Politechnika Śląska w Gliwicach

Wydział Automatyki, Elektroniki
i Informatyki

Laboratorium Programowania Komputerów 3

Tablica Sufiksów

Autor	Mikołaj Smoła
Prowadzący	dr inż. Jolanta Kawulok
Rok akademicki	2020/2021
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	3
Grupa	4
Data sporządzenia sprawozdania	12.11.2020

1. Temat projektu

Program na podstawie ładowanych plików zawierających tekst i poszukiwane wzorce, tworzy posortowaną tablicę sufiksów i wyszukuje w niej załadowane wzorce, tworząc na końcu plik wynikowy ze wzorcami i ich wystąpieniami w tekście.

2. Analiza zadania

Celem zadania jest stworzenie programu, który pobiera dwa pliki w formacie *.txt: plik z tekstem oraz plik z poszukiwanymi wzorcami. Na podstawie pliku z tekstem program tworzy tablicę sufiksów i sortuje ją. Następnie przy użyciu algorytmu poszukiwania binarnego szuka wzorców w tablicy i zapisuje ich wystąpienia. Na końcu generowany jest plik wynikowy w formacie .txt z informacją o wystąpieniach dla każdego z załadowanych wzorców.

2.1 Struktury danych

Program korzysta z następujących struktur danych:

- Generycznej listy jednokierunkowej przechowującej między innymi pobrane z pliku wzorce oraz informacje o wystąpieniach danych wzorców w tablicy sufiksów, umożliwiająca ich zapis do pliku wynikowego.

Na potrzeby programu wyodrębniono dodatkowe następujące struktury danych:

- SuffixTable – przechowująca wyznaczoną tablicę sufiksów,
- LCPTable – struktura, która w założeniu przechowująca wyznaczoną tablicę LCP, w związku z tym, że funkcjonalność związana z algorytmem wyszukiwania wzorców jest opcjonalna, struktura danych LCPTable nie została użyta w programie,
- PatternOccurrence – struktura przechowująca dane wyjściowe wystąpień wzorców w postaci zgodnej ze strukturą pliku wyjściowego.

2.2 Algorytmy

Przez $SUBWORDS(x)$ oznaczamy niepusty, posortowany leksykograficznie zbiór wszystkich podsłów danego tekstu x. Kolejne indeksy podsłów ze zbioru $SUBWORDS(x)$ stanowią tablicę sufiksów $SUF(x)$. Litery alfabetu danego podsłowa o długości n oznaczamy jako $WORD(a_1\ a_2\ \ldots\ a_n)$, gdzie $SUBWORDS(x) \in WORD(a_1\ a_2\ \ldots\ a_n)$. Sufiksem $SUFIKS_i$ zaczynającym się na pozycji i-tej tekstu x, podsłowa $WORD$ i długości l oznaczamy taki ciąg znaków, który stanowi prefiks słowa $WORD$, co oznacza, ze jego litery alfabetu są dokładnie takie same jak początkowe l liter słowa $WORD$.

Tablice sufiksów $SUF(x)$ otrzymano poprzez wyznaczenie indeksów podsłów ze zbioru $SUBWORS(x)$ danego tekstu x. Algorytm wyszukiwania wzorca y w tekście x opiera się na wyszukiwaniu binarnym takich podsłów ze zbioru $SUBWORDS(x)$, których prefiks danego podsłowa $WORD$ jest równy względem wyszukiwanego wzorca y. Cecha zbioru $SUBWORDS(x)$ polegająca na posortowaniu leksykograficznym pozwoliła wykorzystać strategię wyszukiwania binarnego.

Specyfikacja zewnętrzna

Program obsługiwany jest z poziomu wiersza poleceń.

