

Міністерство освіти та науки України  
Київський політехнічний інститут ім.. І. Сікорського  
Кафедра автоматизованих систем обробки інформації та управління

## КУРСОВА РОБОТА

з дисципліни «Об'єктно-орієнтоване програмування»

на тему: Програмна система підтримки обліку пацієнтів для реєстратури  
поліклініки

Студента II курсу групи ПІ-71

Спеціальності 121 «Інженерія програмного забезпечення»

Фоміна Владислава Віталійовича

Керівник асистент кафедри АСОІУ Пономаренко Р.М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_

Національна оцінка \_\_\_\_\_

Члени комісії

_____	ас. каф. АСОІУ Пономаренко Р.М.
(підпис)	(вчене звання, науковий ступінь, прізвище та ініціали)
_____	к.н.т. доц. каф. АСОІУ Муха І.П.
(підпис)	(вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2018 рік

Київський політехнічний інститут ім.. І. Сікорського

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Об'єктно-орієнтоване програмування

Спеціальність «Інженерія програмного забезпечення»

Курс 2 Група ІІ-71

Семестр 3

### **ЗАВДАННЯ на курсову роботу студента**

**Фоміна Владислава Віталійовича**

(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система підтримки обліку пацієнтів для реєстратури поліклініки

2. Строк здачі студентом закінченої роботи \_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_

---

---

---

---

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

---

---

5. Перелік графічного матеріалу ( з точним зазначенням обов'язкових креслень )

---

---

---

---

6. Дата видачі завдання 01.11.2018

## КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	01.11.2018	
2.	Підготовка ТЗ	05.11.2018	
3.	Аналіз предметної області	07.11.2018	
4.	Проектування архітектури програмної системи	09.11.2018	
5.	Розробка сценарію роботи програми	11.11.2018	
6.	Узгодження з керівником інтерфейсу користувача	11.11.2018	
7.	Розробка програмного забезпечення	25.11.2018	
8.	Узгодження з керівником плану тестування	05.12.2018	
9.	Тестування програми	10.12.2018	
10.	Підготовка пояснювальної записки	18.12.2018	
11.	Здача курсової роботи на перевірку	21.12.2018	
12.	Захист курсової роботи	26.12.2018	

Студент \_\_\_\_\_  
(підпис)

Фомін Владислав Віталійович  
(прізвище, ім'я, по батькові)

Керівник \_\_\_\_\_  
(підпис)

Пономаренко Роман Миколайович  
(прізвище, ім'я, по батькові)

"\_\_" \_\_\_\_\_ 2018 р.

## **АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: 48 сторінок, 43 рисунки, 3 таблиці, 4 посилання.

Об'єкт дослідження: програмна система підтримки обліку пацієнтів для реєстратури поліклініки.

Мета роботи: створення програмного забезпечення для реєстратури поліклініки.

Детально вивчено процес обслуговування пацієнтів у реєстратурі.  
Розроблено алгоритм, що значно спрощує взаємодію працівника і пацієнта.

Виконана програмна реалізація системи підтримки обліку пацієнтів для реєстратури поліклініки.

## ЗМІСТ

<b>1</b>	<b>ВСТУП.....</b>	<b>6</b>
<b>2</b>	<b>АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ .....</b>	<b>4</b>
2.1	<i>Опис програми .....</i>	6
2.2	<i>Словник іменників та дієслів .....</i>	9
2.3	<i>Функціональні вимоги .....</i>	6
2.4	<i>Діаграма прецедентів.....</i>	9
2.5	<i>Сценарій роботи програми .....</i>	6
<b>3</b>	<b>ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ.....</b>	<b>4</b>
3.1	<i>UML-діаграма класів .....</i>	6
<b>4</b>	<b>ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....</b>	<b>4</b>
4.1	<i>Опис класів та їх методів.....</i>	6
<b>4</b>	<b>ТЕСТУВАННЯ .....</b>	<b>6</b>
<b>5</b>	<b>ІНСТРУКЦІЯ КОРИСТУВАЧА.....</b>	<b>4</b>
5.1	<i>Призначення програми.....</i>	6
5.2	<i>Вимоги до системи .....</i>	9
5.3	<i>Інструкція з використання та опис інтерфейсу користувача .</i>	6
	<b>ВИСНОВКИ .....</b>	<b>10</b>
	<b>ПЕРЕЛІК ПОСИЛАНЬ.....</b>	<b>4</b>
	<b>ДОДАТКИ.....</b>	<b>4</b>

## **ВСТУП**

Курсовий проект призначений для спрощення та прискорення взаємодії працівника реєстратури лікарні та пацієнтів, для мінімізації архівних матеріалів та зменшення обсягів рутинної роботи, адже, використовуючи програмне забезпечення, розроблене в рамках даної курсової роботи, пацієнту потрібно лише знати номер своєї медичної книги, щоб мати повну інформацію про себе та історію хвороб.

Значно легше натиснути декілька кнопок, щоб внести інформацію до історії пацієнта, ніж знайти його медичну книгу серед архівів, знайти вільне місце, записати інформацію та покласти на місце. А щоб відредагувати яку-небудь інформацію про пацієнта більше не потрібно закреслювати, замальовувати якісь слова в книзі, а ще гірше – міняти її на нову. Потрібно лише натиснути клавішу «Редагувати інформацію про пацієнта».

## АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

### Опис програми

1. Користувач може ввести дані про нового пацієнта, такі як: прізвище, ім'я, по-батькові, ідентифікаційний номер, прописку, дату народження, номер телефону, номер мед. картки.
2. У процесі роботи користувач може змінювати (редагувати) дані про існуючого пацієнта, такі як: прізвище, ім'я, по батькові, ідентифікаційний номер, прописку, дату народження, номер телефону, номер мед. картки.
3. У процесі роботи користувач може додавати існуючому пацієнтові в історію хвороб нові дані, такі як: дату відвідування лікаря, прізвище лікаря, ініціали лікаря, хвороба (власне, через що пацієнт звернувся до лікаря), коментар (необов'язкове поле, при необхідності - заповнити).
4. Користувач має можливість переглянути (вивести на екран) інформацію про існуючого пацієнта, таку як: прізвище, ім'я, по батькові, ідентифікаційний номер, прописку, дату народження, номер телефону, (далі буде виведений такий список) дата відвідування лікаря, прізвище лікаря, ініціали лікаря, хвороба (власне, через що пацієнт звернувся до лікаря), коментар.
5. Користувач має можливість додати інформацію про нового лікаря, таку як: прізвище лікаря, ініціали лікаря, спеціальність (наприклад, хірург), кабінет (до якого приписаний даний лікар), номер телефону лікаря.
6. Користувач має можливість редагувати дані про існуючого лікаря, такі як: прізвище лікаря, ініціали лікаря, спеціальність (наприклад, хірург), кабінет (до якого приписаний даний лікар), номер телефону лікаря.
7. Користувач має можливість додати запис в звітний лист реєстратури. У записі зберігаються такі дані: прізвище та ініціали пацієнта, прізвище та ініціали лікаря, дата додавання записи.

### Словник іменників та дієслів

Іменники	Дієслова
Користувач	Взаємодіє
Новий пацієнт	Створює об'єкт
Існуючий пацієнт	Зберігає, відображає об'єкт
Новий лікар	Створює об'єкт

Існуючий лікар	Зберігає, відображає об'єкт
Реєстратура	Створює, зберігає, відображає записи
Прізвище Ім'я По-батькові Ідентифікаційний номер Прописку Дату народження Номер телефону Номер мед. картки	Записуються Зберігаються Редагуються Проглядаються
Прізвище лікаря Ініціали лікаря Спеціальність (наприклад, хірург) Кабінет (до якого приписаний даний лікар) Номер телефону лікаря	Записуються Зберігаються Редагуються Проглядаються
Дата відвідування лікаря Прізвище лікаря Ініціали лікаря Хвороба (власне, через що пацієнт звернувся до лікаря) Коментар	Записуються Зберігаються Редагуються Проглядаються
Прізвище та ініціали пацієнта Прізвище та ініціали лікаря Дата додавання записи	Записуються Зберігаються Редагуються Проглядаються

