DOKUMENTACJA TECHNICZNA PROJEKTU

16 stycznia 2025

Spis treści

Rozdział 1

Opis techniczny projektu

1.1 Wprowadzenie

Niniejszy dokument opisuje działanie, strukturę oraz funkcjonalność programu napisanego w języku C++. Program służy do przetwarzania danych adresowych z plików CSV oraz bazy danych PostgreSQL. Jego główne zadania to:

- Wczytywanie danych z plików CSV.
- Generowanie nowych danych na podstawie bazy danych PostgreSQL.
- Usuwanie duplikatów współrzędnych geograficznych.
- Zapis przetworzonych danych do nowych plików CSV.

Program został zaprojektowany z myślą o przetwarzaniu dużych zbiorów danych, dlatego wspiera wielowątkowość, co znacząco przyspiesza operacje.

1.2 Opis ogólny kodu

Kod programu można podzielić na kilka głównych komponentów:

1. **Struktury danych:** Program definiuje strukturę `Address`, która przechowuje wszystkie informacje o pojedynczym rekordzie adresowym. Dodatkowo, struktura `FileInfo` przechowuje informacje o plikach CSV, takie jak ścieżka, indeks początkowy i liczba linii.
2. **Funkcje pomocnicze:** Zestaw funkcji, które realizują różne zadania, takie jak wyszukiwanie plików CSV, usuwanie ostatniego znaku z ciągu znaków czy identyfikowanie duplikatów współrzędnych.
3. **Funkcja główna:** Funkcja `processData`, która integruje wszystkie operacje, takie jak wczytywanie danych, generowanie nowych rekordów, usuwanie duplikatów oraz zapis przetworzonych danych.
4. **Wielowątkowość:** Operacje na dużych zbiorach danych, takie jak usuwanie duplikatów, są wykonywane w wielu wątkach, co pozwala na równoczesne przetwarzanie różnych fragmentów danych.
5. **Interfejs użytkownika:** Program oferuje menu w konsoli, które umożliwia użytkownikowi wybór odpowiednich działań, takich jak rozpoczęcie przetwarzania, sprawdzenie postępu lub wyjście z programu.

1.2.1 Ogólny przepływ działania programu

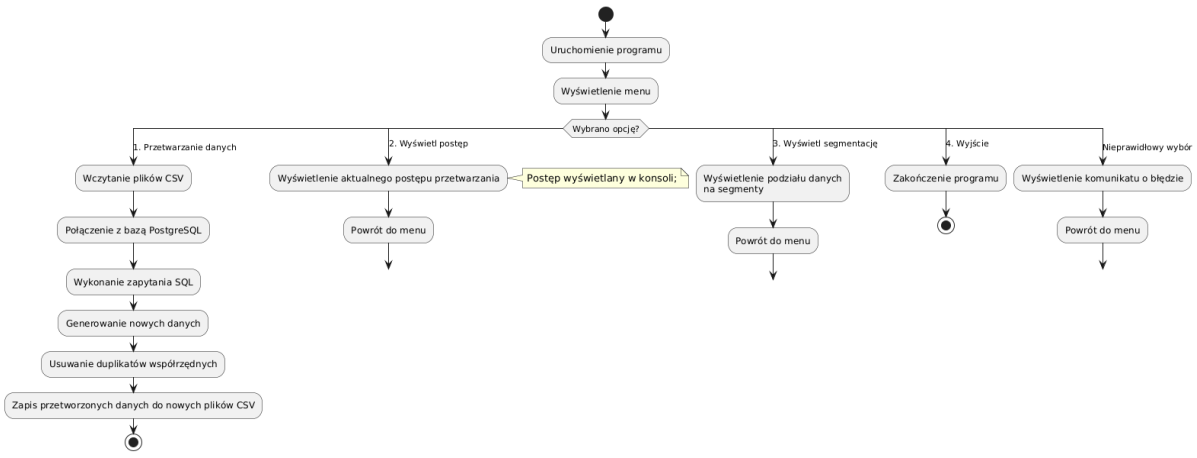
1. Program wczytuje dane z plików CSV znajdujących się w podanej ścieżce.
2. Łączy się z bazą danych PostgreSQL i wykonuje zapytanie SQL w celu wygenerowania nowych danych.
3. Usuwa rekordy zduplikowane na podstawie współrzędnych geograficznych.
4. Zapisuje przetworzone dane do nowych plików CSV z dopiskiem `_final`.
5. Użytkownik może śledzić postęp przetwarzania za pomocą menu w konsoli.

