Politechnika Śląska w Gliwicach

Wydział Automatyki, Elektroniki i Informatyki

Programowanie Komputerów 4

Baza Danych – Przychodnia

autor	Jacek Ganszczyk
prowadzący	dr inż.Roman Starosolski
rok akademicki	2017/2018
kierunek	informatyka
rodzaj studiów	dzienne
semestr	4
grupa	Warunek
sekcja	4
termin oddania sprawozdania	2018-06-28

1. Treść zadania

Napisać program będący implementacją bazy danych obsługiwanej przez przychodnię. Baza ta przechowuje informacje o pacjencie, lekarzu oraz o innych przychodniach. Program powinien zapewniać dodawanie oraz usuwanie z bazy przechowywanych danych. Oprócz tego powinna istnieć możliwość wyświetlenia wszystkich danych pacjentów, lekarzy, czy też przychodni. Program powinien umieć przetwarzać wizyty oraz zapisy pacjentów. W przypadku, gdy lekarz nie ma miejsca do przyjęcia pacjenta, skierować go do innego lekarza lub przychodni.

2. Analiza zadania

Zagadnienie przedstawia problem stworzenia funkcjonalnej bazy danych, przechowującej informacje o pacjencie, lekarzu i innych przychodniach oraz korelującej te dane w odpowiedni sposób.

2.1. Algorytmy, struktury danych, analiza obiektowa

W programie zostało zaimplementowanych 7 klas. Została stworzona klasa ***Adres***, która jest wykorzystywana w większości klas. Oprócz tego klasa bazowa ***osoba*** oraz dziedziczące po niej klasy ***lekarz*** oraz ***pacjent***. Korelację pomiędzy lekarzem, a zapisanymi pacjentami obsługuje klasa ***lekarzLista***. Ponadto stworzono klasę ***przychodnia*** zawierająca informację o dostępnych przychodniach. Wszystkie te klasy są wykorzystywane w klasie ***interfejs***.

3. Specyfikacja zewnętrzna

3.1. Obsługa programu

Program uruchamiany z wiersza poleceń z wykorzystaniem następujących przełączników (ich kolejność jest dowolna):

-o plik z danymi pacjentów
-l plik z danymi lekarzy
-p plik z danymi innych przychodni

Ponadto program można uruchomić wykorzystując przełącznik „*-h*” w celu uzyskania pomocy.

Po uruchomieniu programu użytkownik za pomocą wyświetlanego menu może wybierać odpowiednie opcje, które zostaną wykonane. W przypadku zakończenia pracy programu, baza danych jest zapisywana.

3.2. Format danych wejściowych

Pliki wejściowe powinny być plikami tekstowymi.

3.3. Komunikaty

Program będzie wyświetlał na konsoli komunikaty w przypadkach powodzenia lub niepowodzenia wykonania jakiejś operacji.

Pierwszym komunikatem, jaki może wystąpić jest komunikat o zbyt małej liczbie parametrów przekazanych do programu. Następnie program może poinformować użytkownika o wykorzystaniu błędnego przełącznika.

Program wyświetla odpowiednie zapytania podczas dodawania nowych danych do bazy. Na przykład zapytanie o imię czy nazwisko.

Przy zapisywaniu wizyty u lekarza program wyświetla zapytania dotyczące wyboru lekarza oraz informuje o powodzeniu lub niepowodzeniu zapisania tej wizyty.

Podczas przetwarzania wizyty u lekarza wyświetlane są odpowiednie zapytania oraz komunikaty świadczące o powodzeniu lub niepowodzeniu przetwarzania danej wizyty.

4. Specyfikacja wewnętrzna

Program zawiera zmienne oraz funkcje statyczne. W programie zaimplementowano siedem klas.

4.1. Klasa Adres

4.1.1. Rola klasy

Klasa ta przechowuje informacje o adresie.

4.1.2. Pola klasy

W klasie występują cztery pola typu string służące do przechowywania nazwy ulicy, numeru domu, miasta oraz kodu pocztowego.

4.1.3. Najważniejsze metody klasy

`void WczytajAdres(const std::string& adres);`
Przetwarza podany w parametrze adres i wpisuje go do swoich pól.

`void ZapiszDoPliku(std::ofstream& plik) const;`
Zapisuje adres do pliku.

4.2. Klasa osoba

4.2.1. Rola klasy

Klasa bazowa, przechowująca podstawowe informacje o osobie.

4.2.2. Pola klasy

Klasa posiada zmienne chronione. Imię i nazwisko przechowuje w zmiennych typu *string*. Adres osoby jest przechowywany w zmiennej typu *Adres*. Oprócz tego klasa przechowuje id oraz numer telefonu w zmiennych typu *całkowitoliczbowego*.

4.2.3. Najważniejsze metody klasy

`void WczytajInformacje(const std::string& informacje);`
Przetwarza informacje podane w parametrze i wpisuje do pól.

`void ZapiszOsobeDoPliku(std::ofstream& plik) const;`
Zapisuje osobę do pliku.