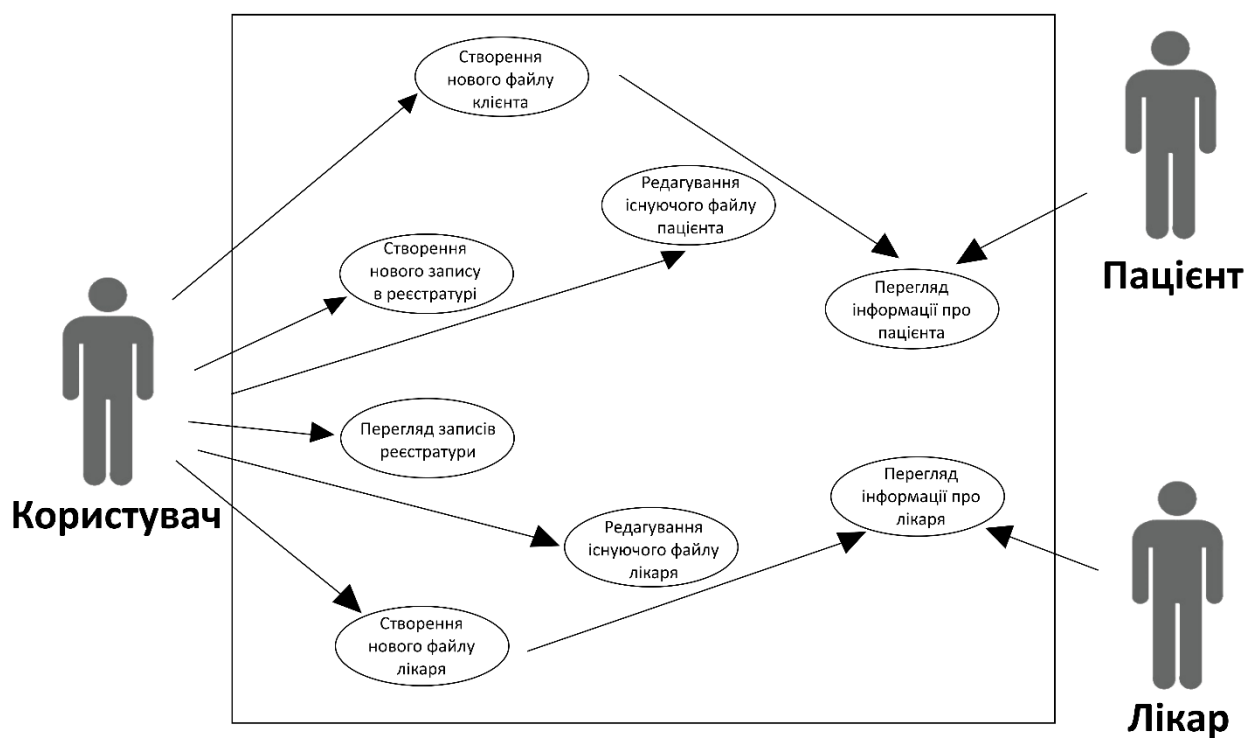
#### Функціональні вимоги:

- Можливість ввести дані про нового пацієнта
- Можливість відредагувати існуючі дані про пацієнта
- Можливість додати інформацію про хворобу пацієнта
- Можливість проглянути інформацію про пацієнта
- Можливість додати інформацію про лікаря



- Можливість відредагувати інформацію про лікаря
- Можливість додати запис у журнал реєстратури
- Можливість проглянути записи у журналі реєстратури

### Діаграма прецедентів



### Сценарій роботи програми:

Після запуску програми перед нами з'являється наступне вікно:




Ми маємо обрати те, з чим ми бажаємо взаємодіяти:

- Журнал реєстратури
- Пацієнт
- Лікар
- Справка

Якщо ми обираємо взаємодію з журналом реєстратури, ми можемо додати запис до журналу реєстратури, проглянути всі записи в журналі та проглянути всі записи конкретного лікаря.

### 1) Додавання запису до журналу реєстратури

Добавление записи в журнал регистратуры



ФИО пациента:

Номер мед. книги:

ФИО врача:

Добавить информацию

### 2) Перегляд всіх записів журналу реєстратури

Записи в журнале регистратуры

	Дата	ФИО пациента	Номер мед. книги	Врач
1	Dec 12 2018 17:08:37	Kolik M.P.	6442	Doctor Who
2	Dec 12 2018 17:31:29	PATIENT	1337	DOCTOR
3	Dec 12 2018 17:31:29	ASDASD	111111	ASDASDASD
4	Dec 12 2018 17:08:37	Kolik M.P.	6442	Doctor Who
5	Dec 12 2018 17:31:29	PATIENT	1337	DOCTOR
6	Dec 12 2018 17:31:29	ASDASD	111111	ASDASDASD
7	Dec 12 2018 17:08:37	Kolik M.P.	6442	Doctor Who
8	Dec 12 2018 17:31:29	PATIENT	1337	DOCTOR
9	Dec 12 2018 17:31:29	ASDASD	111111	ASDASDASD
10	Dec 12 2018 18:16:24	FIO	1337	FIO
11	Dec 12 2018 18:17:55	TEST	6442	test
12	Dec 12 2018 17:08:37	Kolik M.P.	6442	Doctor Who
13	Dec 12 2018 17:08:37	Kolik M.P.	6442	Doctor Who

Получить информацию

### 3) Перегляд всіх записів конкретного лікаря в журналі реєстратури

Записи выбранного доктора в журнале регистратуры ×

Введите ФИО доктора:

	Дата	ФИО пациента	Номер мед. книги	Врач
1	Dec 12 2018 17:31:29	PATIENT	1337	DOCTOR
2	Dec 12 2018 17:31:29	PATIENT	1337	DOCTOR
3	Dec 12 2018 17:31:29	PATIENT	1337	DOCTOR

Якщо ми обираємо взаємодію з пацієнтом, ми можемо додати інформацію про нового пацієнта, редагувати та проглянути інформацію про існуючого пацієнта. Ще можемо додати хворобу в історію хвороб пацієнта.

#### 1) Додавання інформації про нового пацієнта

Добавление информации о новом пациенте ×

Фамилия

Имя

Отчество

Идентификационный номер


Дата рождения

Прописка

ID медицинской книги

Номер телефона

## 2) Редагування інформації про пацієнта


Редактирование информации о пациенте
✕

Введите ID медицинской книги

☐ Фамилия

☐ Имя

☐ Отчество

☐ Идентификационный номер


☐ Дата рождения

☐ Прописка

☐ Номер телефона

Редактировать

## 3) Перегляд інформації про пацієнта


Информация о пациенте
✕

Номер мед. карты  Идент. номер: 65061042640

Фамилия: Fomin Дата рождения: 10.11.1999


Имя: Vladyslav Прописка: Uman, Ukraine

Отчество: Vitalievich Номер телефона: 5555555555555555555

	Дата	Врач	Болезнь	Комментарий
1	10.12.2018	Fomin V.V.	No disease	
2	10.13.2018	Donka V.S.	Disease	Its my comment!
3	10.14.2018	Doccc name	Disease name	Comment sadasd
4	11.12.2018	Doctor	Disease	
5	10.12.2018	Fomin V.V.	No disease	
6	10.13.2018	Donka V.S.	Disease	Its my comment!
7	10.14.2018	Doccc name	Disease name	Comment sadasd
8	11.12.2018	Doctor	Disease	

Получить информацию

#### 4) Додавання інформації про хворобу

 Додание информации о болезни ×

Введите ID медицинской книги

Дата

Имя врача


Болезнь

☐ Комментарий

Добавить

Якщо ми обираємо взаємодію з лікарем, ми можемо додати інформацію про нового лікаря, редагувати та проглянути інформацію про існуючого лікаря.

##### 1) Додавання інформації про нового лікаря

 Добавление информации о новом докторе ×

Фамилия:

Имя:

Отчество:


Специальность:

Кабинет:

Номер телефона:

Добавить

##### 2) Редагування інформації про лікаря

 Редактирование информации о докторе ✕

Введите фамилию:

Введите специальность:

☐ Фамилия

☐ Имя

☐ Отчество


☐ Специальность

☐ Кабинет

☐ Номер телефона

Редактировать

### 3) Перегляд інформації про лікаря

 Информация о докторе ✕

Фамилия:

Специальность:

Фамилия:

Имя:

Отчество:

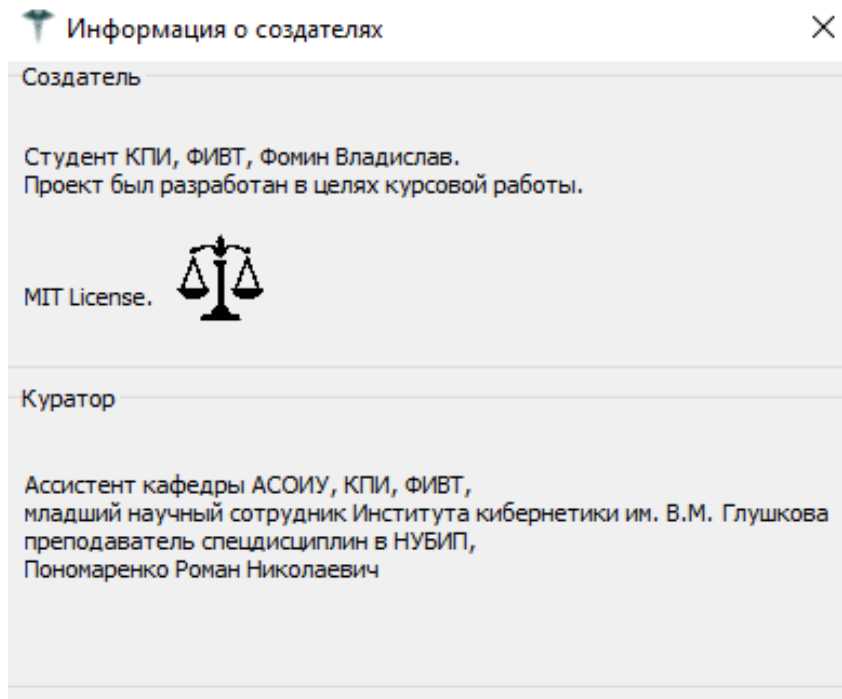
Специальность:

Кабинет:

Номер телефона:

Получить информацию

Ми можемо переглянути справку, тобто, інформацію про творців даного програмного забезпечення. Показано на наступному скріншоті:





# ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

## UML діаграма класів

### Course Work



Я використовую паттерн «Singleton» та «Chaining» у класі «Registry».