1.2.2 Opis diagramu aktywności

Diagram aktywności przedstawia przepływ programu, który działa w następujący sposób:

- Program rozpoczyna się od inicjalizacji i wyświetlenia menu.
- Użytkownik ma do wyboru jedną z czterech opcji:
    1. Przetwarzanie danych: Wczytanie plików, połączenie z bazą danych, wykonanie zapytań SQL, generowanie nowych danych, usuwanie duplikatów i zapis do nowych plików CSV.
    2. Wyświetlenie postępu: Prezentacja aktualnego stanu przetwarzania danych.
    3. Wyświetlenie segmentacji: Podział danych na segmenty i ich prezentacja.
    4. Wyjście: Zakończenie działania programu.
- W przypadku wybrania nieprawidłowej opcji użytkownik wraca do menu.

Na Ryc. 1.1 przedstawiono diagram aktywności ilustrujący powyższy proces.



Rysunek 1.1: Diagram aktywności programu.

1.3 Struktura kodu

Kod programu został podzielony na następujące sekcje:

1.3.1 Struktury danych

- `Address`: Reprezentuje pojedynczy rekord adresowy. Posiada następujące pola:
  - `lp` - liczba porządkowa.
  - `dataPoczatku`, `dataKonca` - okres obowiązywania danych.
  - `sumaUbezpieczenia`, `odnowienia` - informacje finansowe.
  - `ulica`, `kodPocztowy`, `miasto`, `wojewodztwo`, `kraj` - dane lokalizacyjne.
  - `reasekuracjaO`, `reasekuracjaF` - dane o reasekuracji.
  - `szerokosc`, `dlugosc` - współrzędne geograficzne.
  - `flaga1`, `flaga2` - dodatkowe informacje.
  - `nrwoj` - numer województwa.
- `FileInfo`: Przechowuje informacje o plikach CSV:
  - `path` - ścieżka do pliku.
  - `startIndex` - indeks początkowy w danych.
  - `lineCount` - liczba linii w pliku.

1.3.2 Funkcje pomocnicze

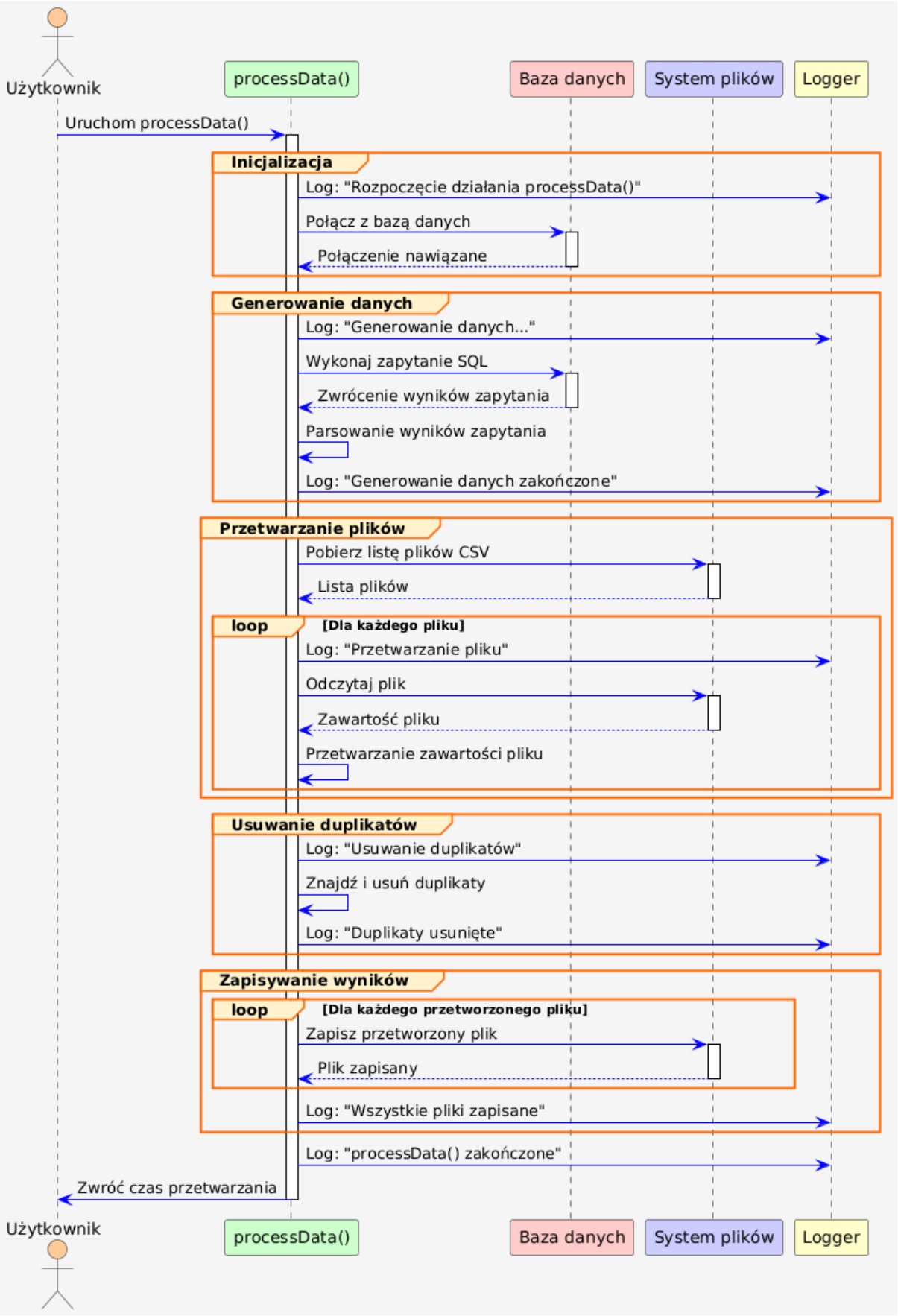
- `getCsvFiles`: Wyszukuje wszystkie pliki CSV w podanej ścieżce.
- `removeLastChar`: Usuwa ostatni znak z ciągu znaków.
- `findDuplicateCoordinates`: Wyszukuje duplikaty współrzędnych w danych.
- `removeMatchingCoordinates`: Usuwa rekordy zduplikowane na podstawie współrzędnych geograficznych.
- `getWojewodztwoMapa`: Mapuje kod pocztowy na województwo.

