Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

W Gliwicach

Programowanie Komputerów

2

Program do rejestracji pacjentów w przychodni

Autor	Agnieszka Góral
Prowadzący	dr inż. Wojciech Łabaj
Rok akademicki	2020/2021
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	2
Termin laboratoriów	Poniedziałek 13:45-15.15 Piątek 13.30-15.00
Sekcja	2
Grupa	2

SYSTEM REJESTRACJI W PRZYCHODNI

1. OPIS DZIAŁANIA PROGRAMU (skrócona analiza programu)

Program ma za zadanie umożliwić pacjentom rejestrację do lekarzy na wybrane godziny, uwzględniając czas pracy konkretnego lekarza. Najpierw pacjent musi się zalogować za pomocą numeru PESEL oraz hasła. Jeśli jeszcze nie ma konta to program poprosi go o podanie danych (imię, nazwisko, data urodzenia, telefon kontaktowy, pesel) i założy konto.

Z poziomu menu (wyświetlanego zaraz po zalogowaniu) pacjent ma 4 opcje:

1. Zapisanie się na wizytę
-> wybór lekarza (przez wyszukiwanie po specjalizacji)
->możliwość wybrania daty
->możliwość sprawdzenia dnia i już umówionych wizyt oraz wybranie godziny dla siebie