Одинак (англ. *Singleton*) — шаблон проектування, відноситься до класу твірних шаблонів. Гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра.

Для деяких класів важливо, щоб існував тільки один екземпляр. Наприклад, хоча у системі може існувати декілька принтерів, може бути тільки один спулер. Повинна бути тільки одна файлова система та тільки один активний віконний менеджер.

Глобальна змінна не вирішує такої проблеми, бо не забороняє створити інші екземпляри класу.

Рішення полягає в тому, щоб сам клас контролював свою «унікальність», забороняючи створення нових екземплярів, та сам забезпечував єдину точку доступу. Це є призначенням шаблону *Одинак*.

Іноді, при використанні або написанні великих класів виникає необхідність викликати поспіль кілька методів об'єкта цього класу.

Прийом «Chaining» дозволяє скоротити код. Для цього ми в кожному нашому методі повернемо посилання на наш об'єкт і вибудуємо виклики в ланцюжок.

## ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### Опис класів та їх методів

№ п/ п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	Disease	addNewObject	Встановлює дані в поля об'єкта даного класу за вхідними параметрами без коментаря	string date – дата створення запису string doctorName – ім'я лікаря string diseaseName – назва хвороби	Немає	disease.h
2		addNewObject	Встановлює дані в поля об'єкта даного класу за вхідними параметрами з коментарем	string date – дата створення запису string doctorName – ім'я лікаря string diseaseName – назва хвороби comment – коментар до запису	Немає	
3		setDate	Встановлює дані в поле date об'єкта даного класу за вхідним параметром	date – дата створення запису	Немає	
4		setDoctorName	Встановлює дані в поле doctorName об'єкта даного класу за вхідним параметром	string doctorName – ім'я доктора	Немає	
5		setDiseaseName	Встановлює дані в поле diseaseName об'єкта даного класу за вхідним параметром	string diseaseName – назва хвороби	Немає	
6		setComment	Встановлює дані в поле comment об'єкта даного класу за вхідним параметром	string comment - коментар	Немає	
7		getDate	Повертає дані, які знаходяться в	Немає	Повертає значення поля string date	

			полі date об'єкта даного класу			
8		getDiseaseName	Повертає дані, які знаходяться в полі diseaseName об'єкта даного класу	Немає	Повертає значення поля string diseaseName	
9		getDoctorName	Повертає дані, які знаходяться в полі doctorName об'єкта даного класу	Немає	Повертає значення поля string doctorName	
10		getComment	Повертає дані, які знаходяться в полі comment об'єкта даного класу	Немає	Повертає значення поля string comment	
11		initialize	Створює один і єдиний об'єкт класу, з яким надалі є можливість взаємодіяти	Немає	Повертає посилання на об'єкт класу, який може бути один і тільки один	
12		addNote	Додати запис до журналу реєстратури	string patientName – ім'я пацієнта string idMedBook – номер медичної книги пацієнта string doctorName – ім'я лікаря	Повертає вказівник на даний об'єкт, щоб ми могли використати паттерн «чейнінг»	
13		readNoteList	Заповнити поля об'єкту класу з файлу реєстратури	Немає	Повертає вказівник на даний об'єкт, щоб ми могли використати паттерн «чейнінг»	
14		printNoteList	Вивести журнал записів на екран	Немає	Немає	
15		search	Пошук записів конкретного доктора	string doctorName – ім'я лікаря, по якому ми будемо шукати записи	Повертає вказівник на даний об'єкт, щоб ми могли використати паттерн «чейнінг»	
16		printDoctorNoteList	Вивести журнал записів конкретного доктора на екран	Немає	Немає	
17		getNotes	Повертає дані, які знаходяться в полі notes об'єкта даного класу	Немає	Повертає list Notes, тобто, записів журналу реєстратури	

Registry

registry.h

18		getDoctorNotes	Повертає дані, які знаходяться в полі doctorNotes об'єкта даного класу	Немає	Повертає list doctorNotes, тобто, записів журналу реєстратури конкретного лікаря	
19	People	addInfoKeyboard	Віртуальна функція, перевизначена у похідних класах	string surname – прізвище пацієнта string name – ім'я пацієнта string patronymic – по-батькові пацієнта string identNum – ідентифікаційний номер string dateOfBirth – дата народження string registration - прописка string phoneNumber – номер телефону string id – номер медичної книги пацієнта	Немає	people.h
20		addInfoFile	Віртуальна функція, перевизначена у похідних класах	string filename – ім'я файлу	Немає	
21		addNewObject	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
22		redactObject	Віртуальна функція, перевизначена у похідних класах	int num – номер, який ми надалі передаємо до конструкції switch string redactArg – параметр, який замінимо надалі	Немає	
23		printObject	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
24		addDisease	Віртуальна функція, перевизначена у похідних класах	string date – дата відвідування лікаря string doctorName – ім'я лікаря	Немає	

				string diseaseName – назва хвороби		
25		addDisease	Віртуальна функція, перевизначена у похідних класах	string date – дата відвідування лікаря string doctorName – ім'я лікаря string diseaseName – назва хвороби string comment – коментар, залишений лікарем	Немає	
26		getSurname	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
27		getName	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
28		getPatronymic	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
29		getIdNum	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
30		getDateOfBirth	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
31		getRegistration	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
32		getPhoneNumber	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
33		getDiseasesCount	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
34		getDiseases	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
35		addInfoKeyboard	Віртуальна функція, перевизначена у похідних класах	string surname – прізвище лікаря string name – ім'я лікаря	Немає	

				string patronymic – по-батькові лікаря string speciality – спеціальність лікаря string cabinet – кабінет лікаря string phoneNumber – номер телефону лікаря		
36		getSpeciality	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
37		getCabinet	Віртуальна функція, перевизначена у похідних класах	Немає	Немає	
38	Patient	addInfoKeyboard	Заповнює значення полів даного класу за значеннями, які передані в аргументи	string surname – прізвище пацієнта string name – ім'я пацієнта string patronymic – по-батькові пацієнта string identNum – ідентифікаційний номер string dateOfBirth – дата народження string registration - прописка string phoneNumber – номер телефону string id – номер медичної книги пацієнта	Немає	patient.h
39		addInfoFile	Заповнює значення полів даного класу за значеннями, які наявні у файлі	string filename – ім'я файлу	Немає	
40		addNewObject	Створює новий об'єкт (у контексті наявного функціоналу - новий файл)	Немає	Немає	
41		redactObject	Дає змогу редагувати інформацію обраного поля за	int num – номер, який ми надалі передаємо до	Немає	

			номером, переданим в аргументи та змінювати його на значення, також передане в аргументи	конструкції switch string redactArg – параметр, який замінимо надалі		
42		printObject	Дає змогу вивести дані на екран	Немає	Немає	
43		addDisease	Додає хворобу до списку хвороб пацієнта без коментаря	string date – дата відвідування лікаря string doctorName – ім'я лікаря string diseaseName – назва хвороби	Немає	
44		addDisease	Додає хворобу до списку хвороб пацієнта з коментарем	string date – дата відвідування лікаря string doctorName – ім'я лікаря string diseaseName – назва хвороби string comment – коментар, залишений лікарем	Немає	
45		getSurname	Надає доступ до поля surname	Немає	Повертає string значення поля surname	
46		getName	Надає доступ до поля name	Немає	Повертає string значення поля name	
47		getPatronymic	Надає доступ до поля patronymic	Немає	Повертає string значення поля patronymic	
48		getIdentNum	Надає доступ до поля identNum	Немає	Повертає string значення поля identNum	
49		getDateOfBirth	Надає доступ до поля dateOfBirth	Немає	Повертає string значення поля dateOfBirth	
50		getRegistration	Надає доступ до поля registration	Немає	Повертає string значення поля registration	