1.3.3 Funkcja główna: `processData`

Funkcja `processData` realizuje główne zadania programu:

1. Łączy się z bazą danych PostgreSQL i wykonuje zapytanie SQL w celu wygenerowania nowych danych.
2. Wczytuje dane z plików CSV i zapisuje je do wspólnego kontenera.
3. Usuwa rekordy zduplikowane na podstawie współrzędnych geograficznych.
4. Zapisuje przetworzone dane do nowych plików CSV.

Na rycina 1.2 przedstawiono diagram sekwencji dla funkcji `processData()`, która realizuje proces generowania, przetwarzania i zapisywania danych. Diagram ilustruje interakcje między głównymi komponentami, takimi jak użytkownik, baza danych, system plików oraz logger.



Rysunek 1.2: Diagram sekwencji dla funkcji `processData()`.

1.3.4 Wielowątkowość

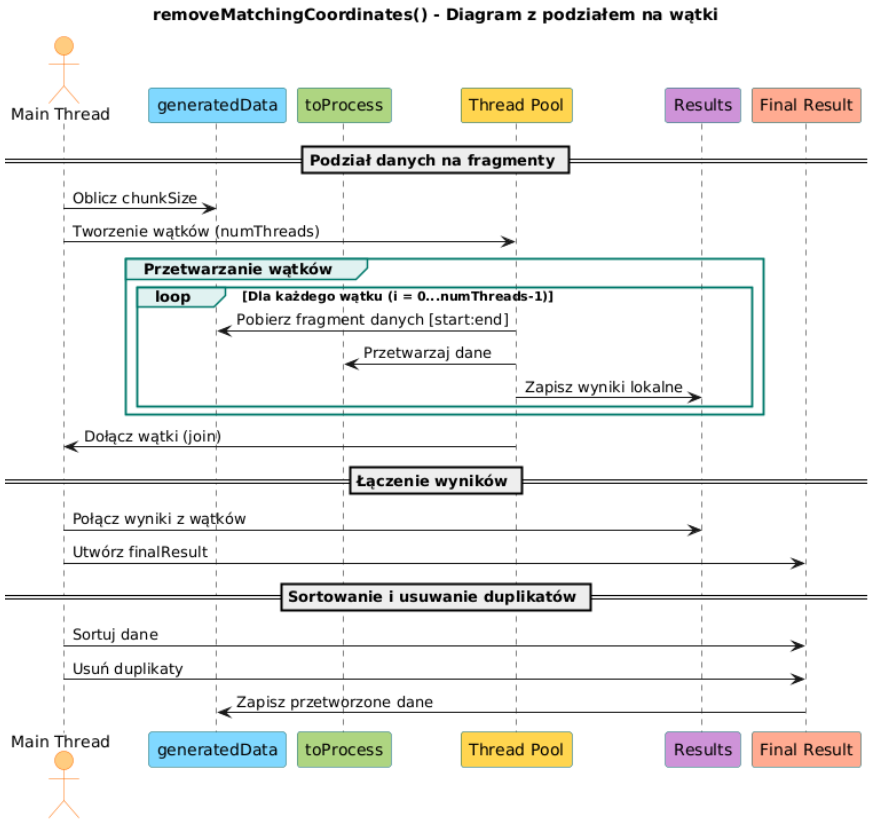
Operacje przy pomocy funkcji `removeMatchingCoordinates` są wykonywane w wielu wątkach, co pozwala na równoczesne przetwarzanie różnych fragmentów danych. Liczba wątków jest określona przez zmienną `numThreads`.

