Zad6 doxygen

Generated by Doxygen 1.12.0

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 documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 CSVLoader Class Reference

Klasa statyczna do ^{3}adowania danych z plików CSV do struktury **Tree** (p. 9).

`#include <CSVLoader.h>`

Static Public Member Functions

- static void **loadCSV** (const std::string &filePath, **Tree** &tree, **Logger** &logger)
  Wczytuje dane z pliku CSV do drzewa.

3.1.1 Detailed Description

Klasa statyczna do ^{3}adowania danych z plików CSV do struktury **Tree** (p. 9).

3.1.2 Member Function Documentation

3.1.2.1 loadCSV()

```
void CSVLoader::loadCSV (
            const std::string & filePath,
             Tree & tree,
             Logger & logger)  [static]
```

Wczytuje dane z pliku CSV do drzewa.

Wczytuje dane z pliku CSV i zapisuje je w strukturze **Tree** (p. 9).

Parameters

filePath	cie¿ka do pliku CSV.
tree	Referencja do obiektu klasy **Tree** (p. 9), gdzie dane zostan¹ wczytane.
logger	Referencja do obiektu klasy **Logger** (p. 6), u¿ywanego do rejestrowania zdarzeñ.

Funkcja wczytuje dane z podanego pliku CSV, pomijajÄc nagÅÃ^{3}wek. W przypadku poprawnych rekordÃ^{3}w dodaje je do drzewa. Niepoprawne rekordy sÄ rejestrowane jako bÅÄdy w loggerze.

Parameters

filePath	ÅcieÅ¼ka do pliku CSV.
tree	Referencja do obiektu klasy **Tree** (p. 9), gdzie dane zostanÄ zapisane.
logger	Referencja do obiektu klasy **Logger** (p. 6), uÅ¼ywanego do rejestrowania zdarzeÅ.

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

- **CSVLoader.h**
- **CSVLoader.cpp**

3.2 Logger Class Reference

Klasa odpowiedzialna za logowanie ró¿nych typów informacji do plików.

`#include <Logger.h>`

Public Member Functions

- void **logData** (const std::string &message)
  Loguje podane dane do pliku tekstowego.
- void **logError** (const std::string &message)
  Loguje komunikaty b^3*êdów do pliku tekstowego.*
- void **logSummary** (int validCount, int invalidCount)
  Loguje podsumowanie danych do pliku tekstowego.

3.2.1 Detailed Description

Klasa odpowiedzialna za logowanie ró¿nych typów informacji do plików.

3.2.2 Member Function Documentation

3.2.2.1 logData()

```
void Logger::logData (
            const std::string & message)
```

Loguje podane dane do pliku tekstowego.

Parameters

message	Wiadomoæ do zapisania w pliku logowania.

Tworzy nowy plik o nazwie zawieraj¹cej aktualny czas i zapisuje do niego wiadomoæ.

Parameters

message	Wiadomoæ do zapisania w pliku logowania.

3.2.2.2 logError()

```
void Logger::logError (
            const std::string & message)
```

Loguje komunikaty b^{3}êdów do pliku tekstowego.

Parameters

message	Wiadomoæ o b^{3}êdzie do zapisania w pliku.

Tworzy nowy plik o nazwie zawieraj¹cej aktualny czas i zapisuje do niego wiadomoæ o b^{3}êdzie.

Parameters

message	Wiadomoæ o b^{3}êdzie do zapisania w pliku.

3.2.2.3 logSummary()

```
void Logger::logSummary (
            int validCount,
            int invalidCount)
```

Loguje podsumowanie danych do pliku tekstowego.

Parameters

validCount	Liczba poprawnych rekordów.
invalidCount	Liczba b^{3}ênych rekordów.

Tworzy nowy plik o nazwie zawieraj¹cej aktualny czas i zapisuje do niego liczbê poprawnych i b^{3}ênych rekordów.

Parameters

validCount	Liczba poprawnych rekordów.
invalidCount	Liczba b^{3}ênych rekordów.

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

- **Logger.h**
- **Logger.cpp**

3.3 Record Class Reference

Klasa reprezentuj¹ca pojedynczy zapis danych energetycznych.

`#include <Record.h>`

Public Member Functions

- **Record** (const std::string & **dateTime**, double **autoConsumption**, double **exportEnergy**, double **importEnergy**, double **consumption**, double **production**)

  *Konstruktor klasy **Record** (p. 8).*

Static Public Member Functions

- static **Record** * **parseRecord** (const std::string &line, bool &isValid)

  *Parsuje liniê tekstu na obiekt **Record** (p. 8).*

Public Attributes

- std::string **dateTime**

  Data i czas zapisu.
- double **autoConsumption**

  Wartoæ auto-konsumpcji energii.
- double **exportEnergy**

  Iloæ energii eksportowanej.
- double **importEnergy**

  Iloæ energii importowanej.
- double **consumption**

  £¹czne zu¿ycie energii.
- double **production**

  £¹czna produkcja energii.

3.3.1 Detailed Description

Klasa reprezentuj¹ca pojedynczy zapis danych energetycznych.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Record()

```
Record::Record (
            const std::string & dateTime,
            double autoConsumption,
            double exportEnergy,
            double importEnergy,
            double consumption,
            double production)
```

Konstruktor klasy **Record** (p. 8).