51		getPhoneNumber	Надає доступ до поля phoneNumber	Немає	Повертає string значення поля phoneNumber	
52		getDiseasesCount	Дає змогу дізнатись кількість хвороб у пацієнта	Немає	Повертає int значення кількості хвороб у списку	
53		getDiseases	Надає доступ до списку хвороб пацієнта	Немає	Повертає list <Disease> - список хвороб	
54		addInfoKeyboard	В даному класі – перевизначена віртуальна функція	string surname – прізвище лікаря string name – ім'я лікаря string patronymic – по-батькові лікаря string speciality – спеціальність лікаря string cabinet – кабінет лікаря string phoneNumber – номер телефону лікаря	Немає	
55		getSpeciality	В даному класі – перевизначена віртуальна функція	Немає	Повертає string «», так як це поле відсутнє у даному класі і це лише перевизначена функція - метод	
56		getCabinet	В даному класі – перевизначена віртуальна функція	Немає	Повертає string «», так як це поле відсутнє у даному класі і це лише перевизначена функція - метод	
57	Doctor	addInfoKeyboard	В даному класі – перевизначена віртуальна функція	string surname – прізвище пацієнта string name – ім'я пацієнта string patronymic – по-батькові пацієнта string identNum – ідентифікаційний номер	Немає	doctor.h

				string dateOfBirth – дата народження string registration - прописка string phoneNumber – номер телефону string id – номер медичної книги пацієнта		
58		addInfoFile	Заповнює значення полів даного класу за значеннями, які наявні у файлі	string filename – ім'я файлу	Немає	
59		addNewObject	Створює новий об'єкт (у контексті наявного функціоналу - новий файл)	Немає	Немає	
60		redactObject	Дає змогу редагувати інформацію обраного поля за номером, переданим в аргументи та змінювати його на значення, також передане в аргументи	int num – номер, який ми надалі передаємо до конструкції switch string redactArg – параметр, який замінимо надалі	Немає	
61		printObject	Дає змогу вивести дані на екран	Немає	Немає	
62		addDisease	В даному класі – перевизначена віртуальна функція	string date – дата відвідування лікаря string doctorName – ім'я лікаря string diseaseName – назва хвороби	Немає	
63		addDisease	В даному класі – перевизначена віртуальна функція	string date – дата відвідування лікаря string doctorName – ім'я лікаря string diseaseName – назва хвороби string comment – коментар, залишений лікарем	Немає	

64		getSurname	Надає доступ до поля surname	Немає	Повертає string значення поля surname	
65		getName	Надає доступ до поля name	Немає	Повертає string значення поля name	
66		getPatronymic	Надає доступ до поля patronymic	Немає	Повертає string значення поля patronymic	
67		getIdentNum	В даному класі – перевизначена віртуальна функція	Немає	Повертає string «», так як це поле відсутнє у даному класі і це лише перевизначена функція - метод	
68		getDateOfBirth	В даному класі – перевизначена віртуальна функція	Немає	Повертає string «», так як це поле відсутнє у даному класі і це лише перевизначена функція – метод	
69		getRegistration	В даному класі – перевизначена віртуальна функція	Немає	Повертає string «», так як це поле відсутнє у даному класі і це лише перевизначена функція - метод	
70		getPhoneNumber	Надає доступ до поля phoneNumber	Немає	Повертає string значення поля phoneNumber	
71		getDiseasesCount	В даному класі – перевизначена віртуальна функція	Немає	Повертає int 0, так як це поле відсутнє у даному класі і це лише перевизначена функція - метод	
72		getDiseases	В даному класі – перевизначена віртуальна функція	Немає	Повертає list <Disease> - список хвороб	
73		addInfoKeyboard	Заповнює значення полів даного класу за	string surname – прізвище лікаря	Немає	

			значеннями, які передані в аргументи	string name – ім'я лікаря string patronymic – по-батькові лікаря string speciality – спеціальність лікаря string cabinet – кабінет лікаря string phoneNumber – номер телефону лікаря		
74		getSpeciality	Надає доступ до поля speciality	Немає	Повертає string значення поля speciality	
75		getCabinet	Надає доступ до поля cabinet	Немає	Повертає string значення поля cabinet	

## ТЕСТУВАННЯ

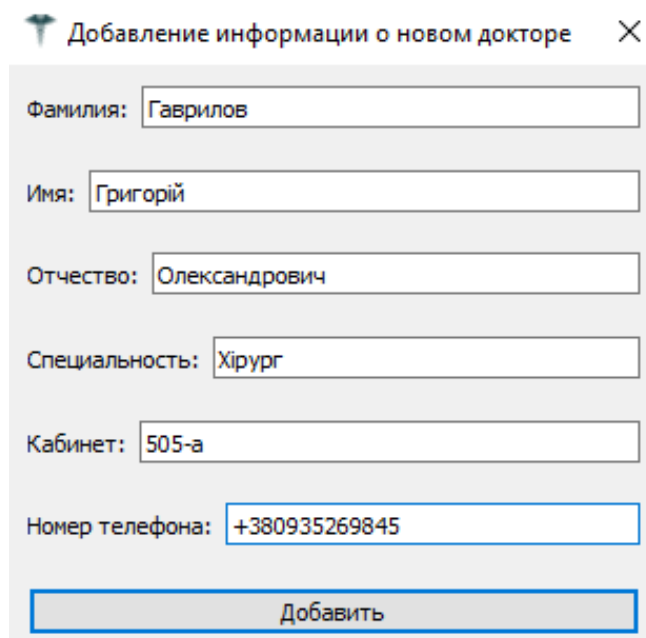
Для коректного тестування програми необхідно змодельовати ситуацію, з якою наша програма і має працювати.

Уявімо такий випадок: у лікарню прийшов лікар, який щойно випустився з університету та хоче працювати тут. Ми маємо занести дані про нього в базу даних нашої програми. Зробимо це наступним чином:

У вкладці «Доктор» оберемо «Добавить информацию о новом докторе».

Вхідні дані:

- Прізвище: Гаврилов
- Ім'я: Григорій
- По-батькові: Олександрович
- Спеціальність: Хірург
- Кабінет: 505-а
- Номер телефону: +380635269845



Добавление информации о новом докторе

Фамилия: Гаврилов

Имя: Григорий

Отчество: Олександрович

Специальность: Хірург

Кабинет: 505-а

Номер телефона: +380935269845

Добавить

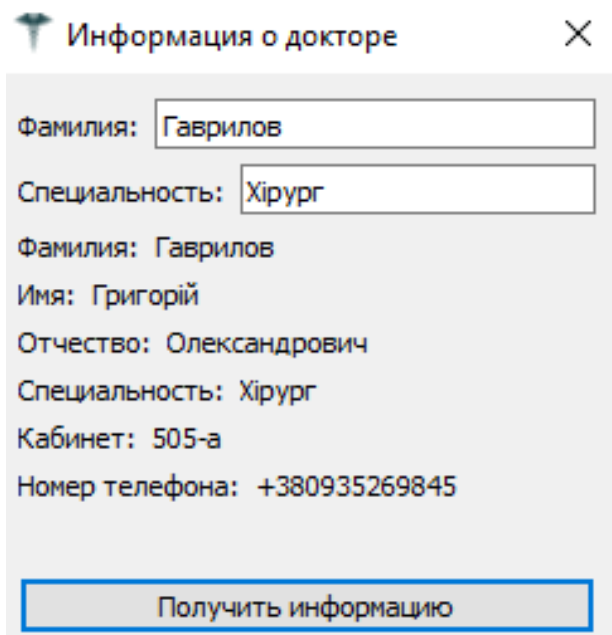
Натиснемо кнопку «Добавить». Все, інформація успішно збереглась, адже нам не прийшло сповіщення про помилку.

Далі ми маємо перевірити, чи справді наша інформація є в базі даних.

Для цього у вкладці «Доктор» оберемо «Посмотреть информацию о докторе».

Вхідні дані:

- Прізвище: Гаврилов
- Спеціальність: Хірург



Інформація о докторе

Фамилия: Гаврилов

Специальность: Хірург

Фамилия: Гаврилов

Имя: Григорій

Отчество: Олександрович

Специальность: Хірург

Кабинет: 505-а

Номер телефона: +380935269845

Получить информацию

Бачимо, що все дійсно збереглося. Але ми помітили, що працівник реєстратури допустив помилку, вказавши в номері мобільного телефону не +380635269845, а +380935269845. Ми можемо це виправити.

Для цього у вкладці «Доктор» оберемо «Редактировать информацию о докторе».

Вхідні дані:

- Прізвище: Гаврилов
- Спеціальність: Хірург
- Коректна інформація, яку ми хочемо відредагувати (відмітивши поле «галочкою»): +380635269845

Редактирование информации о докторе

Введите фамилию: Гаврилов

Введите специальность: Хирург

☐ Фамилия

☐ Имя

☐ Отчество

☐ Специальность

☐ Кабинет

☒ Номер телефона +380635269845

Редактировать

Для перевірки коректності роботи програми, проглянемо ще раз інформацію про лікаря. Бачимо наступне:

Информация о докторе

Фамилия: Гаврилов

Специальность: Хирург

Фамилия: Гаврилов

Имя: Григорий

Отчество: Олександрович

Специальность: Хирург

Кабинет: 505-а

Номер телефона: +380635269845

Получить информацию

Отже, тест програми щодо операцій з лікарями пройдено успішно. А якщо ж до нас завітає новий пацієнт, якого раніше не лікували в нашій лікарні?!

Перш за все потрібно додати інформацію про нового пацієнта.

Для цього у вкладці «Пациент» оберемо «Добавить информацию о новом пациенте».

Вхідні дані:

- Прізвище: Заграва

- Ім'я: Василь
- По-батькові: Львович
- Ідентифікаційний номер: 615622641657661782
- Дата народження: 26.06.1973
- Прописка: Київ, проспект Перемоги 27, кв. 15
- Номер медичної книги: 6232481
- Номер телефону: +380631244587

Добавление информации о новом пациенте

Фамилия

Заграва

Имя

Василь

Отчество

Львович

Идентификационный номер

615622641657661782

Дата рождения

26.06.1973

Прописка

Київ, проспект Перемоги 27, кв. 15

ID медицинской книги

6232481

Номер телефона

+380631244587

Добавить

Перевіримо, чи збереглись наші дані про пацієнта.