2. Przejrzenie grafików lekarzy
-> wybór lekarza (przez wyszukiwanie po specjalizacji)

3. Sprawdzenie konta pacjenta (dane)

3. Wyświetlenie dat zaplanowanych wizyt i tych, które już się odbyły, możliwość odwołania wizyty

4. Wylogowanie się

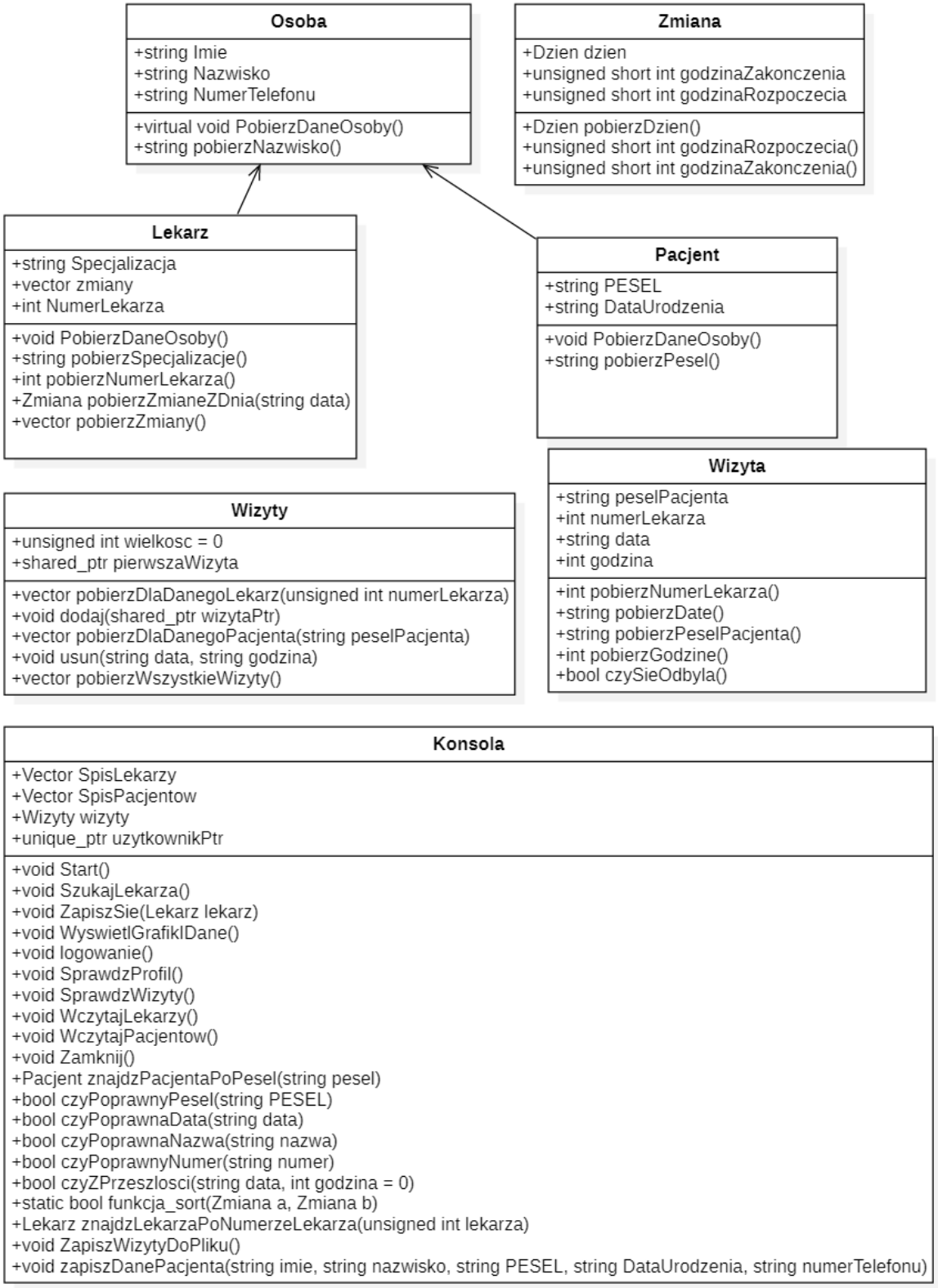
5. Zamknięcie programu

Pacjent może wybrać lekarza, u którego jest zainteresowany wizytą za pomocą wyszukiwarki specjalistów.

Po wybraniu lekarza oraz daty wyświetlany jest jego grafik na zadany dzień oraz dostępne godziny. Pacjent wybiera odpowiednią godzinę oraz zatwierdza wizytę.

Godzina, na którą zapisał się pacjent zostaje oznaczona jako zajęta i inni pacjenci już nie będą mogli się na nią zapisać. Po powrocie do menu, pacjent będzie mógł sprawdzić, czy data zaplanowanej wizyty znajduje się na jego koncie.

2. OPIS KLAS I STRUKTUR DANYCH



OSOBA (class Osoba{};)
Klasa opisująca osobę, podstawowe informacje personalne każdego bywalca przychodni. Zawiera zmienne prywatne: typu string - imie, nazwisko oraz numerTelefonu.

Publiczne metody klasy:

- **virtual void *pobierzDaneOsoby ()*** – metoda zwracająca w stringu „sklejone" dane osoby wraz z opisami.

- ***String pobierzNazwisko()*** – *metoda zwracająca prywatną zmienną klasy - nazwisko*

PACJENT (class Pacjent{};)
Pacjent dziedziczy z klasy Osoba zmienne takie jak imie, nazwisko oraz numerTelefonu. Klasa ta posiada zmienne prywatne DataUrodzenia (typ string) oraz pesel (typ string) - obie zmienne przechowują informacje o pacjencie wg nazw zmiennych - datę urodzenia oraz numer PESEL oraz zmienną Wizyty, typu wizyty (przechowuje ona wizyty, na które umówił się pacjent).

Publiczne metody klasy:

- **string *PobierzDaneOsoby()*** - pozwalająca na pobranie danych z prywatnych zmiennych w klasie (tj. imienia, nazwiska, numeru telefonu, PESELu oraz daty urodzenia pacjenta), zwraca string z tymi danymi

- **string pobierzPesel()** - – *metoda zwracająca prywatną zmienną klasy - pesel*

LEKARZ (class Lekarz{};)

Klasa ta dziedziczy z klasy OSOBA zmienne takie jak imie, nazwisko oraz numer_telefonu. Dodatkowo klasa ta posiada numerLekarza (typ string, zmienna prywatna klasy) - jest to numer nadany lekarzowi (indywidualny numer licencji dla każdego z lekarzy) oraz pole specjalizacja (typ string, zmienna prywatna klasy) - zmienna przechowuje informacje o specjalizacji danego lekarza.

Publiczne metody klasy:

- **string *pobierzDaneOsoby()*** - pozwalająca na pobranie danych z prywatnych zmiennych w klasie (tj. imienia, nazwiska, numeru telefonu, numeru lekarza, specjalizacji), zwraca string z tymi danymi

- **string pobierzSpecjalizacje()** – metoda odpowiedzialna za zwracanie stringa, który przechowuje specjalizacje lekarza.

- **Zmiana pobierzZmianeZDnia (std::string data)** - metoda zraca zmianę lekarza z danego dnia (zmiana tj godziny pracy w danym dniu)

- **int pobierzNumerLekarza()** – metoda zwracająca zmienną prywatną klasy - numerLekarza

- **vector<Zmiana> pobierzZmiany()** - metoda odpowiedzialna za zwracanie zmian (godzin pracy z dniami) danego lekarza

WIZYTA (class Wizyta{};)

Klasa ta ma obrazować pojedynczą wizytę pacjenta u lekarza. Prywatne zmienne klasy: peselPacjenta (typ string), numerLekarza (typ int), data (typ string), godzina (typ int).

