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.

1. **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.

* 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.
  1. **Algorytmy**

Przez oznaczamy niepusty, posortowany leksykograficznie zbiór wszystkich podsłów danego tekstu x. Kolejne indeksy podsłów ze zbioru stanowią tablicę sufiksów. Litery alfabetu danego podsłowa o długości n oznaczamy jako , gdzie Sufiksem zaczynającym się na pozycji -tej tekstu , podsłowa i długości oznaczamy taki ciąg znaków, który stanowi prefiks słowa , co oznacza, ze jego litery alfabetu są dokładnie takie same jak początkowe liter słowa .

Tablice sufiksów otrzymano poprzez wyznaczenie indeksów podsłów ze zbioru danego tekstu . Algorytm wyszukiwania wzorca w tekście opiera się na wyszukiwaniu binarnym takich podsłów ze zbioru , których prefiks danego podsłowa jest równy względem wyszukiwanego wzorca . Cecha zbioru 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

1. **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.

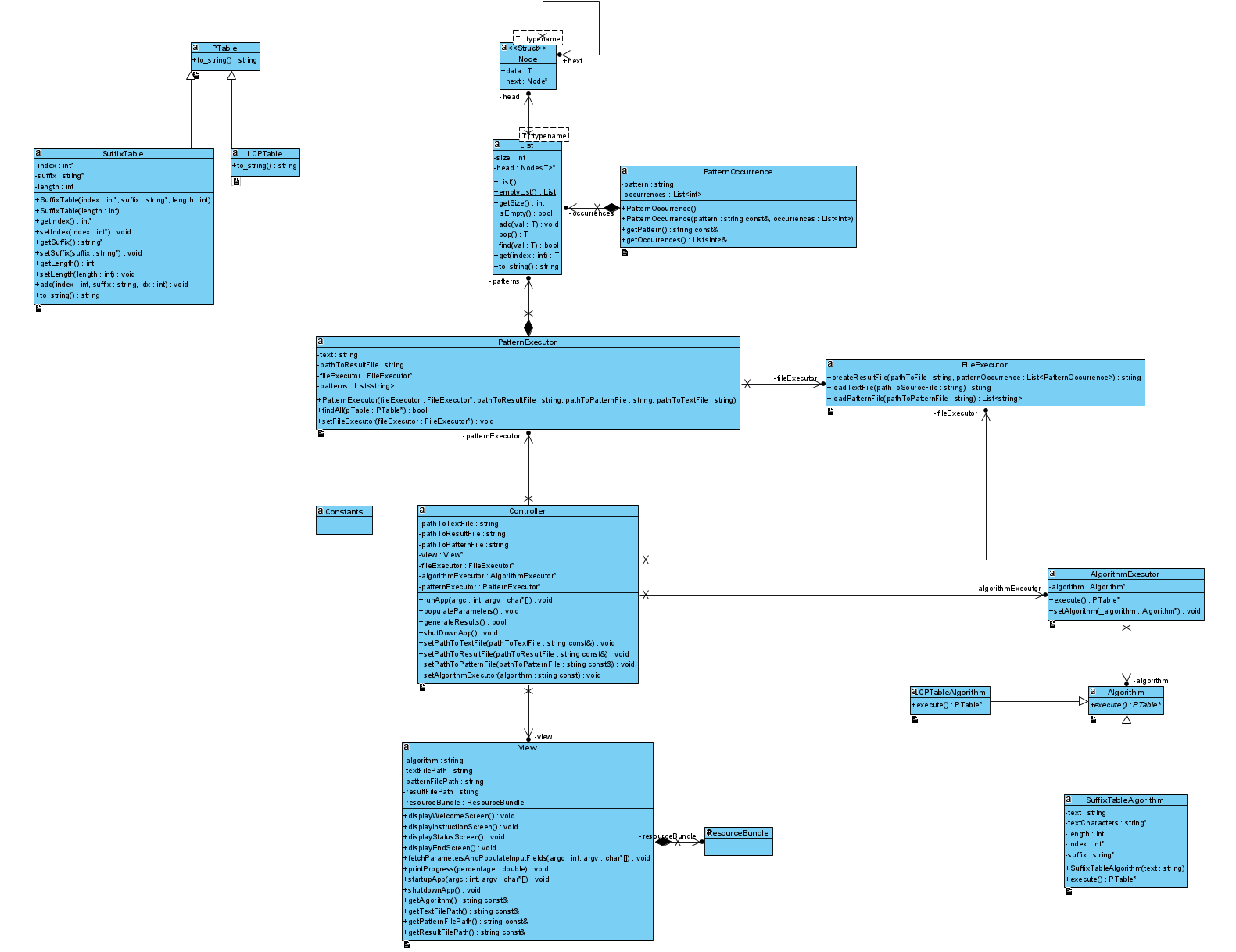
Struktura programu jest zgodna ze wzorem architektonicznym MVC (Model-View-Controller), zgodnie z którym kod aplikacji podzielony jest na trzy następujące części:

* 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.

* 1. **Ogólna struktura programu**

Struktura przedstawiona jest na diagramie na następnej stronie.

****

* 1. **Szczegółowy opis typów i funkcji**

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

1. **Testowanie**

Program został przetestowany dla plików z różnymi zawartościami.

Wynik działania programu dla poprawnej komendy:



Sygnalizacja problemu z plikami wejściowymi:



1. **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**