny pomocnik do dodawania nowej wartosci.

- Znajduje odpowiednie miejsce w drzewie i wstawia nowy wezel, zachowujac wlasnosci drzewa BST.
- **Parametry**

node	Wskaznik na korzen poddrzewa, do ktorego wstawiamy.
value	Wartosc do wstawienia.

Zwraca

Wskaznik na korzen zmodyfikowanego poddrzewa.

4.1.3.3 clear()

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

Usuwa wszystkie wezly z drzewa.

- Po wykonaniu tej metody drzewo staje sie puste (root = nullptr).

4.1.3.4 clearRecursive()

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

Prywatny pomocnik rekursywny do czyszczenia drzewa.

Rekursywny pomocnik do usuwania wszystkich wezlow drzewa.

- Przechodzi przez drzewo w kolejnosci post-order i usuwa kazdy wezel.

- **Parametry**

node	Wskaznik na biezacy wezel do przetworzenia.

4.1.3.5 displayGraphical()

```
void BST::displayGraphical ()
```

Wyswietla graficzna (tekstowa) reprezentacje drzewa.

- Drzewo jest obrocone o 90 stopni w lewo (korzen po lewej, prawe dziecko u gory, lewe na dole).

4.1.3.6 displayGraphicalRecursive()

```
void BST::displayGraphicalRecursive (
            Node * node,
            const string & prefix,
            bool isLeft,
            bool isRoot)  [private]
```

Prywatny pomocnik rekursywny do wyswietlania graficznego.

Rekursywny pomocnik do wyswietlania graficznej reprezentacji.

- Uzywa "odwroconego" przejscia in-order (PKL), aby wydrukowac drzewo od prawej do lewej.

- **Parametry**

node	Biezacy wezel do wyswietlenia.
prefix	Prefiks (wciecie i linie) do wydrukowania przed wezlem.
isLeft	Informacja, czy biezacy wezel jest lewym dzieckiem.
isRoot	Informacja, czy biezacy wezel jest korzeniem calego drzewa.

4.1.3.7 displayInorder()

```
void BST::displayInorder ()
```

Wyswietla drzewo w kolejnosci in-order (LKP).

Wyswietla zawartosc drzewa w kolejnosci in-order (LKP).

4.1.3.8 displayPostorder()

```
void BST::displayPostorder ()
```

Wyswietla drzewo w kolejnosci post-order (LPK).

Wyswietla zawartosc drzewa w kolejnosci post-order (LPK).

4.1.3.9 displayPreorder()

```
void BST::displayPreorder ()
```

Wyswietla drzewo w kolejnosci pre-order (KLP).

Wyswietla zawartosc drzewa w kolejnosci pre-order (KLP).

4.1.3.10 findMin()

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

Prywatny pomocnik do znajdowania minimum w poddrzewie.

Znajduje wezel o minimalnej wartosci w danym poddrzewie.

- Przechodzi w lewo tak daleko, jak to mozliwe.

- **Parametry**

node	Wskaznik na korzen poddrzewa do przeszukania.

Zwraca

Wskaznik na wezel o minimalnej wartosci.

4.1.3.11 inorderRecursive()

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

Prywatny pomocnik rekursywny do wyswietlania in-order.

Rekursywny pomocnik do wyswietlania in-order (LKP).

- **Parametry**

node	Biezacy wezel do przetworzenia.

4.1.3.12 isEmpty()

```
bool BST::isEmpty ()
```

Sprawdza, czy drzewo jest puste.

*

Zwraca

true jesli puste, false w przeciwnym razie.

Zwraca

- true jesli drzewo jest puste (root == nullptr), false w przeciwnym razie.

4.1.3.13 loadFromFile()

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

Wczytuje drzewo z pliku binarnego.

*

Parametry

filename	Nazwa pliku wejsciowego.

- Drzewo jest czyszczone przed wczytaniem. Dane sa czytane jako sekwencja wartosci typu int i dodawane do drzewa.
- **Parametry**

filename	Nazwa pliku do odczytu.

4.1.3.14 postorderRecursive()

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

Prywatny pomocnik rekursywny do wyswietlania post-order.

Rekursywny pomocnik do wyswietlania post-order (LPK).

- **Parametry**

node	Biezacy wezel do przetworzenia.

4.1.3.15 preorderRecursive()

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

Prywatny pomocnik rekursywny do wyswietlania pre-order.

Rekursywny pomocnik do wyswietlania pre-order (KLP).

- **Parametry**

node	Biezacy wezel do przetworzenia.