Diagram, zobrazowany na Ryc. 1.3, ilustruje główne kroki funkcji, w tym podział danych na fragmenty, przetwarzanie w wątkach, łączenie wyników oraz sortowanie i usuwanie duplikatów.

Rycina 1.3 przedstawia następujące kroki:

1. **Podział danych na fragmenty**: Główny wątek oblicza rozmiar fragmentów danych (`chunkSize`) i inicjuje wątki w puli (`Thread Pool`).
2. **Przetwarzanie wątków**: Każdy wątek pobiera odpowiedni fragment danych z `generatedData` oraz `toProcess`, przetwarza je i zapisuje wyniki lokalne w `Results`.
3. **Łączenie wyników**: Po zakończeniu pracy wątków, główny wątek łączy wszystkie wyniki lokalne w `Final Result`.
4. **Sortowanie i usuwanie duplikatów**: Główny wątek sortuje dane w `Final Result` i usuwa duplikaty za pomocą funkcji `std::unique` i `std::sort`.
5. **Zapisanie wyników**: Przetworzone dane są zapisywane z powrotem do `generatedData`.

Każdy etap procesu został zilustrowany na diagramie za pomocą odpowiednich komponentów, takich jak dane wejściowe (`generatedData` i `toProcess`), pula wątków (`Thread Pool`) oraz wyniki lokalne (`Results`). Diagram przedstawia również równoległe przetwarzanie danych przez wątki.



Rysunek 1.3: Diagram działania funkcji `removeMatchingCoordinates`.

1.4 Funkcjonalności

1.4.1 Wczytywanie danych

Program rekurencyjnie przeszukuje katalog wskazany w zmiennej `path`, aby znaleźć pliki CSV. Każdy plik jest wczytywany, a jego zawartość jest parsowana i przechowywana w wektorze obiektów `Address`. Dane są przetwarzane w sposób następujący:

- Wczytywane są wszystkie linie z pliku CSV.
- Nagłówki plików są przechowywane, aby umożliwić późniejszy zapis wyników.
- Dane są przechowywane w kontenerze `toProcess`.

1.4.2 Generowanie danych

Program łączy się z bazą PostgreSQL i wykonuje zapytanie SQL, które generuje dane geograficzne na podstawie tabeli `placex`. Generowanie danych obejmuje:

- Wybór unikalnych rekordów z tabeli na podstawie współrzędnych geograficznych (`lat`, `lon`).
- Mapowanie kodów pocztowych na województwa za pomocą funkcji `getWojewodztwoMapa`.
- Przechowywanie wyników w kontenerze `generatedData`.

Przykład zapytania SQL używanego w programie:

```
SELECT DISTINCT
    postcode, address,
    ST_Y(ST_Centroid(centroid)) AS lat,
    ST_X(ST_Centroid(centroid)) AS lon
FROM placex
WHERE address -> 'housenumber' IS NOT NULL
ORDER BY RANDOM()
LIMIT 30000000;
```

1.4.3 Usuwanie duplikatów

Program identyfikuje i usuwa rekordy zduplikowane na podstawie współrzędnych geograficznych (`szerokosc`, `dlugosc`). Proces obejmuje:

- Analizę danych w kontenerze `toProcess`.
- Usuwanie rekordów, które mają takie same współrzędne geograficzne.
- Wielowątkowe przetwarzanie danych za pomocą funkcji `removeMatchingCoordinates`.