Publiczne medody klasy:

- ***string pobierzDate()*** - metoda zwracająca zmienną prywatną klasy – data

- **int pobierzGodzine()** – metoda zwracająca zmienną prywatną klasy – godzina

- **string pobierzPeselPacjenta()** - metoda zwracająca zmienną prywatną klasy – peselPacjenta

- **int pobierzNumerLekarza()** - metoda zwracająca zmienną prywatną klasy – numerLekarza

- **bool czySieOdbyla()** – metoda sprawdzająca, czy dana wizyta już się odbyła

KONSOLA (class Konsola{};)

Klasa ta jest odpowiedzialna za komunikację z użytkownikiem oraz wyświetlanie danych w konsoli i za wczytanie danych.

W konsoli znajdują się publicznie zmienne takie jak: lekarze (vector <Lekarz>), spisPacjentow(vector <Pacjent>) oraz unique_ptr<Pacjent> uzytkownikPtr – inteligentny wskaźnik na klasę Pacjent. Zmienne spisPacjentow i lekarze to wektory, do których są przekazywane dane pobrane z plików wejściowych. Zmienne te mają symbolizować listę pacjentów oraz lekarzy.

Publiczne metody klasy:

- ***void Start()*** - metoda odpowiedzialna za wyświetlanie ekranu startowego w konsoli oraz udostępnienie użytkownikowi poszczególnych opcji.
  W pierwszej kolejności metoda wywołuje metodę logowanie(), następnie, jeśli użytkownik się zalogował wyświetla menu i opcje do wyboru. Wybór opcji został zaimplementowany za pomocą switch'a.

Prywatne metody klasy:

- ***void SzukajLekarza()*** - pozwala na znalezienie lekarza po specjalizacji oraz umówienie wizyty do wybranego lekarza.
  Metoda posiada zmienne lokalną specjalizacja, typu string, zmienną typu bool czyIstniejeDanaSpecjalizacja, z domyślną wartością false oraz zmienna typu int o domyślnej wartyości 0 numerLekarza.

  Najpierw wywołana metoda prosi o podanie specjalizacji lekarza, jakiego poszukujemy i umieszcza ją w zmiennej specjalizacja. Następnie za pomocą pętli for sprawdza, czy lekarz o podanej specjalizacji istnieje w wektorze lekarze.

  Jeśli istnieje to zmienia zmienną czyIstniejeDanaSpecjalizacja na true i wyświetla wszystkich lekarzy posiadających daną specjalizacje i ich dane. Następnie wyświetlone zostaje menu wyboru – pacjent może wybrać czy chce się zapisać do lekarza(poprzez podanie jego numeru) lub, czy chce wrócić go głównego menu (podając 0). Odpowiedź użytkownika jest przechowywana w zmiennej numerLekarza. Jeśli numer lekarza jest poprawny to wywołana zostaje metoda zapiszSie(), jeśli numer został podany błędnie pojawi się komunikat o niepoprawnie wprowadzonych danych.

  Jeśli dana specjalizacja nie istnieje (jeśli zmienna czyIstniejeDanaSpecjalizacja ma wartość false) to wyświetla komunikat informujący o tym, że w tej przychodni nie przyjmuje lekarz o zadanej specjalizacji i powraca do menu wyboru.

- ***void ZapiszSie()*** - metoda umożliwiająca zapisanie się do konkretnego lekarza..
  W pierwszej kolejności użytkownik proszony jest o podanie daty wizyty jaka go interesuje. Jego odpowiedź jest przechowywana w zmiennej data.
  Następnie szuka tylko grafiku na dany dzień dla danego lekarza. Później wyświetla grafik na dany dzień i informuje o dostępności godzin. Pacjent może podać godzinę jaka go interesuje. Jeśli godzina jest poprawna to utworzony zostaje nowy obiekt klasy wizyta i zostaje ona przypiana do konta pacjenta oraz grafiku lekarza.

- **bool czyPoprawnaNazwa(std::string nazwa)-** metoda odpowiedzialna za sprawdzanie czy w podanym stringu na pewno znajdują się same litery.

- **bool czyPoprawnaNazwa(std::string nazwa)** - metoda odpowiedzialna za sprawdzanie czy w podanym stringu na pewno znajdują się same litery.