Parameters

dateTime	Data i czas.
autoConsumption	Auto-konsumpcja energii.
exportEnergy	Iloæ eksportowanej energii.
importEnergy	Iloæ importowanej energii.
consumption	£¹czne zu¿ycie energii.
production	£¹czna produkcja energii.

3.3.3 Member Function Documentation

3.3.3.1 parseRecord()

```
Record * Record::parseRecord (
            const std::string & line,
            bool & isValid)  [static]
```

Parsuje liniê tekstu na obiekt **Record** (p. 8).

Parameters

line	Linia tekstu do sparsowania.
isValid	Referencja do zmiennej, która wskazuje czy parsowanie by^{3}o udane.

Returns

Wskanik do nowego obiektu **Record** (p. 8), lub nullptr jeli parsowanie nie powiod^{3}o siê.

Parameters

line	Linia tekstu zawieraj¹ca dane w odpowiednim formacie.
isValid	Referencja do zmiennej, która wskazuje czy parsowanie by^{3}o udane.

Returns

Wskanik do nowego obiektu **Record** (p. 8), lub nullptr jeli parsowanie nie powiod^{3}o siê.

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

- **Record.h**
- **Record.cpp**

3.4 Tree Class Reference

Drzewo hierarchiczne przechowuj¹ce rekordy na podstawie daty.

```
#include <Tree.h>
```

Public Member Functions

- **Tree** ()

  Konstruktor inicjalizuj¹cy korzeñ drzewa.
- ∼**Tree** ()

  Destruktor usuwaj¹cy drzewo.
- void **insertRecord** (**Record** ∗record)

  Wstawia rekord do drzewa w odpowiednie miejsce na podstawie daty.
- std::vector< **Record** ∗ > **query** (const std::string &start, const std::string &end)

  Wyszukuje rekordy w zadanym przedziale dat.

3.4.1 Detailed Description

Drzewo hierarchiczne przechowuj¹ce rekordy na podstawie daty.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 ∼Tree()

```
Tree::∼Tree ()
```

Destruktor usuwaj¹cy drzewo.

Destruktor usuwaj¹cy drzewo i wszystkie jego wêz^{3}y.

3.4.3 Member Function Documentation

3.4.3.1 insertRecord()

```
void Tree::insertRecord (
            Record * record)
```

Wstawia rekord do drzewa w odpowiednie miejsce na podstawie daty.

Parameters

record	Wskanik na rekord do dodania.

3.4.3.2 query()

```
std::vector<  Record * > Tree::query (
            const std::string & start,
            const std::string & end)
```

Wyszukuje rekordy w zadanym przedziale dat.

Parameters

start	Data pocz¹tkowa (w formacie "YYYY-MM-DD").
end	Data koñcowa (w formacie "YYYY-MM-DD").

Returns

Wektor wskaników na rekordy w przedziale dat.

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

- Tree.h
- Tree.cpp

3.5 TreeNode Class Reference

Wêze^3 drzewa przechowuj¹cy dzieci oraz powi¹zane rekordy.

`#include <Tree.h>`

Public Member Functions

- ∼**TreeNode** ()

  Destruktor usuwaj¹cy dzieci i rekordy.

Public Attributes

- std::map< std::string, **TreeNode** ∗ > **children**

  Mapa dzieci wêz^{3}a, gdzie kluczem jest nazwa (np. rok, miesi¹c, dzieñ).
- std::vector< **Record** ∗ > **records**

  Wektor wskaników na rekordy zwi¹zane z tym wêz^{3}em.

3.5.1 Detailed Description

Wêze^3 drzewa przechowuj¹cy dzieci oraz powi¹zane rekordy.

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

- Tree.h

Chapter 4

File Documentation

4.1 CSVLoader.cpp File Reference

Implementacja klasy **CSVLoader** (p. 5), odpowiedzialnej za ładowanie danych z plików CSV.

```
#include "CSVLoader.h"
#include <fstream>
#include <sstream>
#include <ctime>
```

4.1.1 Detailed Description

Implementacja klasy **CSVLoader** (p. 5), odpowiedzialnej za ładowanie danych z plików CSV.

4.2 CSVLoader.h File Reference

Deklaracja klasy **CSVLoader** (p. 5), odpowiedzialnej za ^{3}adowanie danych z plików CSV.

```
#include "Tree.h"
#include "Logger.h"
#include <string>
```

Classes

- class **CSVLoader**

  *Klasa statyczna do ^{3}adowania danych z plików CSV do struktury **Tree** (p. 9).*

4.2.1 Detailed Description

Deklaracja klasy **CSVLoader** (p. 5), odpowiedzialnej za ^{3}adowanie danych z plików CSV.