4.1.3.16 remove()

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

Usuwa wezel o podanej wartosci.

Publiczna metoda usuwajaca wezel o podanej wartosci.

*

Parametry

value	Wartosc do usuniecia.

- Rozpoczyna rekursywne usuwanie od korzenia.

- **Parametry**

value	Wartosc do usuniecia z drzewa.

4.1.3.17 removeRecursive()

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

Prywatny pomocnik rekursywny do usuwania.

Rekursywny pomocnik do usuwania wezla o podanej wartosci.

- Przeszukuje drzewo w poszukiwaniu wartosci i usuwa ja, zachowujac wlasnosci drzewa BST.

- **Parametry**

node	Wskaznik na korzen poddrzewa, z ktorego usuwamy.
value	Wartosc do usuniecia.

Zwraca

Wskaznik na korzen zmodyfikowanego poddrzewa.

4.1.3.18 saveRecursive()

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

Prywatny pomocnik rekursywny do zapisu do pliku (pre-order).

Rekursywny pomocnik do zapisu drzewa do pliku (pre-order).

- Zapisuje wartosc biezacego wezla, nastepnie rekursywnie lewe i prawe poddrzewo.
- **Parametry**

node	Biezacy wezel do zapisu.
outFile	Strumien pliku wyjsciowego (binarny).

4.1.3.19 saveToFile()

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

Zapisuje drzewo binarnie do pliku (pre-order).

Zapisuje drzewo do pliku binarnego.

*

Parametry

filename	Nazwa pliku wyjsciowego.

- Uzywa kolejnosci pre-order do zapisu, co pozwala na odtworzenie tej samej struktury drzewa przy ponownym wczytaniu.
- **Parametry**

filename	Nazwa pliku do zapisu.

4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 root

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

Wskaznik na korzen drzewa.

Dokumentacja dla tej klasy została wygenerowana z plików:

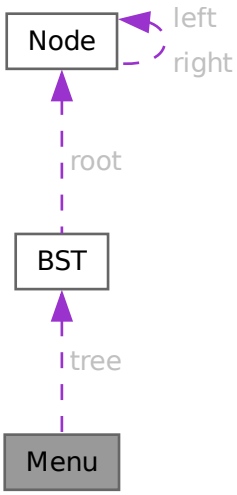
- BST.h
- BST.cpp

4.2 Dokumentacja klasy Menu

Klasa Menu zarzadzajaca interfejsem uzytkownika i operacjami na Drzewie BST.

```
#include <Menu.h>
```

Diagram współpracy dla Menu:



Metody publiczne

- Menu ()

  Konstruktor klasy Menu.
- ~Menu ()

  Destruktor klasy Menu.
- void run ()

  Uruchamia glowna petle programu i wyswietla menu glowne.

Metody prywatne

- void handleDodaj ()

  Obsluguje dodawanie nowego elementu do drzewa.
- void handleUsun ()

  Obsluguje usuwanie elementu z drzewa.
- void handleWyswietl ()

  Obsluguje wyswietlanie drzewa (wybor metody: graficzna, pre-, in-, post-order).
- void handleUsunWszystko ()

  Obsluguje usuniecie wszystkich elementow z drzewa.
- void handleZapiszDoPliku ()

  Obsluguje zapisanie aktualnego stanu drzewa do pliku.
- void handleWczytajZPliku ()

  Obsluguje wczytanie drzewa z pliku, zastepujac biezacy stan.

Atrybuty prywatne

- std::vector< std::pair< std::string, std::function< void()> > > list

  Lista opcji menu.
- BST tree

  Glowny obiekt Drzewa BST, na ktorym wykonywane sa operacje.

4.2.1 Opis szczegółowy

Klasa Menu zarzadzajaca interfejsem uzytkownika i operacjami na Drzewie BST.

Klasa obsluguje wyswietlanie glownego menu i podmenu, zbieranie danych wejsciowych od uzytkownika oraz wywolywanie odpowiednich metod na wewnetrznym obiekcie BST.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Menu()

```
Menu::Menu ()
```

Konstruktor klasy Menu.

- Inicjuje liste opcji menu.
- Inicjuje menu glowne, dodajac opcje zwiazane z operacjami na Drzewie BST.

4.2.2.2 ~Menu()

```
Menu::~Menu ()
```

Destruktor klasy Menu.

- Obecnie pusty, ale moze byc uzyty do zwolnienia zasobow w przyszlosci.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 handleDodaj()

```
void Menu::handleDodaj ()  [private]
```

Obsluguje dodawanie nowego elementu do drzewa.