4.3. Klasa pacjent

4.3.1. Rola klasy

Klasa służy do przechowywania informacji o pacjencie. Dziedziczy po klasie osoba.

4.3.2. Pola klasy

Klasa przechowuje informacje o numerze pesel w zmiennej całkowitoliczbowej oraz numer ubezpieczenia w zmiennej typu *string*. Klasa korzysta ze zmiennej statycznej typu *int* do odpowiedniego przypisywania id podczas dodawania pacjentów.

4.3.3. Najważniejsze metody klasy

`void WczytajDanePacjenta(const std::string& _dane_pacjenta);`
Przetwarza informacje podane w parametrze i wpisuje do pól.

`void ZapiszPacjentaDoPliku(std::ofstream& plik) const;`
Zapisuje pacjenta do pliku.

4.4. Klasa lekarz

4.4.1. Rola klasy

Klasa służy do przechowywania informacji o lekarzu. Dziedziczy po klasie osoba.

4.4.2. Pola klasy

Klasa przechowuje informacje o specjalizacji danego lekarza w zmiennej typu *string*. Klasa korzysta ze zmiennej statycznej typu *int* do odpowiedniego przypisywania id podczas dodawania lekarzy.

4.4.3. Najważniejsze metody klasy

`void WczytajDaneLekarzy(const std::string& _dane_lekarza);`
Przetwarza informacje podane w parametrze i wpisuje do pól.

`void ZapiszLekarzaDoPliku(std::ofstream& plik) const;`
Zapisuje lekarza do pliku.

4.5. Klasa lekarzLista

4.5.1. Rola klasy

Klasa służy do przechowywania lekarza oraz listy pacjentów, którzy są do niego umówieni.

4.5.2. Pola klasy

Klasa przechowuje lekarza w zmiennej typu *lekarz* oraz kolejkę pacjentów w strukturze *queue<int>.* Klasa przechowuje również maksymalną liczbę wizyt u lekarza.

4.5.3. Najważniejsze metody klasy

`void WczytajDoLekarzaZLista(const std::string& informacja);`
Przetwarza informacje podane w parametrze i wczytuje do pól.

`void ZapiszLekarzListaDoPliku(std::ofstream& plik);`
Zapisuje lekarza z listą pacjentów do pliku.

`std::string WpiszPacjenta(const std::shared_ptr<pacjent>& wybrany_pacjent);`
Wpisuje pacjenta przekazanego przez parametr do kolejki.

4.6. Klasa przychodnia

4.6.1. Rola klasy

Klasa służy do przechowywania informacji dotyczących przychodni.

4.6.2. Pola klasy

Klasa przechowuje nazwę przychodni w zmiennej typu *string*. Adres przychodni jest przechowywany w zmiennej typu *Adres*. Specjalizacje obsługiwane przez daną przychodnie są przechowywane w liście jednokierunkowej *forward_list<string>.*

4.6.3. Najważniejsze metody klasy

`void WczytajInformacjePrzychodni(const std::string& informacje);`
Przetwarza informacje podane w parametrze i wczytuje do pól.

`void DodajSpecjalizacje();`
Dodaje specjalizacje obsługiwaną przez przychodnie do listy.

`void ZapiszPrzychodnieDoPliku(std::ofstream& plik) const;`
Zapisuje przychodnie do pliku.

`bool PrzychodniaDanejSpecjalnosci(const std::string specjalnosc);`
Sprawdza czy przychodnia obsługuję specjalność podana w parametrze.

4.7. Klasa interfejs

4.7.1. Rola klasy

Klasa służy do interakcji z użytkownikiem. Jak sama nazwa mówi, jest to interfejs programu.

4.7.2. Pola klasy

Klasa przechowuje nazwy plików z pacjentami, lekarzami oraz przychodniami w zmiennych typu *string*. Pacjenci są przechowywani w strukturze drzewiastej *multimap<string, shared_ptr<pacjent>>*, kluczem tego drzewa jest nazwisko pacjenta. Podobnie przechowywani są lekarze wraz z listą wizyt pacjentów (*mulimap<string,shared_ptr<lekarzLista>>*, klucz nazwisko lekarza). Przychodnie są przechowywane w jednokierunkowej liście *forward_list<shared_ptr<przychodnie>>*. Niestety struktura ta nie posiada kontroli jej rozmiaru. Dlatego klasa przechowuje również rozmiar tej listy. Oprócz wymienionych pól, klasa przechowuje

również zmienna typu *bool* do kontroli czy program powinien się zakończyć.

4.7.3. Najważniejsze metody klasy

`void Program(int argc, char** argv);`
Funkcja publiczna. Służy do uruchomienia programu.

`bool PobierzParametry(int argc, char ** argv);`
Funkcja pobierająca parametry i wpisująca nazwy plików do odpowiednich pól.

`void DodawanieDoBazy();`
Funkcja odpowiadająca za dodawanie danych do bazy.

`void ZarzadzaniePacjentami();`
Funkcja odpowiadająca za zarządzanie pacjentami.