Для цього у вкладці «Пациент» оберемо «Посмотреть информацию о пациенте».

Вхідні дані:

- Номер медичної книги: 6232481

Информация о пациенте

Номер мед. карты

Идент. номер:

6232481

615622641657661782

Фамилия: Заграва

Дата рождения:

26.06.1973

Имя: Василь

Прописка: Київ, проспект Перемоги 27, кв. 15

Отчество: Львович

Номер телефона: +380631244587

Дата	Врач	Болезнь	Комментарий

Получить информацию



І знову по необережності наш працівник реєстратури вказав некоректну інформацію. Потрібно в прописку додати назву країни.

Для цього у вкладці «Пациент» оберемо «Редактировать информацию о пациенте».

Вхідні дані:

- Номер медичної книги: 6232481
- Коректна інформація, яку ми хочемо відредагувати (відмітивши поле «галочкою»): Україна, Київ, проспект Перемоги 27, кв. 15

І відразу ж додамо інформацію про хворобу. У нашого пацієнта цукровий діабет.

Для цього у вкладці «Пациент» оберемо «Добавить информацию о болезни».

Вхідні дані:

- Номер медичної книги: 6232481
- Дата: 10.12.2018
- Ім'я лікаря: Гаврилов Григорій Олександрович
- Хвороба: Цукровий діабет
- Коментар (відмітити поле «галочкою», при необхідності): Колоти інсулін двічі на день по 1 «кубику».

Добавление информации о болезни

Введите ID медицинской книги

Дата

Имя врача

Болезнь

☒ Комментарий

Добавить

Перевіримо, чи збереглася інформація про пацієнта.

Информация о пациенте

Номер мед. карты  Идент. номер: 615622641657661782

Фамилия: Заграва Дата рождения: 26.06.1973

Имя: Василь Прописка: Україна, Київ, проспект Перемоги 27, кв. 15

Отчество: Львович Номер телефона: +380631244587

	Дата	Врач	Болезнь	Комментарий
1	10.12.2018	Гаврилов Григ...	Цукровий діабет	Колоти інсулін...

Получить информацию

Як бачимо, все збережено правильно та редагування даних пройшло успішно. Ще потрібно протестувати коректність роботи журналу реєстратури.

Для цього змодельуємо наступну ситуацію:

Пацієнт прийшов на мед. огляд до лікаря. Працівник реєстратури має зробити відповідний запис в журналі реєстратури.


Для цього у вкладці «Регистратура» оберемо «Добавить запис».

Вхідні дані:

- ПІБ пацієнта: Заграва В.Л.
- Номер медичної книги пацієнта: 6232481

- ПІБ лікаря: Гаврилов Г.О.

Добавление записи в журнал регистратуры


 ФИО пациента:

Номер мед. книги:

ФИО врача:

Проглянемо, чи збереглися наші дані.  
Для цього у вкладці «Регистратура» оберемо «Посмотреть журнал».

Записи в журнале регистратуры


	Дата	ФИО пациента	Номер мед. книги	Врач
1	Дес 12 2018 11:45:05	Куцик А.Л.	6232481	Гаврилов Г.О.
2	Дес 13 2018 10:31:22	Болотенюк В.І.	6232481	Петров А.І.
3	Дес 13 2018 13:15:15	Кудря І.І.	6232481	Петров А.І.
4	Дес 13 2018 14:23:18	Іванов П.П.	6232481	Гаврилов Г.О.
5	Дес 14 2018 09:51:03	Антосев Д.І.	6232481	Петров А.І.
6	Дес 14 2018 12:32:21	Федорович І.К.	6232481	Петров А.І.
7	Дес 15 2018 16:49:45	Заграва В.Л.	6232481	Гаврилов Г.О.

Все успішно збереглося. Але ми хочемо переглянути активність роботи лікаря Гаврилова Г.О.

Для цього у вкладці «Регистратура» оберемо «Посмотреть записи доктора».

Вхідні дані:

- ПІБ лікаря: Гаврилов Г.О.

 Записи выбранного доктора в журнале регистратуры ×

Введите ФИО доктора:

	Дата	ФИО пациента	Номер мед. книги	Врач
1	Dec 12 2018 11:45:05	Куцик А.Л.	6232481	Гаврилов Г.О.
2	Dec 13 2018 14:23:18	Иванов П.П.	6232481	Гаврилов Г.О.
3	Dec 15 2018 16:49:45	Заграва В.Л.	6232481	Гаврилов Г.О.

Получить информацию

## ІНСТРУКЦІЯ КОРИСТУВАЧА

























### Призначення програми



















































Програма призначена для користування у реєстратурі поліклініки. Для ведення обліку медичних книжок пацієнтів, лікарів та журналу реєстратури. Та для швидкого і зручного доступу до інформації про них.

### Вимоги до системи

Програма призначена для користування на платформі Windows.

Файли, з яких складається програма:

 debug	✓	12.12.2018 19:25	Папка с файлами	45.1 МБ
 doctors	✓	17.12.2018 19:06	Папка с файлами	
 img	✓	12.12.2018 16:45	Папка с файлами	1.02 МБ
 patients	✓	17.12.2018 19:24	Папка с файлами	
 release	✓	10.12.2018 16:58	Папка с файлами	
 .qmake.stash	✓	10.12.2018 16:58	Файл "STASH"	2 КБ
 CourceWork_resource.rc	✓	12.12.2018 15:24	Сценарий ресурс...	1 КБ
 Makefile	✓	12.12.2018 18:47	Файл	29 КБ
 Makefile.Debug	✓	12.12.2018 18:47	Файл "DEBUG"	208 КБ
 Makefile.Release	✓	12.12.2018 18:47	Файл "RELEASE"	209 КБ
 registry.txt	✓	17.12.2018 20:01	Текстовый докум...	1 КБ
 resources.qrc	✓	12.12.2018 02:33	Файл "QRC"	1 КБ
 ui_addnewdisease.h	✓	12.12.2018 18:47	Заголовок C/C++	7 КБ
 ui_addnewdoctor.h	✓	12.12.2018 18:47	Заголовок C/C++	8 КБ
 ui_addnewpatient.h	✓	12.12.2018 18:47	Заголовок C/C++	8 КБ
 ui_creatorinformation.h	✓	12.12.2018 18:47	Заголовок C/C++	6 КБ
 ui_mainwindow.h	✓	12.12.2018 18:47	Заголовок C/C++	11 КБ
 ui_redactdoctorinformation.h	✓	12.12.2018 18:47	Заголовок C/C++	9 КБ
 ui_redactpatientinformation.h	✓	12.12.2018 18:47	Заголовок C/C++	9 КБ
 ui_registryaddnote.h	✓	12.12.2018 19:02	Заголовок C/C++	6 КБ
 ui_registrydoctorshow.h	✓	12.12.2018 19:02	Заголовок C/C++	5 КБ
 ui_registryshow.h	✓	12.12.2018 19:02	Заголовок C/C++	3 КБ
 ui_showdoctorinformation.h	✓	12.12.2018 18:47	Заголовок C/C++	10 КБ
 ui_showpatientinformation.h	✓	12.12.2018 18:47	Заголовок C/C++	10 КБ