Obsluguje opcje dodawania elementu do drzewa BST.

- Prosi uzytkownika o podanie wartosci i dodaje ja do wewnetrznego obiektu `tree`.

TODO: add data to tree

4.2.3.2 handleUsun()

```
void Menu::handleUsun ()  [private]
```

Obsluguje usuwanie elementu z drzewa.

Obsluguje opcje usuwania elementu z drzewa BST.

- Prosi uzytkownika o podanie wartosci i usuwa ja z wewnetrznego obiektu `tree`.

4.2.3.3 handleUsunWszystko()

```
void Menu::handleUsunWszystko ()  [private]
```

Obsluguje usuniecie wszystkich elementow z drzewa.

Obsluguje opcje usuwania calego drzewa BST.

- Wywoluje metode `clear()` na wewnetrznym obiekcie `tree`.

4.2.3.4 handleWczytajZPliku()

```
void Menu::handleWczytajZPliku ()  [private]
```

Obsluguje wczytanie drzewa z pliku, zastepujac biezacy stan.

Obsluguje opcje wczytywania drzewa BST z pliku.

- Prosi uzytkownika o podanie nazwy pliku i wywoluje `loadFromFile()` na obiekcie `tree`.

4.2.3.5 handleWyswietl()

```
void Menu::handleWyswietl ()  [private]
```

Obsluguje wyswietlanie drzewa (wybor metody: graficzna, pre-, in-, post-order).

Obsluguje opcje wyswietlania drzewa BST.

- Wyswietla podmenu, pozwalajac uzytkownikowi wybrac jedna z czterech metod wyswietlania: graficzne, pre-order, in-order lub post-order.

4.2.3.6 handleZapiszDoPliku()

```
void Menu::handleZapiszDoPliku ()  [private]
```

Obsluguje zapisanie aktualnego stanu drzewa do pliku.

Obsluguje opcje zapisywania drzewa BST do pliku.

- Prosi uzytkownika o podanie nazwy pliku i wywoluje `saveToFile()` na obiekcie `tree`.

4.2.3.7 run()

```
void Menu::run ()
```

Uruchamia glowna petle programu i wyswietla menu glowne.

Glowna petla programu.

- Kontynuuje dzialanie do momentu wybrania opcji 'Zamknij program'.
- Wyswietla menu glowne, obsluguje wybor uzytkownika i wywoluje odpowiednie funkcje obslugi. Kontynuuje dzialanie, dopoki uzytkownik nie wybierze opcji zamkniecia programu (0).

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 list

```
std::vector<std::pair<std::string, std::function<void()> > > Menu::list  [private]
```

Lista opcji menu.

- Kazda para przechowuje nazwe opcji (string) oraz funkcje do wywolania (std::function<void()>) w momencie wyboru opcji.

4.2.4.2 tree

```
BST Menu::tree  [private]
```

Glowny obiekt Drzewa BST, na ktorym wykonywane sa operacje.

Dokumentacja dla tej klasy została wygenerowana z plików:

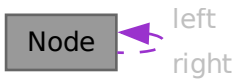
- Menu.h
- Menu.cpp

4.3 Dokumentacja struktury Node

Reprezentuje pojedynczy wezel w drzewie BST.

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

Diagram współpracy dla Node:



Metody publiczne

- Node (int value)

  Konstruktor wezla.

Atrybuty publiczne

- int data

  Wartosc przechowywana w wezle.
- Node * left

  Wskaznik na lewe dziecko.
- Node * right

  Wskaznik na prawe dziecko.

4.3.1 Opis szczegółowy

Reprezentuje pojedynczy wezel w drzewie BST.

- Przechowuje wartosc (data) oraz wskazniki na lewe i prawe dziecko.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 Node()

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

Konstruktor wezla.

Parametry

value	Wartosc do przechowania w nowym wezle.

4.3.3 Dokumentacja atrybutów składowych

4.3.3.1 data

```
int Node::data
```

Wartosc przechowywana w wezle.

4.3.3.2 left

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

Wskaznik na lewe dziecko.