4.3 CSVLoader.h

Go to the documentation of this file.

```
00001
00006 #ifndef CSVLOADER_H
00007 #define CSVLOADER_H
00008
00009 #include "Tree.h"
00010 #include "Logger.h"
00011 #include <string>
00012
00017 class CSVLoader {
00018 public:
00026     static void loadCSV(const std::string& filePath, Tree& tree, Logger& logger);
00027 };
00028
00029 #endif // CSVLOADER_H
```

4.4 Logger.cpp File Reference

Implementacja klasy **Logger** (p. 6) do obs^{3}ugi logowania danych, b^{3}êdów i podsumowañ.

```
#include "Logger.h"
#include <fstream>
#include <ctime>
```

4.4.1 Detailed Description

Implementacja klasy **Logger** (p. 6) do obs^{3}ugi logowania danych, b^{3}êdów i podsumowañ.

4.5 Logger.h File Reference

Deklaracja klasy **Logger** (p. 6) do obs^{3}ugi logowania danych, b^{3}êdów i podsumowañ.

```
#include <string>
```

Classes

- class **Logger**

  Klasa odpowiedzialna za logowanie ró¿nych typów informacji do plików.

4.5.1 Detailed Description

Deklaracja klasy **Logger** (p. 6) do obs^{3}ugi logowania danych, b^{3}êdów i podsumowañ.

4.6 Logger.h

Go to the documentation of this file.

```
00001
00006 #ifndef LOGGER_H
00007 #define LOGGER_H
00008
00009 #include <string>
00010
00015 class Logger {
00016 public:
00021     void logData(const std::string& message);
00022
00027     void logError(const std::string& message);
00028
00034     void logSummary(int validCount, int invalidCount);
00035 };
00036
00037 #endif // LOGGER_H
```

4.7 Record.cpp File Reference

Implementacja metod klasy **Record** (p. 8).

```
#include "Record.h"
```

4.7.1 Detailed Description

Implementacja metod klasy **Record** (p. 8).

4.8 Record.h File Reference

Deklaracja klasy **Record** (p. 8) do przechowywania danych energetycznych i ich parsowania.

```
#include <string>
#include <sstream>
```

Classes

- class **Record**

  Klasa reprezentuj¹ca pojedynczy zapis danych energetycznych.

4.8.1 Detailed Description

Deklaracja klasy **Record** (p. 8) do przechowywania danych energetycznych i ich parsowania.

4.9 Record.h

Go to the documentation of this file.

```
00001
00006 #ifndef RECORD_H
00007 #define RECORD_H
00008
00009 #include <string>
00010 #include <sstream>
00011
00016 class Record {
00017 public:
00021     std::string dateTime;
00022
00026     double autoConsumption;
00027
00031     double exportEnergy;
00032
00036     double importEnergy;
00037
00041     double consumption;
00042
00046     double production;
00047
00058     Record(const std::string& dateTime, double autoConsumption, double exportEnergy,
00059         double importEnergy, double consumption, double production);
00060
00068     static Record* parseRecord(const std::string& line, bool& isValid);
00069 };
00070
00071 #endif // RECORD_H
```

4.10 Tree.h

```
00001 #ifndef TREE_H
00002 #define TREE_H
00003
00004 #include "Record.h"
00005 #include <map>
00006 #include <vector>
00007 #include <string>
00008
00013 class TreeNode {
00014 public:
00018     std::map<std::string, TreeNode*> children;
00019
00023     std::vector<Record*> records;
00024
00028     ~TreeNode() {
00029         for (auto& pair : children)
00030             delete pair.second;
00031         for (auto* record : records)
00032             delete record;
00033     }
00034 };
00035
00040 class Tree {
00041 private:
00045     TreeNode* root;
00046
00047 public:
00051     Tree();
00052
00056     ~Tree();
00057
00062     void insertRecord(Record* record);
00063
00070     std::vector<Record*> query(const std::string& start, const std::string& end);
00071 };
00072
00073 #endif // TREE_H
```

4.11 zad6.cpp File Reference

Główny plik programu odpowiedzialny za obsługę procesu wczytywania danych z pliku CSV oraz interakcji użytkownika.

```
#include "CSVLoader.h"
#include "Tree.h"
#include "Logger.h"
#include <iostream>
```

Functions

- int **main** ()

  Główna funkcja programu.

4.11.1 Detailed Description

Główny plik programu odpowiedzialny za obsługę procesu wczytywania danych z pliku CSV oraz interakcji użytkownika.

Zawiera główną funkcję programu, która inicjalizuje strukturę drzewa i loggera, a następnie pozwala użytkownikowi wczytać dane z pliku CSV.

4.11.2 Function Documentation

4.11.2.1 main()

```
int main ()
```

Główna funkcja programu.

Funkcja główna inicjalizuje obiekty **Tree** (p. 9) oraz **Logger** (p. 6), pobiera ścieżkę pliku CSV od użytkownika i wywołuje funkcję wczytującą dane z tego pliku do struktury drzewa. Po zakończeniu informuje użytkownika o zakończeniu procesu wczytywania danych.

Returns

Zwraca 0 po poprawnym zakończeniu działania programu.

< Struktura drzewa, do której wczytywane są dane z pliku CSV.

< **Logger** (p. 6) odpowiedzialny za zapisywanie logów działania programu.

< Ścieżka do pliku CSV podana przez użytkownika.

< Wczytanie danych z pliku CSV do drzewa.

Index