 addnewdisease.obj	✓	12.12.2018 18:47	Объектный файл	1 345 КБ
 addnewdoctor.obj	✓	12.12.2018 18:47	Объектный файл	975 КБ
 addnewpatient.obj	✓	12.12.2018 18:47	Объектный файл	1 370 КБ
 CourceWork.exe	✓	15.12.2018 16:49	Приложение	1 985 КБ
 CourceWork.ilc	✓	15.12.2018 16:49	Добавочный фай...	7 026 КБ
 CourceWork.pdb	✓	15.12.2018 16:49	База данных отла...	6 164 КБ
 CourceWork.vc.pdb	✓	15.12.2018 16:49	База данных отла...	3 372 КБ
 CourceWork_resource.res	✓	12.12.2018 18:47	Сценарий скомп...	1 КБ
 creatorinformation.obj	✓	12.12.2018 18:47	Объектный файл	200 КБ
 disease.obj	✓	12.12.2018 18:47	Объектный файл	29 КБ
 DLLCollector.exe	✓	01.12.2014 09:49	Приложение	7 971 КБ
 doctor.obj	✓	12.12.2018 18:47	Объектный файл	34 КБ
 main.obj	✓	12.12.2018 18:47	Объектный файл	79 КБ
 mainwindow.obj	✓	12.12.2018 18:47	Объектный файл	370 КБ
 moc_addnewdisease.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_addnewdisease.obj	✓	12.12.2018 18:47	Объектный файл	74 КБ
 moc_addnewdoctor.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_addnewdoctor.obj	✓	12.12.2018 18:47	Объектный файл	74 КБ
 moc_addnewpatient.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_addnewpatient.obj	✓	12.12.2018 18:47	Объектный файл	74 КБ
 moc_creatorinformation.cpp	✓	12.12.2018 18:47	C++ Source file	3 КБ
 moc_creatorinformation.obj	✓	12.12.2018 18:47	Объектный файл	74 КБ
 moc_mainwindow.cpp	✓	12.12.2018 18:47	C++ Source file	6 КБ
 moc_mainwindow.obj	✓	12.12.2018 18:47	Объектный файл	78 КБ
 moc_predefs.h	✓	12.12.2018 18:47	Заголовок C/C++	1 КБ
 moc_redactdoctorinformation.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_redactdoctorinformation.obj	✓	12.12.2018 18:47	Объектный файл	75 КБ
 moc_redactpatientinformation.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_redactpatientinformation.obj	✓	12.12.2018 18:47	Объектный файл	75 КБ
 moc_registryaddnote.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_registryaddnote.obj	✓	12.12.2018 18:47	Объектный файл	74 КБ
 moc_registrydoctorshow.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_registrydoctorshow.obj	✓	12.12.2018 18:47	Объектный файл	75 КБ
 moc_registryshow.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_registryshow.obj	✓	12.12.2018 18:47	Объектный файл	74 КБ
 moc_showdoctorinformation.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_showdoctorinformation.obj	✓	12.12.2018 18:47	Объектный файл	75 КБ
 moc_showpatientinformation.cpp	✓	12.12.2018 18:47	C++ Source file	4 КБ
 moc_showpatientinformation.obj	✓	12.12.2018 18:47	Объектный файл	75 КБ
 patient.obj	✓	12.12.2018 18:47	Объектный файл	77 КБ
 people.obj	✓	12.12.2018 18:47	Объектный файл	29 КБ
 qrc_resource.cpp	✓	12.12.2018 18:47	C++ Source file	5 205 КБ
 qrc_resource.obj	✓	12.12.2018 18:47	Объектный файл	1 013 КБ
 redactdoctorinformation.obj	✓	12.12.2018 18:47	Объектный файл	1 016 КБ
 redactpatientinformation.obj	✓	12.12.2018 18:47	Объектный файл	1 389 КБ
 registry.obj	✓	15.12.2018 16:49	Объектный файл	90 КБ
 registryaddnote.obj	✓	15.12.2018 16:49	Объектный файл	715 КБ
 registrydoctorshow.obj	✓	15.12.2018 16:49	Объектный файл	1 170 КБ
 registryshow.obj	✓	15.12.2018 16:49	Объектный файл	1 111 КБ
 settings.ini	✓	12.12.2018 19:25	Параметры конф...	1 КБ

## Інструкція з використання та опис інтерфейсу користувача

З моменту запуску програми перед нами відкривається діалогове вікно, в якому ми маємо обрати суб'єкт взаємодії:

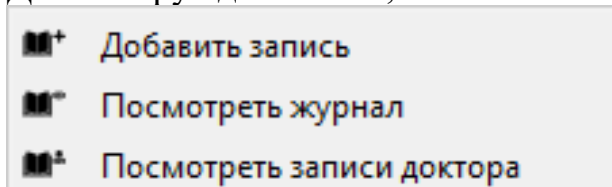
- Реєстратура
- Пацієнт
- Лікар
- Справка



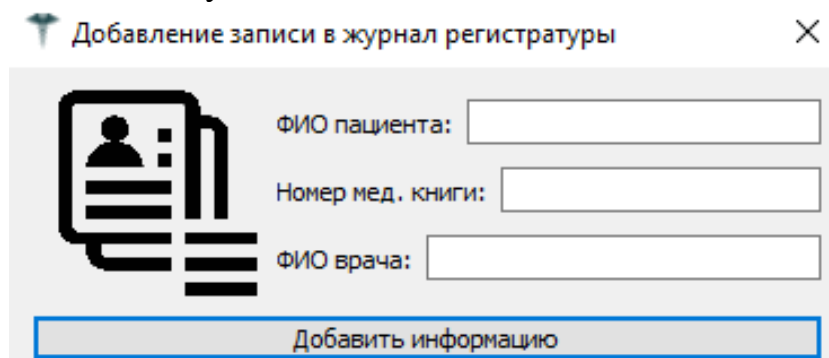
Якщо ми обрали суб'єкт реєстратури, то можемо виконати наступні дії:

- Додати запис до журналу реєстратури
- Переглянути журнал реєстратури
- Переглянути записи конкретного лікаря

Для вибору однієї з них, маємо натиснути на одну з наступних дій:



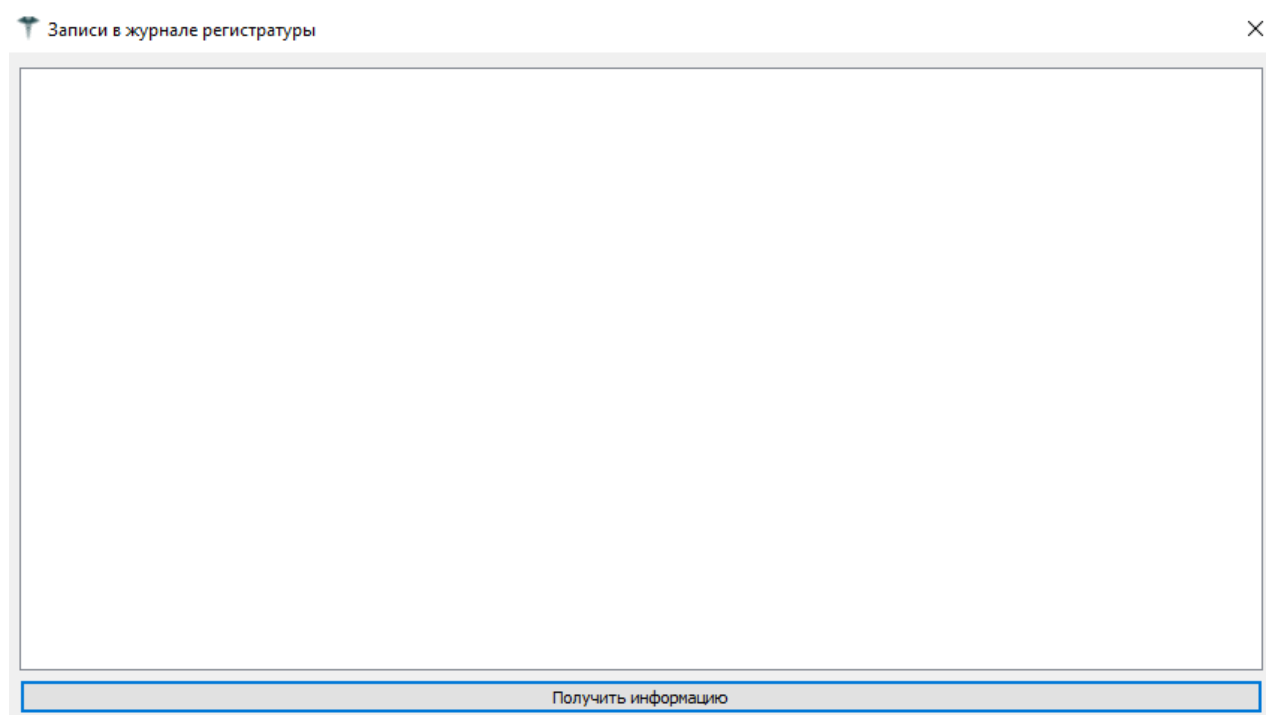
Якщо ми хочемо додати запис до журналу реєстратури, обираємо цю дію і бачимо наступне діалогове вікно:



Діалогове вікно з заголовком «Добавление записи в журнал регистратуры» та кнопкою закриття «X». У вікні є іконка папки з документами. Праворуч від іконки розташовані три текстові поля для введення даних: «ФИО пациента:», «Номер мед. книги:» та «ФИО врача:». У нижній частині вікна розташована кнопка «Добавить информацию».

Ми повинні заповнити всі пусті поля відповідною інформацією та натиснути кнопку «Добавить информацию».

Для того, щоб переглянути весь журнал реєстратури, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Діалогове вікно з заголовком «Записи в журнале регистратуры» та кнопкою закриття «X». Основна частина вікна є порожнім прямокутником, призначеним для виводу списку записів. У нижній частині вікна розташована кнопка «Получить информацию».

Для перегляду журналу реєстратури потрібно натиснути на кнопку «Получить информацию».



Для того, щоб переглянути записи конкретного лікаря в журналі реєстратури, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:

Записи выбранного доктора в журнале регистратуры

Введите ФИО доктора:





Получить информацию

Для перегляду потрібно ввести ПІБ лікаря та натиснути на кнопку «Получить информацию».