Przy uruchamianiu programu należy podać komendę stworzoną z trzech kolejnych. Pierwsza to dokładna ścieżka do pliku z tekstem wejściowym podawana z użyciem przełącznika -t. Przykładowa komenda:

```
-t C:\Users\User\git\SuffixTable\textFile.txt
```

Użytkownik podaje także dokładną ścieżkę do pliku ze wzorcami używając przełącznika `-p`, na przykład:

```
-p C:\Users\User\git\SuffixTable\patternFile.txt
```

Na końcu z użyciem przełącznika `-o` podajemy ścieżkę gdzie ma zostać zapisany plik wynikowy, przykładowo:
`-o C:\Users\User\git\SuffixTable`

Wszystkie trzy są oddzielane spacjami i tworzą jeden ciąg.
W razie wystąpienia jakichkolwiek błędów lub w przypadku podania przez użytkownika nieprawidłowych danych, program wyświetli odpowiednie powiadomienie.
Dla uzyskania właściwych wyników, pliki wejściowe nie powinny zawierać polskich znaków.
W wyniku pracy programu zostaje wygenerowany plik `resultFile.txt` w podanej wcześniej lokalizacji na dysku, o czym program poinformuje komunikatem na konsoli, po czym program kończy pracę.

Przykłady poprawnych zawartości plików:

- Zawartość pliku tekstowego do przeszukania:

```
she sells sea shell
```

- Zawartość pliku ze wzorcami:

```
se
set
he
he
```

- Zawartość pliku wynikowego:

```
se: 10, 4
set: brak
he : brak
he: 1, 14
```

3. Specyfikacja wewnętrzna

Podczas implementacji programu zastosowano wzorzec projektowy *Strategia*. Oznacza to, że w projekcie możemy wyodrębnić poszczególne algorytmy mające wzajemne powiązania. Model ten umożliwia łatwą rozbudowę programu o kolejne algorytmy, a także edycję lub wymianę już istniejących.

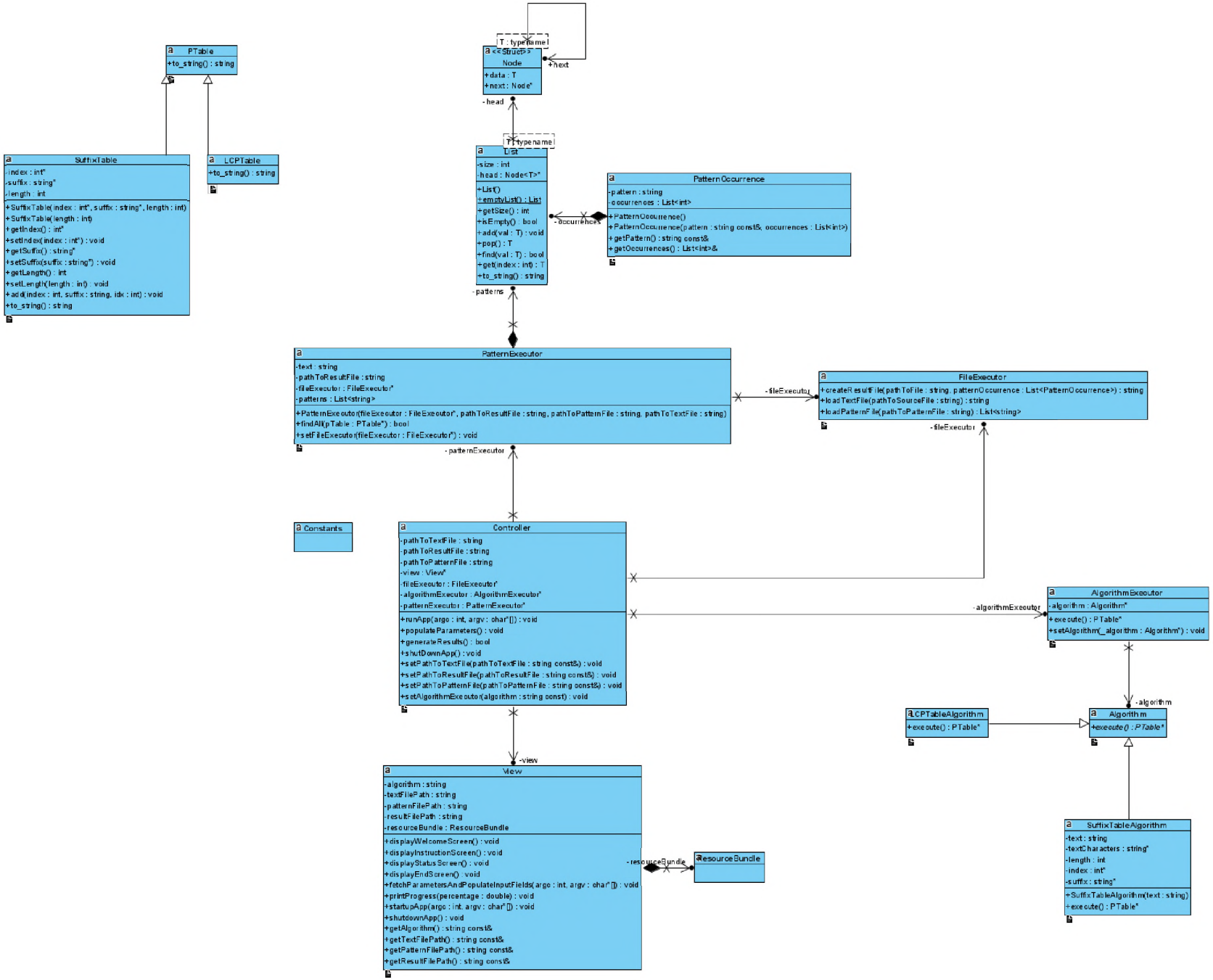
Struktura programu jest zgodna ze wzorem architektonicznym MVC (Model-View-Controller), zgodnie z którym kod aplikacji podzielony jest na trzy następujące części:

- Model – zawiera struktury danych oraz reużywalne, niezależne implementacje algorytmów,
- Widok – składa się na zaimplementowany interfejs użytkownika,
- Kontroler – stanowi rodzaj spoiwa pomiędzy częścią Widoku i Modelu oraz zawiera implementacje głównych funkcji programu.

W implementacji listy jednokierunkowej został wykorzystany typ szablonowy, który umożliwia tworzenie list dowolnego typu i wykonywanie na nich predefiniowanych operacji. Uniwersalność tego typu pozwala uniknąć redundancji kodu.

3.1 Ogólna struktura programu

Struktura przedstawiona jest na diagramie na następnej stronie.


PTable
+to_string() : string
SuffixTable
-index : int*
-suffix : string*
-length : int
+SuffixTable(index : int*, suffix : string*, length : int)
+SuffixTable(length : int)
+getIndex() : int*
+setIndex(index : int*) : void
+getSuffix() : string*
+setSuffix(suffix : string*) : void
+getLength() : int
+setLength(length : int) : void
+add(index : int, suffix : string, idx : int) : void
+to_string() : string
LCPTable
+to_string() : string
T : typename
<<Struct>>
Node
+data : T
+next : Node*
+next
-head
T : typename
List
-size : int
-head : Node<T>*
+List()
+emptyList() : List
+getSize() : int
+isEmpty() : bool
+add(val : T) : void
+pop() : T
+find(val : T) : bool
+get(index : int) : T
+to_string() : string
-occurrences
PatternOccurrence
-pattern : string
-occurrences : List<int>
+PatternOccurrence()
+PatternOccurrence(pattern : string const&, occurrences : List<int>)
+getPattern() : string const&
+getOccurrences() : List<int>&
-patterns
PatternExecutor
-text : string
-pathToResultFile : string
-fileExecutor : FileExecutor*
-patterns : List<string>
+PatternExecutor(fileExecutor : FileExecutor*, pathToResultFile : string, pathToPatternFile : string, pathToTextFile : string)
+findAll(pTable : PTable*) : bool
+setFileExecutor(fileExecutor : FileExecutor*) : void
-fileExecutor
FileExecutor
+createResultFile(pathToFile : string, patternOccurrence : List<PatternOccurrence>) : string
+loadTextFile(pathToSourceFile : string) : string
+loadPatternFile(pathToPatternFile : string) : List<string>
-fileExecutor
-patternExecutor
Constants
Controller
-pathToTextFile : string
-pathToResultFile : string
-pathToPatternFile : string
-view : View*
-fileExecutor : FileExecutor*
-algorithmExecutor : AlgorithmExecutor*
-patternExecutor : PatternExecutor*
+runApp(argc : int, argv : char*[]) : void
+populateParameters() : void
+generateResults() : bool
+shutDownApp() : void
+setPathToTextFile(pathToTextFile : string const&) : void
+setPathToResultFile(pathToResultFile : string const&) : void
+setPathToPatternFile(pathToPatternFile : string const&) : void
+setAlgorithmExecutor(algorithm : string const) : void
-algorithmExecutor
AlgorithmExecutor
-algorithm : Algorithm*
+execute() : PTable*
+setAlgorithm(_algorithm : Algorithm*) : void
-algorithm
LCPTableAlgorithm
+execute() : PTable*
Algorithm
+execute() : PTable*
-view
View
-algorithm : string
-textFilePath : string
-patternFilePath : string
-resultFilePath : string
-resourceBundle : ResourceBundle
+displayWelcomeScreen() : void
+displayInstructionScreen() : void
+displayStatusScreen() : void
+displayEndScreen() : void
+fetchParametersAndPopulateInputFields(argc : int, argv : char*[]) : void
+printProgress(percentage : double) : void
+startupApp(argc : int, argv : char*[]) : void
+shutdownApp() : void
+getAlgorithm() : string const&
+getTextFilePath() : string const&
+getPatternFilePath() : string const&
+getResultFilePath() : string const&
-resourceBundle
ResourceBundle
SuffixTableAlgorithm
-text : string
-textCharacters : string*
-length : int
-index : int*
-suffix : string*
+SuffixTableAlgorithm(text : string)
+execute() : PTable*

