Politechnika Śląska

Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 3

Implementacja Listy Jednokierunkowej

Autor: Agnieszka Góral  
Prowadzący: dr inż. Jolanta Kawulok  
Rok akademicki: 2021/2022  
Kierunek: informatyka  
Rodzaj studiów: SSI  
Semestr: 3  
Termin laboratorium: wtorek 11:15 – 12:45

Sekcja: 2  
Termin oddania 2022-04-01

1. **OGÓLNY OPIS PROGRAMU**

Celem projektu było zaimplementowanie listy jednokierunkowej i w wybrany sposób przedstawienie jej działania.

* 1. POKAZANIE DZIAŁANIA CZĘŚCI FUNKCJONALNOŚCI LISTY NA PODSTAWIE PROJEKTU „WYSZUKIWANIE LEKÓW”

Za pomocą tej części projektu chciałam pokazać praktyczne zastosowanie listy jednokierunkowej. Celem projektu było stworzenie systemu dopasowywania leków do pacjentów, którzy podają swoje objawy i wiek. Do programu podać należy dwa pliki tekstowe, jeden z informacjami o leku (nazwa, minimalny wiek, maksymalny wiek oraz lista id symptomów, na które działa dany lek), drugi z listą symptomów (id symptomu oraz jego nazwa).

Przykładowy plik z lekami:

Obraz zawierający stół

Opis wygenerowany automatycznie

Przykładowy plik z objawami:

Obraz zawierający stół

Opis wygenerowany automatycznie

Oba rodzaje danych wczytywane są do programu z dwóch osobnych plików. Pacjent podaje swój wiek oraz objawy. W odpowiedzi otrzymuje wszystkie leki, które działają na podane objawy.

Program jest uruchamiany z linii poleceń. Gdzie podawane są parametry wejściowe:

-m (nazwa pliku z lekami np. Medicines.txt)

-s (nazwa pliku z symptomami np. Symptoms.txt)

1.2 POKAZANIE DZIAŁANIA CZĘŚCI FUNKCJONALNOŚCI LISTY NA PODSTAWIE KLASY TESTOWEJ

Klasa testowa, za pomocą której są pokazane funkcjonalności zaimplementowanej listy, głównie te, które nie zostały użyte w pierwszej części programu.

1. **SPECYFIKACJA ZEWNĘTRZNA:**

Program jest uruchamiany z linii poleceń, za pomocą następujących przełączników:

-m (nazwa pliku z lekami np. Medicines.txt)

-s (nazwa pliku z symptomami np. Symptoms.txt)

Użytkownik musi pamiętać, że pliki muszą posiadać rozszerzenie „.txt”. Jeśli użytkownik poda błędne parametry lub program nie będzie w stanie otworzyć pliku to w konsoli pojawi się komunikat informujący o błędzie. Jeśli użytkownik będzie próbował uruchomić program bez podania parametrów to zostaną zastosowane domyślne wartości.

Po uruchomieniu, aby użytkownik mógł wyszukać lek najpierw musi podać swój wiek (bez spacji, tylko cyfry np. 20), następnie ma wymienić swoje objawy - przedzielając je średnikiem, każdy objaw z wielkiej litery, dwuczłonowe wyrazy zapisywane za pomocą „\_” , nie należy używać polskich znaków (przykład podania objawów: Katar;Kaszel;Bol\_glowy).

Obraz zawierający tekst

Opis wygenerowany automatycznie

Po wypisaniu leków program spyta czy użytkownik chce powtórzyć jego działanie i ponownie podać wiek i objawy. Jeśli wpisze „TAK” to w konsoli pokaże się:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Jeśli użytkownik wpisze cokolwiek innego program wyświetli efekt działania klasy testowej (ukazanie działalności listy):

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. **SPECYFIKACJA WEWNĘTRZNA**
   1. ZAIMPLEMENTOWANE KLASY

* **linked\_list (template<class T> class linked\_list{};)**  
  Lista jednokierunkowa w postaci klasy jako szablon.

Prywatne elementy:

* + **struct Node{}; -** struktura, która ilustruje pojedynczy element listy, zawiera: zmienną „data”, która będzie przechowywała dany element listy typu "T" oraz inteligentny wskaźnik na kolejny element.
  + **Inteligentny wskaźnik na głowę listy -** shared\_ptr<Node> head
  + **Inteligentny wskaźnik na ogon listy -** shared\_ptr<Node> tail

Publiczne metody klasy:

* **void save\_to\_bin\_file(const string& file\_name)** – metoda zapisująca dane do pliku binarnego, przyjmująca nazwę pliku .
* **void read\_bin\_file (const string& file\_name)** – metoda odczytująca dane z pliku binarnego, przyjmująca nazwę pliku .
* **void add (T element)** – metoda odpowiedzialna za dodawanie elementu do listy (dodaje element na koniec listy), przyjmująca element do dodania
* **T get(int index)** – metoda odpowiedzialna za zwracanie elementu z listy, przyjmuje indeks elementu, z pod którego chcemy zwrócić dane.
* **void remove(int index)**– metoda odpowiedzialna za usuwanie elementu z listy, przyjmuje indeks elementu, który chcemy usunąć.
* **void sort\_asc()** – metoda odpowiedzialna za sortowanie rosnąco elementów listy (wykorzystany algorytm bubble sort).
* **void sort\_desc()** – metoda odpowiedzialna za sortowanie malejąco elementów listy (wykorzystany algorytm bubble sort)
* **int count()** – metoda odpowiedzialna za zwracanie liczby elementów na liście

W tej klasie zostały użyte także: konstruktor kopiujący,   
konstruktor bezargumentowy, destruktor oraz operatory przypisania/przeniesienia. Dokładne definicje i opisy szczegółowe metod znajdują się w komentarzach w kodzie (również dla pozostałych klas).

