Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Zarządzanie aptekami

Autor:	Tomasz Szewczyk
Prowadzący:	mgr inż. Anna Glodek
Rok akademicki:	2020/2021
Kierunek:	Informatyka
Rodzaj studiów:	SSI
Semestr:	4
Termin laboratorium:	wtorek 13:15-14:45
Sekcja:	2

1. Treść zadania

Tematem projektu programistycznego jest stworzenie systemu do zarządzania aptekami w mieście. Projekt będzie posiadał 6 klas. Projekt będzie zawierać klasę MIASTO. Będzie ona zawierać jeden z kontenerów STL. Może to być wektor. Będzie przechowywał nazwy miast. Następną klasą będzie APTEKA Będzie możliwość zarządzania aptekami od strony menadżera apteki PRACOWNIK. Pracownik będzie miał możliwość sprawdzenia czy dany lek jest dostępny w bazie, do czego będzie służyć klasa LEKARSTWA. Lekarstwa będą posiadać dwie klasy dziedziczące SYROPY oraz TABLETKI. PRACOWNIK Będzie mógł stworzyć raport, w którym będzie zawierać się informacja dot. jego apteki. Będą dostępne lekarstwa na receptę i bez recepty co będzie można sprawdzić. W programie muszą znajdować się funkcje dodawania, usuwania, wyszukiwania, wypisywania aptek. Dodawania i usuwania pracowników. Przeciążone operatory. Dziedziczenie. Funkcje wirtualne min.1. Algorytmy i iteratory STL, obsługa wyjątków, funkcje lambda, REGEX, kontenery STL.

2. Analiza zadania

Zadanie przedstawia problem zarządzania aptekami w mieście. Analizując treść zadania od początku warto zastanowić się jak taki system w ogóle miałby działać(jak takie systemy funkcjonują w rzeczywistości). Program został wykonany testowo w programie głównym main. Z racji tego, że mamy do czynienia z aptekami na początku warto zrobić research dot. aptek. Jak ma wyglądać struktura przemieszania się po programie, jakich klas oraz struktur danych użyję. Z racji tego, że program działa od strony wewnętrznej apteki warto rozpatrzyć ten problem z pozycji menedżera takiej apteki. Program został także wykonany zgodnie z wytycznymi wyznaczonymi przez prowadzącego przedmiot Programowanie Komputerów IV.

2.1 Struktury danych

Program jako struktury danych używa zmiennych typu int, double, string, które zawierają się w poszczególnych klasach oraz tworzonych obiektów tj MIASTA, APTEKI, LEKARSTWA. Dodatkową strukturą przechowującą dane jest lista z biblioteki STL do przechowywania obiektów miast, pracowników, leków i aptek.

2.2 Algorytmy

Program wykorzystuje algorytmy należące do biblioteki STL a dokładniej kontenera jakim jest lista jednokierunkowa. Podstawową operacją jest dodawanie do listy jednokierunkowej *push_back()*, *size(),remove_if, erase(),* funkcje *begin()*, *end()* dotyczących iteratorów, funkcje związane z *regex* z biblioteki *boost* tj *regex_match()*. Dodatkowo program wzbogacony jest o *wyjątki* a algorytmy z tym związane to np. *what().*

3. Specyfikacja zewnętrzna

Program jest uruchamiany przy użyciu programu testowego. W programie tym zostały przetestowane wszystkie dostępne funkcje programu, tworzenie obiektów poprzez konstruktory, wszystkie destruktory i zwalnianie pamięci.

4.Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem obiektowym. W programie rozdzielono interfejs(komunikację z użytkownikiem) od logiki aplikacji (zarządzenie aptekami). Zastosowano zastosowano algorytmy i iteratory STL, kontenery STL, wyjątki, wyrażenia lambda, regex.

4.1 Ogólna struktura programu

W funkcji głównej, wywołujemy konstruktor klasy miasto, tworząc pierwszą instancję dla przechowywania danych. Konstruktor ten posiada dwa parametry string oraz int. Kolejno dla nazwy miasta oraz jego adresu pocztowego. W polu publicznym posiada on listę, która przechowywuje apteki, które następnie zostaną utworzone poprzez konstruktory z parametrem nazwy apteki. Mając już taką strukturę programu można używać funkcji