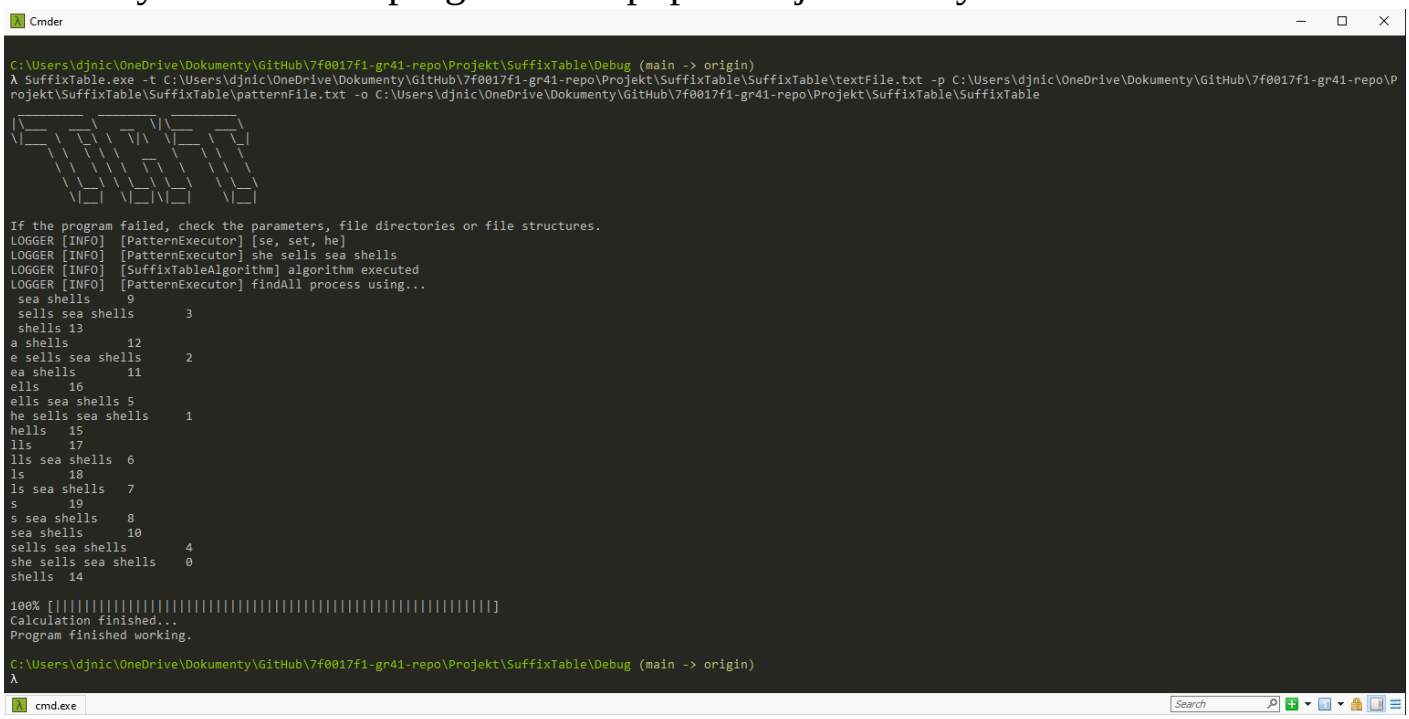
3.2 Szczegółowy opis typów i funkcji

Szczegółowy opis zawarty jest w dokumentacji Doxygen zawartej w załączniku na końcu sprawozdania.