1.4.4 Wielowątkowe przetwarzanie

Program wykorzystuje wielowątkowość w celu przyspieszenia przetwarzania dużych zbiorów danych. Liczba wątków jest konfigurowalna i ustawiona w zmiennej:

```
size_t numThreads = 32;
```

Każdy wątek przetwarza fragment danych, co pozwala na równoległe usuwanie duplikatów i filtrowanie wyników.

1.4.5 Zapis do plików CSV

Przetworzone dane są zapisywane do nowych plików CSV. Każdy plik wynikowy otrzymuje dopisek `_final`, aby odróżnić go od oryginału. Proces zapisu obejmuje:

- Otwieranie pliku wynikowego.
- Zapis przetworzonych danych z zachowaniem oryginalnego nagłówka.
- Obsługę błędów w przypadku problemów z zapisem.

1.4.6 Kolorowe logowanie

Program używa kolorów w konsoli, aby lepiej wyróżniać komunikaty informacyjne, ostrzeżenia i błędy. Przykład użycia kolorów:

```
Color::Modifier red(Color::FG_RED);
Color::Modifier green(Color::FG_GREEN);
Color::Modifier def(Color::FG_DEFAULT);

std::cout << red << "[INFO] " << def << "Rozpoczęto przetwarzanie danych." << std::endl;
```

1.5 Konfiguracja

1.5.1 Ścieżka do plików CSV

Ścieżka do katalogu z plikami CSV jest określona w zmiennej:

```
fs::path path = "/mnt/c/Users/szczkr/2_po_geokodowaniu";
```

1.5.2 Parametry bazy danych PostgreSQL

Połączenie z bazą danych jest konfigurowane w funkcji `processData`:

```
pqxx::connection c(
    "dbname=nominatim user=nominatim password=nominatim host=localhost port=5432"
);
```

1.5.3 Parametry wielowątkowości

Liczba wątków, które program wykorzystuje do przetwarzania danych, jest ustawiona w zmiennej:

```
size_t numThreads = 32;
```

1.6 Struktury danych

1.6.1 Struktura `Address`

Struktura `Address` reprezentuje pojedynczy rekord danych adresowych. Definicja:

```
struct Address {
    std::string lp;
    std::string dataPoczatku;
    std::string dataKonca;
    std::string sumaUbezpieczenia;
    std::string odnowienia;
    std::string ulica;
    std::string kodPocztowy;
    std::string miasto;
    std::string wojewodztwo;
    std::string kraj;
    std::string reasekuracjaO;
    std::string reasekuracjaF;
    std::string szerokosc;
    std::string dlugosc;
    std::string flaga1;
    std::string flaga2;
    std::string nrwoj;
};
```

1.7 Przykładowy przebieg działania programu

1.7.1 Generowanie danych

Po uruchomieniu programu użytkownik wybiera opcję generowania danych. Program łączy się z bazą PostgreSQL, wykonuje zapytanie SQL i generuje dane w pamięci. Przykład komunikatu w konsoli:

```
[INFO] Rozpoczęto generowanie danych...
[INFO] Generowanie danych zakończone.
```

1.7.2 Wczytywanie plików CSV

Program przeszukuje katalog w celu znalezienia plików CSV, a następnie wczytuje ich zawartość. Przykład komunikatu:

```
[INFO] Przetwarzanie pliku: data1.csv
[INFO] Przetwarzanie pliku: data2.csv
```

1.7.3 Usuwanie duplikatów i zapis wyników

Po usunięciu duplikatów program zapisuje wyniki do nowych plików CSV. Przykład komunikatu:

```
[INFO] Zapisywanie do pliku: data1_final.csv
[INFO] Zapisywanie do pliku: data2_final.csv
```

1.8 Instrukcja obsługi

1.8.1 Kompilacja programu

Program należy skompilować za pomocą kompilatora obsługującego standard C++17 lub nowszy:

```
g++ -std=c++17 -lpqxx -lpq main.cpp -o program
```

1.8.2 Uruchomienie programu

Uruchom program w terminalu za pomocą poniższego polecenia:

```
./program
```

1.8.3 Interfejs użytkownika

Po uruchomieniu programu użytkownik ma do wyboru następujące opcje:

1. Rozpocznij przetwarzanie danych - inicjuje proces wczytywania, generowania i przetwarzania danych.
2. Pokaż postęp prac - wyświetla aktualny stan przetwarzania, w tym liczbę przetworzonych plików i rekordów.
3. Pokaż segmentację kontenera - wyświetla szczegóły dotyczące podziału danych na fragmenty.
4. Wyjdź - kończy działanie programu.