4.3.3.3 right

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

Wskaznik na prawe dziecko.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- BST.h

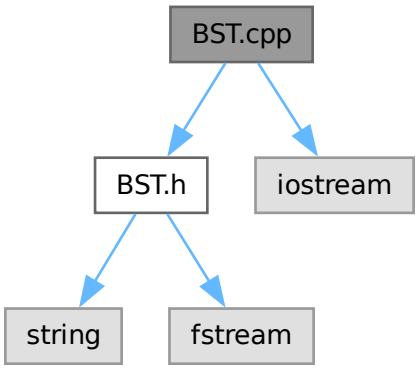
Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku BST.cpp

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

Wykres zależności załączania dla BST.cpp:



Definicje

- #define BST_CPP

5.1.1 Dokumentacja definicji

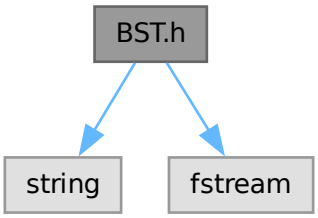
5.1.1.1 BST_CPP

```
#define BST_CPP
```

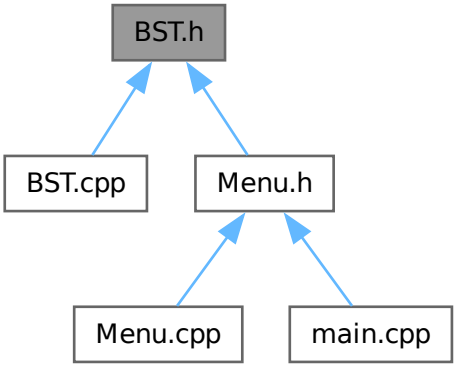
5.2 Dokumentacja pliku BST.h

```
#include <string>
#include <fstream>
```

Wykres zależności załączania dla BST.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- struct Node

  Reprezentuje pojedynczy wezel w drzewie BST.
- class BST

  Implementuje binarne drzewo poszukiwan (BST).

5.3 BST.h

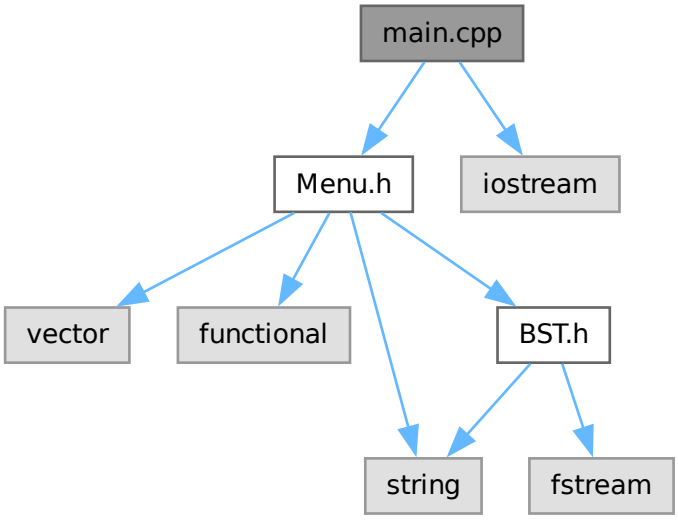
Idź do dokumentacji tego pliku.

```
00001 #ifndef BST_H
00002 #define BST_H
00003
00004 #include <string>
00005 #include <fstream>
00006
00007 using namespace std;
00008
00014 struct Node {
00015     int data;
00016     Node* left;
00017     Node* right;
00018
00023     Node(int value) : data(value), left(nullptr), right(nullptr) {}
00024 };
00025
00031 class BST {
00032 private:
00033     Node* root;
00034
00035     // --- Prywatne metody pomocnicze ---
00036     // Pelna dokumentacja znajduje sie w pliku implementacji (.cpp)
00037
00039     Node* removeRecursive(Node* node, int value);
00040
00042     Node* findMin(Node* node);
00043
00045     Node* addRecursive(Node* node, int value);
00046
00048     void saveRecursive(Node* node, ofstream& outFile);
00049
00051     void clearRecursive(Node* node);
00052
00054     void inorderRecursive(Node* node);
00055
00057     void preorderRecursive(Node* node);
00058
00060     void postorderRecursive(Node* node);
00061
00063     void displayGraphicalRecursive(Node* node, const string& prefix, bool isLeft, bool isRoot);
00064
00065 public:
00067     BST();
00068
00070     ~BST();
00071
00073     void clear();
00074
00078     bool isEmpty();
00079
00083     void add(int value);
00084
00088     void remove(int value);
00089
00091     void displayInorder();
00092
00094     void displayPreorder();
00095
00097     void displayPostorder();
00098
00100     void displayGraphical();
00101
00105     void saveToFile(const string& filename);
00106
00110     void loadFromFile(const string& filename);
00111
00112 };
00113
00114 #endif
```