4. Testowanie

Program został przetestowany dla plików z różnymi zawartościami.

Wynik działania programu dla poprawnej komendy:



Sygnalizacja problemu z plikami wejściowymi:

5. Wnioski

Celem projektu była implementacja programu wyszukującego wzorce w podanym przez użytkownika tekście. Cel projektu został zrealizowany. Aplikacja może być rozwijana o kolejne algorytmy dzięki implementacji wykorzystującej wzorce projektowe. Implementacja projektu była dla mnie możliwością do poszerzenia wiedzy na temat technik programowania w języku C++ i algorytmów analizy tekstów.

Załącznik – dokumentacja programu wygenerowana w Doxygen

Tablica Sufiksow

Generated by Doxygen 1.8.20

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

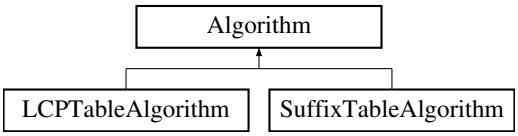
Chapter 3

Class Documentation

3.1 Algorithm Class Reference

`#include <Algorithm.h>`

Inheritance diagram for Algorithm:



Public Member Functions

- virtual PTable * **execute** ()=0

3.1.1 Detailed Description

Zamwiera czysto wirtualna metode bazowa execute implementowana przez klasy pochodne

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

- Algorithm.h

3.2 AlgorithmExecutor Class Reference

`#include <AlgorithmExecutor.h>`

Public Member Functions

- PTable * **execute** ()
- void **setAlgorithm** (Algorithm *_algorithm)

3.2.1 Detailed Description

Funkcja wybierajaca algorytm tablicy sufiksow

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

- AlgorithmExecutor.h
- AlgorithmExecutor.cpp

3.3 Constants Class Reference

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

- Constants.h

3.4 Controller Class Reference

`#include <Controller.h>`

Public Member Functions

- void runApp (int argc, char ∗argv[])
- void populateParameters ()
- bool generateResults ()
- void shutDownApp ()
- void **setPathToTextFile** (const std::string &pathToTextFile)
- void **setPathToResultFile** (const std::string &pathToResultFile)
- void **setPathToPatternFile** (const std::string &pathToPatternFile)
- void **setAlgorithmExecutor** (const std::string algorithm)

3.4.1 Detailed Description

Klasa modulu kontroli programu

Parameters

pathToTextFile	sciezka do pliku z tekstem
pathToResultFile	sciezka do pliku wynikowego
pathToPatternFile	sciezka do pliku ze wzorcami

3.4.2 Member Function Documentation

3.4.2.1 generateResults()

```
bool Controller::generateResults ( )
```

Metoda generowania wynikow

3.4.2.2 populateParameters()

```
void Controller::populateParameters ( )
```

Metoda przesylajaca niezbedne parametry pracy dla programu

3.4.2.3 runApp()

```
void Controller::runApp (
            int argc,
            char * argv[] )
```

Metoda rozpoczynajaca dzialanie programu

3.4.2.4 shutDownApp()

```
void Controller::shutDownApp ( )
```

Metoda konczenia pracy programu

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

- Controller.h
- Controller.cpp

3.5 FileExecutor Class Reference

```
#include <FileExecutor.h>
```

Public Member Functions

- std::string createResultFile (std::string pathToFile, List< PatternOccurrence > patternOccurrence)
- std::string loadTextFile (std::string pathToSourceFile)
- List< std::string > loadPatternFile (std::string pathToPatternFile)

3.5.1 Detailed Description

Klasa zarzadzajaca plikami

3.5.2 Member Function Documentation

3.5.2.1 createResultFile()

```
std::string FileExecutor::createResultFile (
            std::string pathToFile,
            List< PatternOccurrence > patternOccurrence )
```