* **symptom (class symptom(){})**  
  Klasa obrazująca symptom choroby.

Prywatne elementy:

* + **id symptomu -** int id
  + **nazwa symptomu -** string name

Publiczne metody klasy:

* **int get\_id()** – metoda odpowiedzialna za zwracanie ID symptomu, czyli jego unikalnego numeru
* **string get\_name()** – metoda odpowiedzialna za zwracanie nazwy symptomu
* **medicine (class medicine(){})**  
  Klasa obrazująca dany lek.

Prywatne elementy:

* + **id leku -** int id
  + **nazwa leku -** string name
  + **minimalny wiek zażycia –** int age\_min
  + **maksymalny wiek zażycia –** int age\_max
  + **lista symptomów na jakie działa lek -** inked\_list<symptom> symptoms

Publiczne metody klasy:

* **string convert()** – metoda konwertująca wszystkie elementy klasy do wspólnego stringa, zwraca stringa właśnie z tymi elementami.
* **string get\_name()** – metoda odpowiedzialna za zwracanie nazwy leku
* **int get\_id()** – metoda odpowiedzialna za zwracanie ID leku, czyli jego unikalnego numeru
* **int get\_age\_min()** – metoda odpowiedzialna za zwracanie minimalnego wieku
* **int get\_age\_max()** – metoda odpowiedzialna za zwracanie maksymalnego wieku
* **linked\_list<symptom> get\_symptoms() –** metoda odpowiedzialna za zwracanie listy symptomów danego leku
* **konsola (class konsola(){})**  
   Klasa zajmująca się głównym działaniem programu.

Publiczne metody klasy:

* **static void start(string input\_med, string input\_symp)** – metoda wywołująca kluczowe funkcje programu, przyjmująca nazwę pliku wejściowego dla objawów oraz dla leków.
* **test (class test(){})**  
  Klasa za pomocą której pokazane jest działanie listy.

Prywatne elementy:

* + **int a**
  + **string b**

Publiczne metody klasy:

* **string to\_string()** – metoda łącząca w stringa wszystkie elementy klasy i zwracająca go.
  1. METODY ZAIMPLEMENTOWANE POZA KLASAMI:

**string get\_element(string line, int position, char delimiter) –** metoda, która zwraca podany elemet ze stringa – przyjmuje linie tekstu, pozycje z której chcemy dany ciąg oraz informacje o tym, w jaki sposób są rozdzielone wyrazy lub ich ciągi.

**bool read\_from\_console(int arg, char \*argv[], string &input\_file\_name\_med, string &input\_file\_name\_symp) –** metoda odpowiedzialna za wczytywanie parametrów z konsoli

**int get\_elements\_count(string line, char delimiter) –** metoda, która zwraca ilość elementów w linii, przyjmuje linie tekstu oraz znak jakim te elementy są rozdzielone.

**symptom create\_symptom\_from\_line(string line)–** metoda, która po otrzymaniu linijki tworzy z niej symptom i zwraca otworzony obiekt. (wykorzystywana przy zczytywaniu z pliku)

**linked\_list<symptom> get\_symptoms\_from\_file(string file\_name) –** metoda, która wczytuje symptopmy z pliku i zwraca ich listę, przyjmuje nazwę pliku wejściowego dla symptomów

**linked\_list<medicine> get\_medicines\_with\_symptoms\_from\_files(string &symptom\_file\_name, string &medicine\_file\_name) -** metoda, która wczytuje leki z pliku i zwraca ich listę, przyjmuje nazwę pliku wejściowego dla symptomów i leków, gdyż od razu tworzy powiązania między nimi.

* 1. UŻYTE BIBLIOTEKI:

W programie zostały użyte biblioteki: iostream, string oraz fstram.

* 1. OGÓLNY SCHEMAT DZIAŁANIA PROGRAMU:

Program po otrzymaniu parametrów i sprawdzeniu ich poprawności czyta dane z plików i umieszcza je w obiektach, a te umieszcza w listach. Następnie przy wyszukiwaniu przez użytkownika przeszukuje listy i znajduje pożądane elementy.

1. **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).

Podczas testowania został wyeliminowany przypadek, gdy podany plik wejściowy jest nieprawidłowy oraz przypadek, gdy podczas wyszukiwania leków przez pacjenta jeden z nich odpowiada kilku objawom (nie wyświetli się on drugi raz w finalnym efekcie działania programu, tylko jeden raz wraz z wypisanymi objawami na jakie działa).

Program został przetestowany pod kątem wycieków pamięci za pomocą dostępnych w programie Clion narzędzi.