dodaj_apteke, która dodaje daną aptekę do listy aptek w klasie miasto, **wypisz_apteki** oraz **usun_apteke**. Nastepnie mozemy przejsc do rozbudowy struktury programu o konstruktory dla pracowników, którzy posiadają **trzy pola** dla **nazwiska, numeru pracowniczego oraz adresu email.** Gdy wywołamy taki konstruktor tworzymy instancję pracownika, która umożliwia nam rozwinięcie metod **dodaj_pracownika_do_apteki**, **pokaz_liste_pracownikow**, **usun_pracownika_RODO** dla obiektu Apteka. Apteka posiada również metodę **dodaj_syrop_do_apteki**, **dodaj_tabletki_do_apteki,** które tworzą odpowiednio obiekty Syrop oraz Tabletka lecz obydwa są przechowane w liście typu Lekarstwa, ponieważ dochodzi do polimorfizmu, dzięki czemu doszło do wyabstrahowania dwóch obiektów. Apteka posiada również dostęp do metody **wypisz_liste_lekarstw** Można także utworzyć osobny obiekt dla konkretnego lekarstwa dzięki czemu można dowiedzieć się za pomocą metody **pokaz_informacje** więcej nt. danego lekarstwa.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku doxyfile na platformie.

5. Testowanie

Program został sprawdzony pod kątem wycieków pamięci. Program został przetestowany na dwóch komputerach. Na obydwóch działał prawidłowo.

6. Wnioski

Program *Zarządzanie aptekami* to mój autorski pomysł. Najwięcej problemów sprawiło poznanie mechaniki działania aptek oraz wymyślenie sposobu przepływu danych. Dzięki temu projektowi nauczyłem się korzystać poprawnie z wielu funkcji biblioteki STL, biblioteki regex, iteratorów stl, obsługi wyjątków. Dzięki researchowi poznałem wiele mechanik związanych z obsługa aptek.

Literatura:

-Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Wprowadzenie do algorytmów. Wydawnictwa Naukowo-Techniczne, Warszawa, 2001.
-https://cpp0x.pl/artykuly/Inne-artykuly/C++-Wyrazenia-regularne-C++11-boost/47

My Project

Generated by Doxygen 1.9.1

Chapter 1

README

Tutaj umieszczać projekt

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Apteka Class Reference

`#include <klasy.h>`

Public Member Functions

- **Apteka** (string apteka)
- void **pokaz_liste_pracownikow** ()
- void **usun_pracownika_RODO** (int numer_pracownika)
- void **dodaj_pracownika_do_apteki** (**Pracownik** &nazwa_pracownika, string email)
- void **dodaj_syrop_do_apteki** (string nazwa, bool recepta, int cena, double pojemnosc)
- void **dodaj_tabletki_do_apteki** (string nazwa, bool recepta, int cena, int pojemnosc)
- void **wypisz_liste_lekarstw** ()
- void **usun_lekarstwo_z_apteki** (string nazwa_lekarstwa, **Apteka** apteka)
- ~**Apteka** ()

Public Attributes

- string **nazwa_apteki**

  Zmienna reprezentujca nazwe apteki.
- list< **Pracownik** > **lista_pracownikow**

  Kontener stl reprezentujcy liste przechowujaca liste pracownikow.
- list< **Lekarstwo** * > **lista_lekarstw**

  Kontener stl reprezentujcy liste przechowujaca liste lekarstw.

5.1.1 Detailed Description

Klasa reprezentujca Apteke

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Apteka()

```
Apteka::Apteka (
            string apteka )  [inline]
```

Konstruktor

Parameters

apteka	Nazwa apteki

5.1.2.2 ∼Apteka()

```
Apteka::∼Apteka ( )  [inline]
```

Destruktor klasy apteka <Kontener stl reprezentujcy liste przechowujaca liste pracownikow

<Kontener stl reprezentujcy liste przechowujaca liste pracownikow

5.1.3 Member Function Documentation

5.1.3.1 dodaj_pracownika_do_apteki()

```
void Apteka::dodaj_pracownika_do_apteki (
             Pracownik & nazwa_pracownika,
            string email )
```

Metoda dodajca pracownika do apteki

Parameters

nazwa_pracownika	Przekazanie obiektu typu **Pracownik** (p. 15)
email	przekazanie emailu pracownika w celu walidacji regex

5.1.3.2 dodaj_syrop_do_apteki()

```
void Apteka::dodaj_syrop_do_apteki (
            string nazwa,
            bool recepta,
            int cena,
            double pojemnosc )
```

Metoda tworzca obiekt typu syrop i dodajca go do listy Lekarstw

Parameters

nazwa	Nazwa lekarstwa
recepta	zmienna bool 1-na recepte 0- bez recepty
cena	zmienna typu int reprezentujca cene
pojemnosc	zmienna typu double reprezentujaca pojemnosc