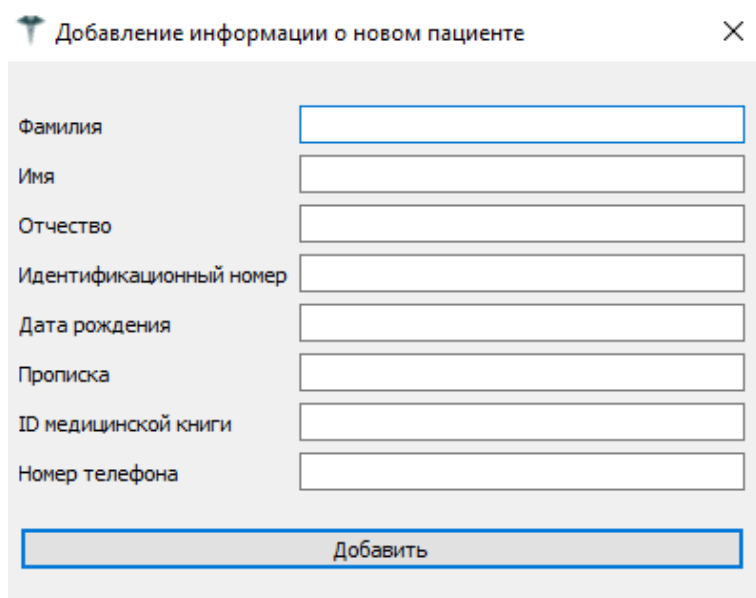
Якщо ми обрали суб'єкт пацієнта, то можемо виконати наступні дії:

- Додати інформацію про нового пацієнта
- Редагувати інформацію про пацієнта
- Переглянути інформацію про пацієнта
- Додати інформацію про хворобу

Для вибору однієї з них, маємо натиснути на одну з наступних дій:

-  Додати інформацію о новом пациенте
-  Редактировать информацию о пациенте
-  Посмотреть информацию о пациенте
-  Добавить информацию о болезни

Для того, щоб додати інформацію про нового пацієнта, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Добавление информации о новом пациенте

Фамилия

Имя

Отчество

Идентификационный номер

Дата рождения

Прописка

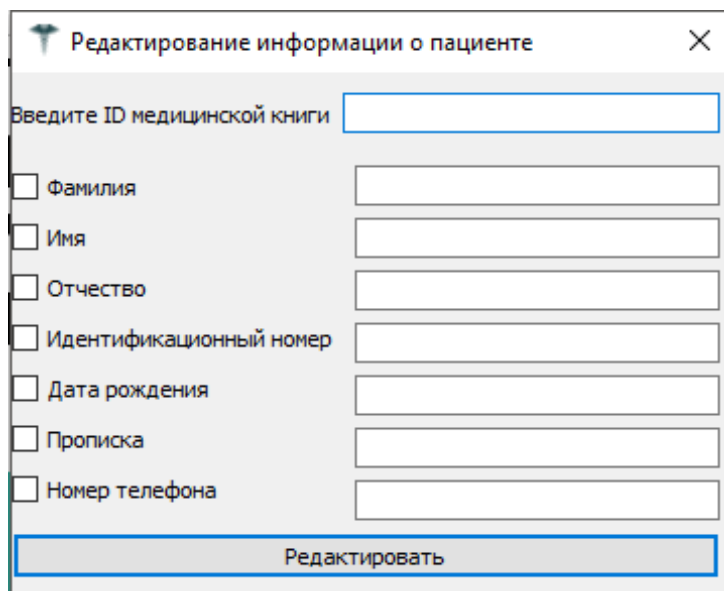
ID медицинской книги

Номер телефона

Добавить

Ми повинні заповнити всі пусті поля відповідною інформацією та натиснути кнопку «Добавить».

Для того, щоб редагувати інформацію про пацієнта, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Редактирование информации о пациенте

Введите ID медицинской книги

☐ Фамилия

☐ Имя

☐ Отчество

☐ Идентификационный номер

☐ Дата рождения

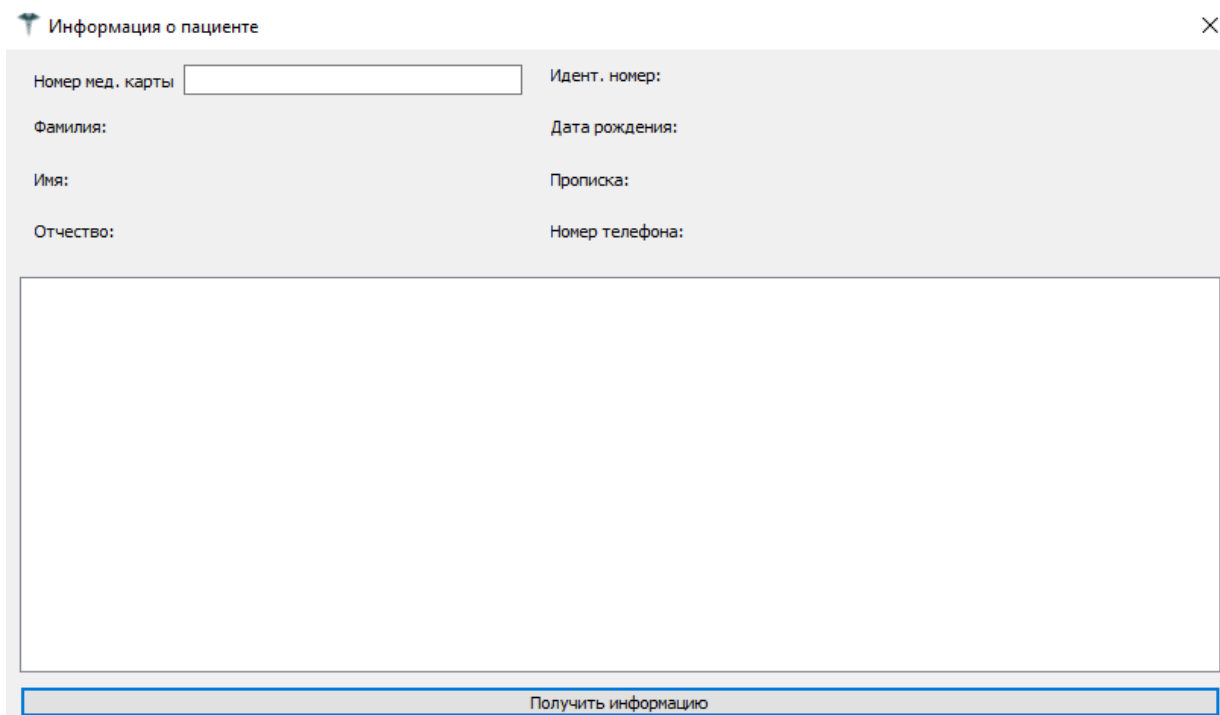
☐ Прописка

☐ Номер телефона

Редактировать

Ми повинні ввести номер медичної книги пацієнта, поставити «галочку» біля інформації, яку хочемо відредагувати та заповнити відповідне пусте поле потрібною нам інформацією і натиснути кнопку «Редактировать».

Для того, щоб переглянути інформацію про пацієнта, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Информация о пациенте

Номер мед. карты  Идент. номер:

Фамилия: Дата рождения:

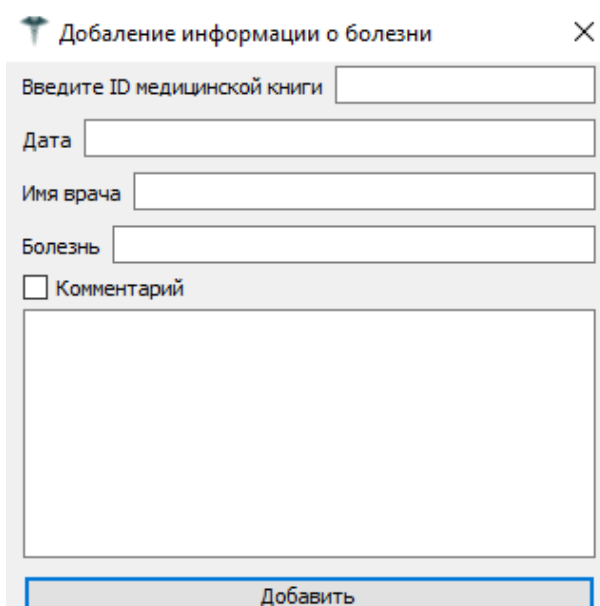
Имя: Прописка:

Отчество: Номер телефона:

Получить информацию

Ми повинні ввести номер медичної книги пацієнта та натиснути кнопку «Получить інформацію».

Для того, щоб додати інформацію про хворобу, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Добавление информации о болезни

Введите ID медицинской книги

Дата

Имя врача

Болезнь

☐ Комментарий

Добавить




Ми повинні заповнити всі пусті поля відповідними даними, та, при

необхідності, поставити «галочку» до коментаря та натиснути кнопку «Добавить».