5.4 Dokumentacja pliku main.cpp

```
#include "Menu.h"
#include <iostream>
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int main ()

5.4.1 Dokumentacja funkcji

5.4.1.1 main()

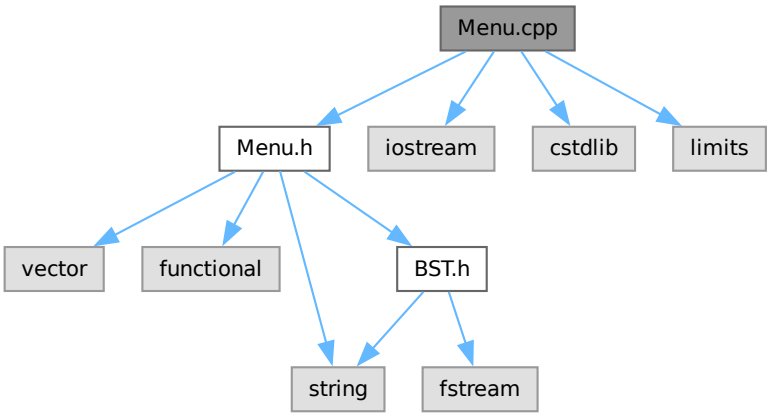
```
int main ()
```

5.5 Dokumentacja pliku Menu.cpp

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

```
#include <limits>
```

Wykres zależności załączania dla Menu.cpp:



Definicje

- #define MENU_CPP

Funkcje

- void clearScreen ()

  Funkcja pomocnicza do czyszczenia ekranu konsoli.
- void pause ()

  Wstrzymuje wykonanie programu do momentu nacisniecia klawisza Enter.

5.5.1 Dokumentacja definicji

5.5.1.1 MENU_CPP

```
#define MENU_CPP
```

5.5.2 Dokumentacja funkcji

5.5.2.1 clearScreen()

```
void clearScreen ()
```

Funkcja pomocnicza do czyszczenia ekranu konsoli.

- Uzywa "cls" dla systemow Windows i "clear" dla systemow uniksowych (Linux/macOS).

  Zobacz również

  https://medium.com/@ryan_forrester_/c-screen-clearing-how-to-guide-cff5bf764ccd

5.5.2.2 pause()

```
void pause ()
```

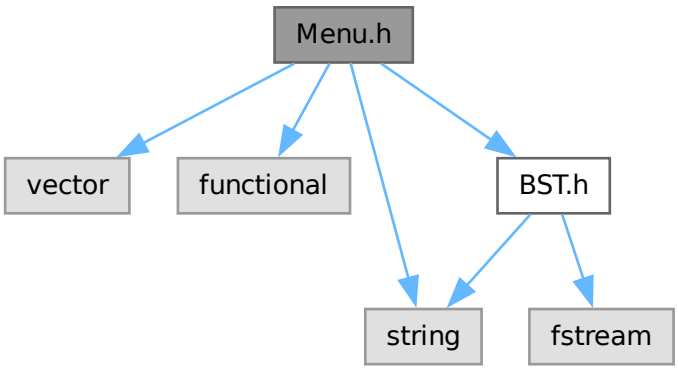
Wstrzymuje wykonanie programu do momentu nacisniecia klawisza Enter.

- Czyści bufor wejsciowy, aby zapewnic, ze oczekuje na swieze nacisniecie klawisza Enter.

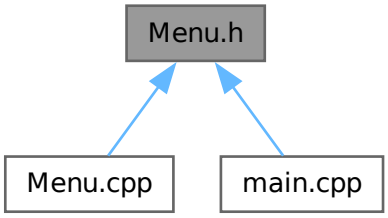
5.6 Dokumentacja pliku Menu.h

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

Wykres zależności załączania dla Menu.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class Menu

  Klasa Menu zarzadzajaca interfejsem uzytkownika i operacjami na Drzewie BST.

5.7 Menu.h

Idź do dokumentacji tego pliku.

```
00001 #ifndef MENU_H
00002 #define MENU_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <string>
00007 #include "BST.h"
00008
00015 class Menu {
00016 private:
00022     std::vector<std::pair<std::string, std::function<void()>> list;
00023
00027     BST tree;
00028
00032     void handleDodaj();
00033
00037     void handleUsun();
00038
00042     void handleWyswietl();
00043
00047     void handleUsunWszystko();
00048
00052     void handleZapiszDoPliku();
00053
00057     void handleWczytajZPliku();
00058
00059 public:
00064     Menu();
00065
00069     ~Menu();
00070
00075     void run();
00076 };
00077
00078 #endif
```

5.8 Dokumentacja pliku README.md

Skorowidz