5.1.3.3 dodaj_tabletki_do_apteki()

```
void Apteka::dodaj_tabletki_do_apteki (
             string nazwa,
             bool recepta,
             int cena,
             int pojemnosc )
```

Metoda tworzca obiekt typu tabletka i dodajca go do listy Lekarstw

Parameters

nazwa	Nazwa lekarstwa
recepta	zmienna bool 1-na recepte 0- bez recepty
cena	zmienna typu int reprezentujca cene
pojemnosc	zmienna typu int reprezentujaca pojemnosc

5.1.3.4 pokaz_liste_pracownikow()

```
void Apteka::pokaz_liste_pracownikow ( )
```

Metoda pokazujca liste pracownikow

5.1.3.5 usun_lekarstwo_z_apteki()

```
void Apteka::usun_lekarstwo_z_apteki (
             string nazwa_lekarstwa,
              Apteka apteka )
```

Metoda usuwajca lekarstwo z listy lekarstw

Parameters

nazwa_lekarstwa	Nazwa lekarstwa typ string
apteka	Obiekt typu **Apteka** (p. 9)

5.1.3.6 usun_pracownika_RODO()

```
void Apteka::usun_pracownika_RODO (
             int numer_pracownika )
```

Metoda usuwajca pracownikow po numerze pracownika

Parameters

numer_pracownika	Numer pracowniczy

5.1.3.7 wypisz_liste_lekarstw()

```
void Apteka::wypisz_liste_lekarstw ( )
```

Metoda wypisujca liste lekarstw

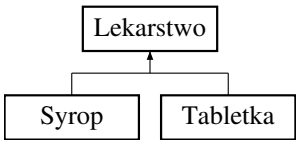
The documentation for this class was generated from the following files:

- **klasy.h**
- klasy.cpp

5.2 Lekarstwo Class Reference

```
#include <klasy.h>
```

Inheritance diagram for Lekarstwo:



Public Member Functions

- **Lekarstwo** (std::string **nazwa**, bool **recepta**, int **cena**)
- virtual void **pokaz_informacje** ()=0

Public Attributes

- string **nazwa**

  Zmienna reprezentujca nazwe leku.
- bool **recepta**

  Zmienna reprezentujca recepte.
- int **cena**

  Zmienna reprezentujca cene.

Friends

- ostream & **operator**<< (ostream &wyjscie, const **Lekarstwo** &s)

5.2.1 Detailed Description

Klasa absrakcyjna reprezentujca lekarstwo

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Lekarstwo()

```
Lekarstwo::Lekarstwo (
            std::string nazwa,
            bool recepta,
            int cena )  [inline]
```

Konstruktor

Parameters

nazwa	Nazwa lekarstwa
recepta	Zmienna typu bool 1-recepta 0- bez recepty
cena	Cena lekarstwa

5.2.3 Member Function Documentation

5.2.3.1 pokaz_informacje()

```
virtual void Lekarstwo::pokaz_informacje ( )  [pure virtual]
```

Wirtualna metoda pokazujaca infromacje nt. lekarstwa

Implemented in **Tabletka** (p. 19), and **Syrop** (p. 18).

5.2.4 Friends And Related Function Documentation

5.2.4.1 operator<<

```
ostream& operator<< (
            ostream & wyjscie,
            const  Lekarstwo & s )  [friend]
```

Operator przeciazony wypisywania

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

- **klasy.h**

5.3 Miasto Class Reference

`#include <klasy.h>`

Public Member Functions

- **Miasto** (string miasto, string zip_code)
- void **wypisz_apteki** ()
- void **usun_apteke** (string nazwa_apteki)
- void **dodaj_apteke** (**Apteka** nazwa_apteki)
- ~**Miasto** ()

Public Attributes

- list< **Apteka** > **apteki**

  Kontener stl typu lista reprezentujca apteki.