- **bool czyPoprawnaData(std::string data)** - metoda odpowiedzialna za sprawdzanie czy data została podana w poprawny sposób.

- **bool czyPoprawnyNumer(std::string numer)** - metoda odpowiedzialna za sprawdzanie czy w podanym stringu (numerze telefonu) na pewno znajdują się same cyfry oraz czy ciąg składa się z 9 cyfr.

- **bool czyPoprawnyPesel(std::string PESEL) -** metoda odpowiedzialna za sprawdzanie czy podany sting (Pesel) został wprowadzony poprawnie tj, czy spełnia założenia sumy kontrolnej (ustalonej odgórnie) oraz czy ma 11 cyfr.

- **void wczytajPacjentow() -** metoda odpowiedzialna za wczytywanie danych pacjentów z pliku.

- **void wczytajLekarzy() -** metoda odpowiedzialna za wczytywanie danych lekarzy z pliku.

- **void ZapiszWizytyDoPliku() -** metoda odpowiedzialna za wczytywanie wizyt z pliku tekstowego.

- **void SzukajLekarza() -** metoda odpowiedzialna za wyszukiwanie lekarza.

- **bool czyZPrzeszlosci(std::string data, int godzina) -** metoda odpowiedzialna za sprawdzanie czy data wraz z godziną nie jest z przeszłości (tj. czy na pewno wyprzedza czas pobrany z komputera).

- ***Lekarz znajdzLekarzaPoNumerzeLekarza(unsigned int numerLekarza)* -** metoda odpowiedzialna za wyszukiwanie lekarza po jego numerze.

- ***void Logowanie()*** - metoda odpowiedzialna za logowanie do systemu oraz zakładanie konta użytkownikom. Posiada zmienną lokalną typu bool NiePoprawnyPesel, o domyślnej wartości true. Najpierw prosi użytkownika o podanie numeru pesel, w celu sprawdzenia, czy istnieje już w bazie, czy należy utworzyć nowe konto. W pierwszej kolejności sprawdza poprawność wprowadzonego numeru (jeśli jest błędny to wyświetla komunikat i prosi o ponowne wprowadzenie) , następnie (jeśli numer jest poprawny) za pomocą pętli „for” sprawdza czy podany pesel znajduje się w wektorze pacjentów spisPacjentow. Jeśli tak to loguje użytkownika, jeśli jest to osoba niezarejestrowana to przechodzi to sekcji rejestracji i prosi o podanie danych osobowych (tworzy przy tym nowy obiekt klasy Pacjent) oraz przypisuje do zmiennej urzytkownikPtr wskaźnik na danego (zalogowanego) użytkownika.

- **bool funkcja_sort(Zmiana a, Zmiana b)** - metoda odpowiedzialna za sortowanie godzin pracy lekarzy.

- ***void SprawdzProfil()*** - metoda umożliwiająca sprawdzenie swojego profilu przez pacjenta.

- **Pacjent znajdzPacjentaPoPesel(string pesel) –** metoda, która przyjmuje zmienną typu string (przekazujemy do niej pesel osoby szukanej), następnie przeszukuje wektor – spisPacjentow i zwraca instancje o podanym numerze pesel.

- ***void SprawdzWizyty()*** - metoda służąca do wyświetlania wizyt zalogowanego użytkownika.

- **void WyswietlGrafikIDane()-** metoda odpowiedzialna za wyświetlanie grafiku oraz danych wybranego lekarza.

- ***void WyszukiwanieLekrza()* -** Metoda odpowiedzialna za wyszukiwanie lekarzy (specjalizacji).

- ***void wczytajPacjentow()*** -wczytuje pacjentow z pliku tekstowego (pacjenci.txt), a następnie umieszcza je w wektorze o nazwie spisPacjentow. Po zakończeniu zczytywania danych zamyka plik.

- ***void zapiszDanePacjenta(std::string imie, std::string nazwisko, std::string PESEL, std::string DataUrodzenia, std::string numerTelefonu)*** - metoda odpowiedzialna za zapisywanie danych pacjenta do pliku. Po zakończeniu zapisywania zamyka plik.

WIZYTY (class Wizyty{};)

Zbiór pojedynczych wizyt dla pacjenta i dla lekarza. Zawiera prywatne zmienne wielkosc (unsigned int) oraz shared_ptr<Wizyta> pierwszaWizyta – inteligentny wskaźnik na pierwszą wizytę.