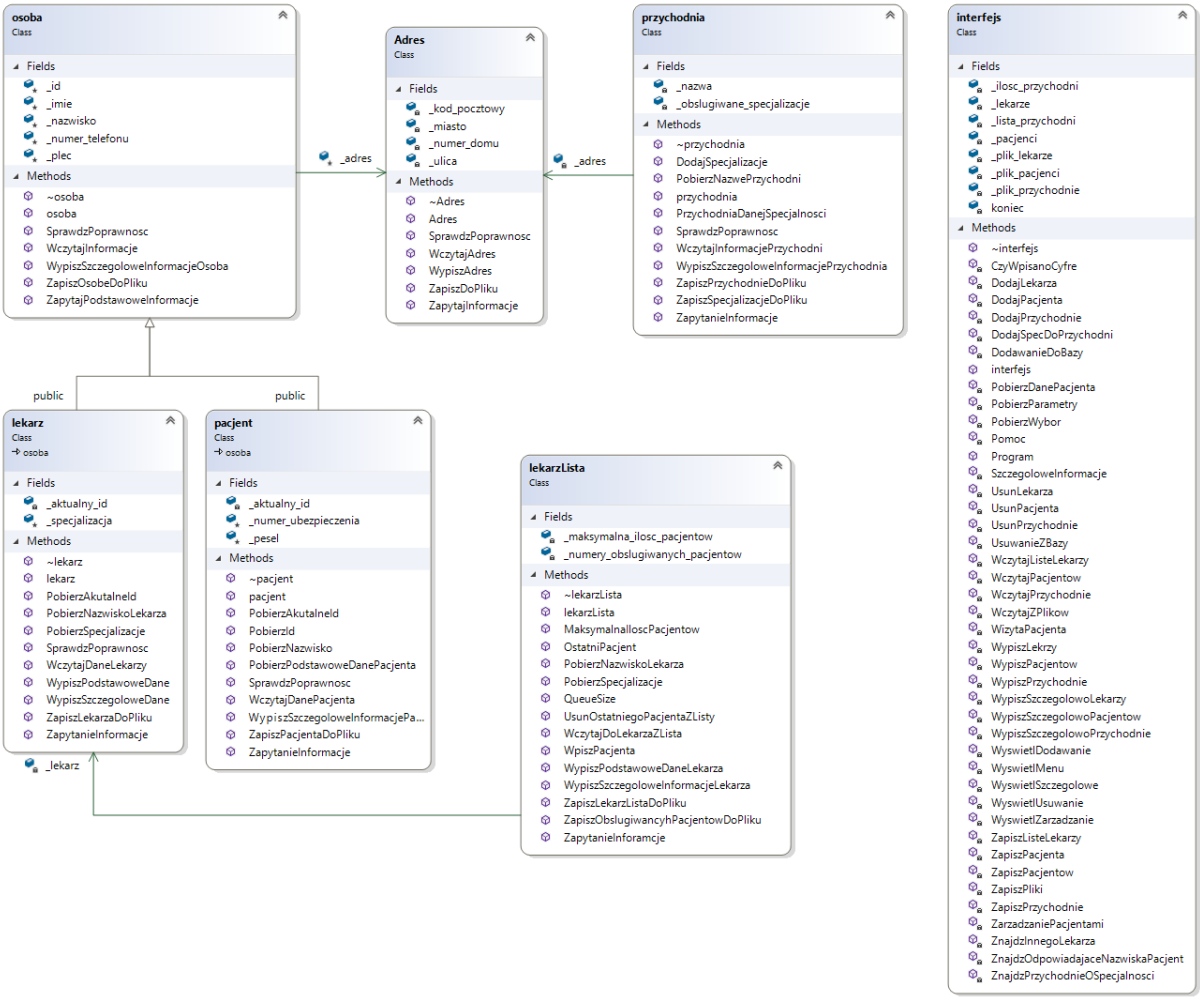
`void UsuwanieZBazy();`
Funkcja odpowiadająca za usuwanie danych z bazy.

`void SzczegoloweInformacje();`
Funkcja odpowiadająca za wypisywanie szczegółowych informacji wybranych danych.

`void WczytajZPlikow();`
Funkcja wczytująca informacje z plików do programu.

`void ZapiszPliki();`
Funkcja zapisująca informacje z programu do plików.

4.8. Diagram klas:



4.9. Przepływ sterowania:

5. Testowanie i uruchamianie

5.1. Testy:

Aplikacja była testowana róznymi danymi wejściowymi zarówno w przypadku danych pobieranych z plików jak i danych wprowadzanych w konsoli. Wraz z kolejnymi pojawiającymi się błędami były wprowadzane dodatkowe formy walidacji danych. Głównie wyrażenia regularne sprawdzające ich poprawanność. Również zostało zastosowane sprawdzanie wyjątków w przypadku wczytywania plików, w przypadku złego formatu czy formatowania danych w plikach wyrzucany jest wyjątek i przerwywana praca programu. To samo tyczy się wprowadzania danych nie odpowiedniego typu co kończyło by się błędem w rzutowaniu.