5.3.1 Detailed Description

Klasa reprezentujca miasto

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Miasto()

```
Miasto::Miasto (
            string miasto,
            string zip_code )  [inline]
```

Konstruktor

Parameters

miasto	Nazwa miasta
zip_code	Kod pocztowy

5.3.2.2 ~Miasto()

```
Miasto::~Miasto ( )  [inline]
```

Destruktor klasy **Miasto** (p. 14)

5.3.3 Member Function Documentation

5.3.3.1 dodaj_apteke()

```
void Miasto::dodaj_apteke (
              Apteka nazwa_apteki )
```

Metoda dodajca apteke do listy

Parameters

nazwa_apteki	obiekt typu **Apteka** (p. 9)

5.3.3.2 usun_apteke()

```
void Miasto::usun_apteke (
             string nazwa_apteki )
```

Metoda usuwajca apteke z listy

Parameters

nazwa_apteki	Przekazanie nazwy apteki do usunia

5.3.3.3 wypisz_apteki()

```
void Miasto::wypisz_apteki ( )
```

Metoda wypisujca listtek

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

- **klasy.h**
- klasy.cpp

5.4 Pracownik Class Reference

```
#include <klasy.h>
```

Public Member Functions

- **Pracownik** (string nazwisko_prac, int numer_prac, string **email**)

Public Attributes

- string **nazwisko**

  Zmienna reprezentujca nazwisko.
- int **numer_pracownika**

  Zmienna reprezentujca numer pracownika.
- string **email**

  Zmienna reprezentujca email.

Friends

- ostream & **operator**<< (ostream &wyjscie, const **Pracownik** &s)

5.4.1 Detailed Description

Klasa reprezentujca Pracownika

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Pracownik()

```
Pracownik::Pracownik (
            string nazwisko_prac,
            int numer_prac,
            string email )  [inline]
```

Konstruktor

Parameters

nazwisko_prac	nazwisko pracownika typ string
numer_prac	reprezentuje numer pracowniczy typ int
email	email typu string

5.4.3 Friends And Related Function Documentation

5.4.3.1 operator<<

```
ostream& operator<< (
            ostream & wyjscie,
            const  Pracownik & s )  [friend]
```

Przeciony operator wypisywania

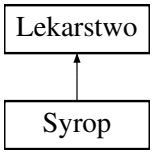
The documentation for this class was generated from the following file:

- **klasy.h**

5.5 Syrop Class Reference

```
#include <klasy.h>
```

Inheritance diagram for Syrop:



Public Member Functions

- **Syrop** (string **nazwa**, bool **recepta**, int **cena**, double pojemnosc)
- virtual void **pokaz_informacje** ()

Additional Inherited Members

5.5.1 Detailed Description

Klasa reprezentujca **Syrop** (p. 17)

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Syrop()

```
Syrop::Syrop (
            string nazwa,
            bool recepta,
            int cena,
            double pojemnosc )  [inline]
```

Konstruktor

Parameters

nazwa	Nazwa lekarstwa
recepta	Zmienna typu bool 1-recepta 0- bez recepty
cena	Cena lekarstwa
pojemnosc	zmienna okreslajaca pojemnosc leku

5.5.3 Member Function Documentation

5.5.3.1 pokaz_informacje()

```
void Syrop::pokaz_informacje ( )  [virtual]
```

Wirtualna metoda pokazujaca infromacje nt. lekarstwa

Implements **Lekarstwo** (p. 13).

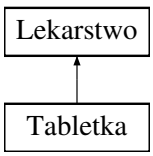
The documentation for this class was generated from the following files:

- **klasy.h**
- klasy.cpp

5.6 Tabletka Class Reference

```
#include <klasy.h>
```

Inheritance diagram for Tabletka:



Public Member Functions

- virtual void **pokaz_informacje** ()
- **Tabletka** (string **nazwa**, bool **recepta**, int **cena**, int pojemnosc)

Additional Inherited Members

5.6.1 Detailed Description

Klasa reprezentujca Tabletki

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Tabletka()

```
Tabletka::Tabletka (
            string nazwa,
            bool recepta,
            int cena,
            int pojemnosc )  [inline]
```

Konstruktor

Parameters

nazwa	Nazwa lekarstwa
recepta	Zmienna typu bool 1-recepta 0- bez recepty
cena	Cena lekarstwa
pojemnosc	zmienna okreslajaca pojemnosc leku

5.6.3 Member Function Documentation

5.6.3.1 pokaz_informacje()

```
void Tabletka::pokaz_informacje ( )  [virtual]
```

Wirtualna metoda pokazujaca infromacje nt. lekarstwa

Implements **Lekarstwo** (p. 13).

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

- **klasy.h**
- klasy.cpp

Chapter 6

File Documentation

6.1 klasy.h File Reference

```
#include <iostream>
#include <list>
```

Classes

- class **Miasto**
- class **Apteka**
- class **Pracownik**
- class **Lekarstwo**
- class **Syrop**
- class **Tabletka**

Index