Publiczne metody klasy:

- ***void Dodaj (std::shared_ptr<Wizyta> wizytaPtr)*** - pozwalająca na dodanie wizyty do grafiku

- ***void Usun (std::string data, std::string godzina)*** - pozwalająca na usunięcie wizyty z grafiku

- **vector<Wizyta> pobierzDlaDanegoLekarz(unsigned int numerLekarza) –** metoda odpowiedzialna za zwrócenie wizyt danego lekarza.

- **vector<Wizyta> pobierzDlaDanegoPacjenta(std::string peselPacjenta)** - metoda odpowiedzialna za zwrócenie wizyt danego pacjenta.

- **vector<Wizyta> pobierzWszystkieWizyty()** - metoda która zwraca wszystkie wizyty jakie istnieją

ZMIANA (class Zmiana{};)

Obrazuje pojedynczy dzień pracy lekarza tj dzień tygodnia, wraz z godzinami pracy. Klasa zawiera zmienne prywatne dzień (typ Dzien – zdefiniowany w pliku nagłówkowym Dzien.h) oraz godzinaRozpoczecia i godzinaZakonczenia (typu unsigned short int).

Publiczne metody klasy:

- ***Dzien pobierzDzien()*** - metoda odpowiedzialna za zwracanie zmiennej typu Dzien, zwraca wartość ze zmiennej prywatnej klasy o nazwie dzień.

- ***int pobierzGodzineRozpoczecia()***- metoda odpowiedzialna za zwracanie zmiennej typu int, zwraca wartość ze zmiennej prywatnej klasy o nazwie godzinaRozpoczecia.

- **int pobierzGodzineZakonczenia()–** metoda odpowiedzialna za zwracanie zmiennej typu int, zwraca wartość ze zmiennej prywatnej klasy o nazwie godzinaZakonczenia.

3. SPECYFIKACJA ZEWNĘTRZNA

Program prowadzi użytkownika przez poszczególne opcje, użytkownik dokonuje wyborów za pomocą wpisywania poleceń w konsolę. Głównym punktem nawigacyjnym jest menu główne.

4. SPECYFIKACJA WEWNĘTRZNA

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikacji z użytkownikiem) od logiki aplikacji.

4.1 OGÓLNA STRUKTURA PROGRAMU

Program tworzy obiekt klasy Konsola(), a następnie wywołuje metodę Start(). Pierwsze pojawia się logowanie za pomocą numeru pesel. Użytkownik może się zalogować lub utworzyć konto, jeśli nie znajduje się w bazie pacjentów następnie za pomocą menu opartego na instrukcji switch może wybrać opcje, która go interesuje. Może np. zapisać się na wizytę lub ją odwołać. Program na chwilę po rozpoczęciu pracy pobiera dane z plików

tekstowych i zapisuje w odpowiednych kontenerach na dane (pobiera listę pacjentów, lekarzy oraz spis wizyt do lekarzy), po zakończeniu sczytywania plików zamyka je. Po zakończeniu pracy wypisuje wizyty do pliku (jeśli jakieś przybyły od ostatniego uruchomienia.

4.2 SZCZEGÓŁOWY OPIS TYPÓW I FUNKCJI

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5. TESTOWANIE

Program został przetestowany na różnych plikach wejściowych. Włącznie z plikami, w których nie wszystkie dane są prawidłowe (część linijek jest pusta, w pliku znajdują się losowe znaki lub brakuje jakiejś danej).

Program wczytując dane od użytkownika sprawdza ich poprawność, w razie podania błędnych danych wyświetla komunikat.

Program został przetestowany pod kątem wycieków pamięci. Pomimo dużych rozmiarów plików na jakich program był testowany, nie pojawił się komunikat o naruszeniu ochrony pamięci.

6. WNIOSKI

Program podczas tworzenia sprawił mi kilka trudności. Przede wszystkim dużym utrudnieniem okazało się pracowanie z biblioteką związaną z czasem – potrzeba była ona do kontrolowania aktualności daty i oceniania, czy podana data aby na pewno jest z przyszłości. Rzutowanie dat na typ int oraz wyliczanie konkretnych wartości dla tm. Kolejnym dość sporym problemem okazał się dostęp do zmiennych z klas. Jednak finalnie większość problemów została rozwiązana i zdobyłam nową wiedzę.

Literatura:

- „C++ Przewodnik Dla Początkujących” Alex Allain
- „Podstawy Programowania w Języku C++” Józef Zieliński