Metoda tworzaca plik wynikowy

Parameters

pathToFile	sciezka do pliku wynikowego
patternOccurrence	lista przechowujaca wystapienia danego wzorca

3.5.2.2 loadPatternFile()

```
List< std::string > FileExecutor::loadPatternFile (
            std::string pathToPatternFile )
```

Metoda ladujaca plik ze wzorcami

Parameters

pathToPatternFile	sciezka do pliku ze wzorcami

3.5.2.3 loadTextFile()

```
std::string FileExecutor::loadTextFile (
            std::string pathToSourceFile )
```

Metoda ladujaca plik z tekstem do przeszukania

Parameters

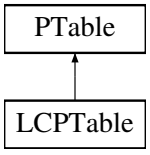
pathToSourceFile	sciezka do pliku z tekstem wejsciowym

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

- FileExecutor.h
- FileExecutor.cpp

3.6 LCPTable Class Reference

Inheritance diagram for LCPTable:



Public Member Functions

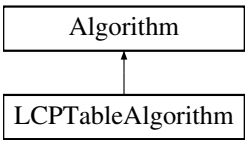
- std::string **to_string** ()

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

- LCPTable.h
- LCPTable.cpp

3.7 LCPTableAlgorithm Class Reference

Inheritance diagram for LCPTableAlgorithm:



Public Member Functions

- PTable * **execute** ()

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

- LCPTableAlgorithm.h
- LCPTableAlgorithm.cpp

3.8 List< T > Class Template Reference

`#include <List.h>`

Public Member Functions

- int **getSize** () const
- bool **isEmpty** () const
- void **add** (T val)
- T **pop** ()
- bool **find** (T val)
- T **get** (int index)
- std::string **to_string** ()

Static Public Member Functions

- static List **emptyList** ()

3.8.1 Detailed Description

template<typename T>
class List< T >

Lista (typ szablonowy)

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

- List.h

3.9 Node< T > Struct Template Reference

Public Attributes

- T **data**
- Node * **next**

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

- List.h

3.10 PatternExecutor Class Reference

`#include <PatternExecutor.h>`

Public Member Functions

- **PatternExecutor** (FileExecutor *fileExecutor, std::string pathToResultFile, std::string pathToPatternFile, std::string pathToTextFile)
- bool findAll (PTable *pTable)
- void **setFileExecutor** (FileExecutor *fileExecutor)

3.10.1 Detailed Description

Klasa obslugujaca wykonanie algorytmu poszukiwania

Parameters

patterns	Lista poszukiwanych wzorcow
text	Przesylany tekst
pathToResultFile	Sciezka do pliku wynikowego

3.10.2 Member Function Documentation

3.10.2.1 findAll()

```
bool PatternExecutor::findAll (
            PTable * pTable )
```

Metoda nakazujaca wykonanie algorytmu

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

- PatternExecutor.h
- PatternExecutor.cpp

3.11 PatternOccurrence Class Reference

```
#include <PatternOccurrence.h>
```

Public Member Functions

- **PatternOccurrence** (const std::string &pattern, List< int > occurrences)
- const std::string & **getPattern** () const
- List< int > & **getOccurrences** ()

3.11.1 Detailed Description

Klasa zarzadzajaca danymi o wzorcach i ich wystapieniach

Parameters

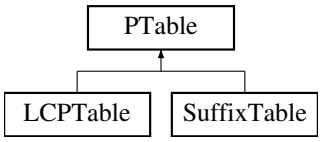
pattern	wzorzec
occurences	lista wystapien wzorca

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

- PatternOccurrence.h
- PatternOccurrence.cpp

3.12 PTable Class Reference

Inheritance diagram for PTable:



Public Member Functions

- virtual std::string **to_string** ()

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

- PTable.h
- PTable.cpp

3.13 ResourceBundle Class Reference

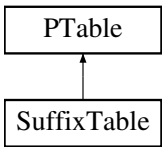
The documentation for this class was generated from the following file:

- ResourceBundle.h

3.14 SuffixTable Class Reference

`#include <SuffixTable.h>`

Inheritance diagram for SuffixTable:



Public Member Functions

- **SuffixTable** (int ∗index, std::string ∗suffix, int length)
- **SuffixTable** (int length)
- int ∗ **getIndex** () const
- void **setIndex** (int ∗index)
- std::string ∗ **getSuffix** () const
- void **setSuffix** (std::string ∗suffix)
- int **getLength** () const
- void **setLength** (int length)
- void add (int index, std::string suffix, int idx)
- std::string **to_string** ()

3.14.1 Detailed Description

Klasa zarzadzajaca tablica sufiksow

Parameters

index	tablica indeksow
suffix	tablica sufiksow
length	dlugosc tablicy sufiksow

3.14.2 Member Function Documentation

3.14.2.1 add()

```
void SuffixTable::add (
            int index,
            std::string suffix,
            int idx )
```

Metoda dodajaca do tablicy sufiksow

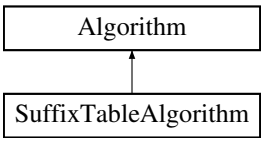
The documentation for this class was generated from the following files:

- SuffixTable.h
- SuffixTable.cpp

3.15 SuffixTableAlgorithm Class Reference

```
#include <SuffixTableAlgorithm.h>
```

Inheritance diagram for SuffixTableAlgorithm:



Public Member Functions

- SuffixTableAlgorithm (std::string text)
- PTable * **execute** ()

3.15.1 Detailed Description

Klasa algorytmu tworzenia tablicy sufiksParameters *text* Przesylany tekst

---

textCharacters Tablica do tworzenia sufiksow

---

length Dlugosc tablicy

---

suffix Sufiksy

3.15.2 Constructor & Destructor Documentation

3.15.2.1 SuffixTableAlgorithm()

```
SuffixTableAlgorithm::SuffixTableAlgorithm (
            std::string text )
```

Metoda rozpoczynajaca tworzenie tablicy sufiksow

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

- SuffixTableAlgorithm.h
- SuffixTableAlgorithm.cpp

3.16 View Class Reference

```
#include <View.h>
```

Public Member Functions

- void displayWelcomeScreen ()
- void displayInstructionScreen ()
- void displayStatusScreen ()
- void displayEndScreen ()
- void fetchParametersAndPopulateInputFields (int argc, char ∗argv[ ])
- void printProgress (double percentage)
- void startupApp (int argc, char ∗argv[ ])
- void shutdownApp ()
- const std::string & **getAlgorithm** () const
- const std::string & **getTextFilePath** () const
- const std::string & **getPatternFilePath** () const
- const std::string & **getResultFilePath** () const

3.16.1 Detailed Description

Klasa obslugujaca interfejs uzytkownika.

Parameters

algorithm	Informacja o wybranym algorytmie
textFilepath	Sciezka do pliku z tekstem
patternFilepath	Sciezka do pliku ze wzorcami
resultFilePath	Sciezka, gdzie ma byc stworzony plik wynikowy

3.16.2 Member Function Documentation

3.16.2.1 displayEndScreen()

```
void View::displayEndScreen ( )
```

Metoda wyswietlajaca tekst koncowy

3.16.2.2 displayInstructionScreen()

```
void View::displayInstructionScreen ( )
```

Metoda wyswietlajaca tekst instrukcji

3.16.2.3 displayStatusScreen()

```
void View::displayStatusScreen ( )
```

Metoda wyswietlajaca status programu

3.16.2.4 displayWelcomeScreen()

```
void View::displayWelcomeScreen ( )
```

Metoda wyswietlajaca tekst powitalny

3.16.2.5 fetchParametersAndPopulateInputFields()

```
void View::fetchParametersAndPopulateInputFields (
            int argc,
            char * argv[] )
```

Metoda przekazujaca polecenia wejsciowe programu

3.16.2.6 printProgress()

```
void View::printProgress (
            double percentage )
```

Metoda wyswietlajaca postep pracy programu

3.16.2.7 shutdownApp()

```
void View::shutdownApp ( )
```

Metoda wyswietlajaca komunikaty koncowe

3.16.2.8 startupApp()

```
void View::startupApp (
            int argc,
            char * argv[] )
```

Metoda pobierajaca polecenia wejsciowe programu

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

- View.h
- View.cpp