Якщо ми обрали суб'єкт лікаря, то можемо виконати наступні дії:

- Додати інформацію про нового лікаря
- Редагувати інформацію про лікаря
- Переглянути інформацію про лікаря

Для вибору однієї з них, маємо натиснути на одну з наступних дій:

-  Додати інформацію о новом докторе
-  Редактировать информацию о докторе
-  Посмотреть информацию о докторе

Для того, щоб додати інформацію про нового лікаря, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:

 Добавление информации о новом докторе ×

Фамилия:

Имя:

Отчество:

Специальность:

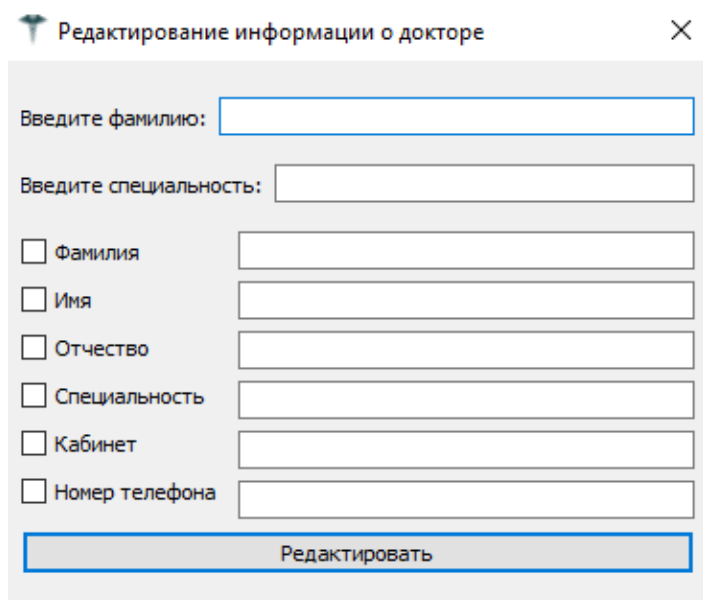
Кабинет:

Номер телефона:

Добавить

Ми повинні заповнити всі пусті поля відповідною інформацією та натиснути кнопку «Добавить».

Для того, щоб редагувати інформацію про лікаря, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Редактирование информации о докторе

Введите фамилию:

Введите специальность:

☐ Фамилия

☐ Имя

☐ Отчество

☐ Специальность

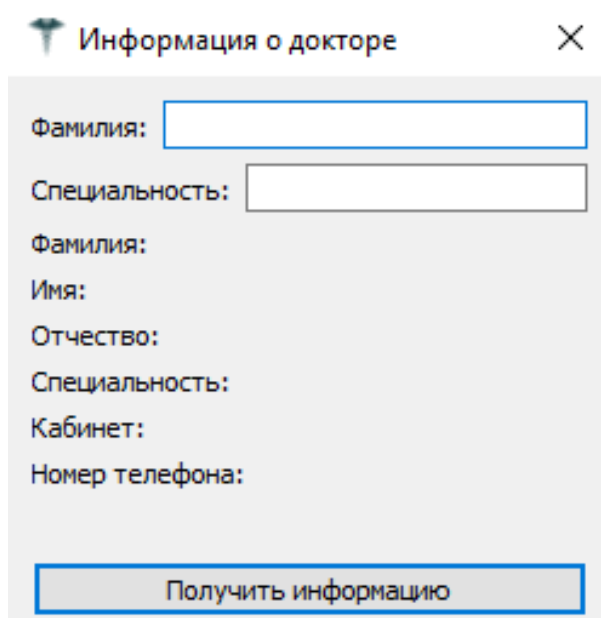
☐ Кабинет

☐ Номер телефона

Редактировать

Ми повинні ввести прізвище лікаря та спеціальність, поставити «галочку» біля інформації, яку хочемо відредагувати та заповнити відповідне пусте поле потрібною нам інформацією і натиснути кнопку «Редактировать».

Для того, щоб переглянути інформацію про лікаря, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



Информация о докторе

Фамилия:

Специальность:

Фамилия:

Имя:

Отчество:

Специальность:

Кабинет:

Номер телефона:

Получить информацию

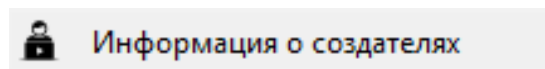
Ми повинні ввести прізвище лікаря та спеціальність та натиснути кнопку

«Получить информацию».

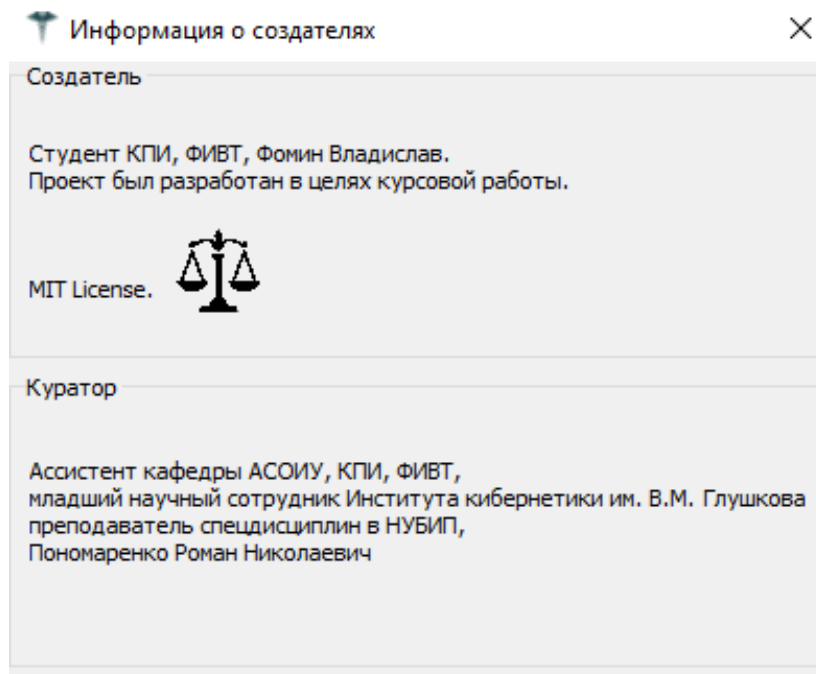
Якщо ми обрали суб'єкт справки, то можемо виконати наступні дії:

- Проглянути інформацію про засновників

Для вибору однієї з них, маємо натиснути на одну з наступних дій:



Для того, щоб переглянути інформацію засновників, ми маємо обрати відповідну дію та далі побачимо наступне діалогове вікно:



## ВИСНОВКИ

Виконуючи даний курсовий проект, у мене не виникло труднощів, адже я слідував чітко розробленому плану.

Спочатку мені довелося глибоко вникнути в задачу: висвітлити сутність задачі, розробити технічне завдання та проаналізувати його. Після виконання цих дій, я вже мав чітке уявлення про мету та кінцевий результат, до якого я прямував.

Аналіз предметної області дала уявлення про взаємодію всіх суб'єктів, наявних в моїй задачі. Узагальненням цього етапу була діаграма прецедентів.

Далі по плану – опис архітектури програмної системи. На цьому етапі я розробив проект моєї задачі, який узагальнив UML-діаграмою класів.

Наступним етапом був опис програмного забезпечення, тобто, опис класів та методів, які будуть використовуватись у моїй програмі.

Маючи все, що потрібно, я розпочав написання програмного коду та графічного інтерфейсу користувача, слідуючи діаграмі та опису класів і аналізу предметної області.

Написавши програмний код, потрібно було протестувати його. Склавши алгоритм тестування, вхідні та очікувані вихідні дані, я пересвідчився у правильності роботи розробленого програмного забезпечення.

Маючи перед очима результат своєї праці, можна сказати, що мета курсової роботи досягнута.

Щодо якісних показників програми, можна засвідчити, що програмний інтерфейс користувача надає максимальну простоту використання та досягнення результату задачі при виконанні мінімальної кількості дій, адже він був ретельно продуманий до найменших аспектів.

Щодо кількісних показників розробленого програмного забезпечення, можна засвідчити, що програма займає 46.6 МБ, що дозволяє встановити її на обчислювальні пристрої, які не мають великої кількості пам'яті. Але вона знадобиться нам для комфортного користування, адже лікарня обслуговує далеко не одного пацієнта, а для зберігання інформації про них пам'ять – невід'ємна частина.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Бьерн Страуструп Программирование. Принципы и практика с использованием C++ / Б. Страуструп. - Лондон, 2016. - [1268] с. - (Информ. листок о науч.-техн. достижении / АрхЦНТИ; N 71-62).
2. Code-Live (інтернет-ресурс). <https://code-live.ru/>
3. Ravesli (інтернет-ресурс). <https://ravesli.com/uroki-cpp/>
4. Cpp Studio (інтернет-ресурс). <http://cppstudio.com/uk/>



## **ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**

Київський політехнічний інститут ім. І. Сікорського  
Кафедра автоматизованих систем обробки інформації та управління

Затвердив

Керівник: Пономаренко Р.М.

« 19 » листопада 2018 р.

Виконавець:

Студент Фомін В.В.

« 19 » листопада 2018 р.

### **ТЕХНІЧНЕ ЗАВДАННЯ**

на виконання курсової роботи

на тему: «Програмна система підтримки обліку  
пацієнтів для реєстратури поліклініки»

з дисципліни:

«Об'єктно-орієнтоване програмування»

Київ 2018

1. *Мета:* Метою курсової роботи є розробка системи підтримки обліку пацієнтів для реєстратури поліклініки
2. *Дата початку роботи:* «01» листопада 2018р.
3. *Дата закінчення роботи:* «»
4. *Вимоги до програмного забезпечення.*

1) Функціональні вимоги:

- Можливість ввести дані про нового пацієнта
- Можливість відредагувати існуючі дані про пацієнта
- Можливість додати інформацію про хворобу пацієнта
- Можливість проглянути інформацію про пацієнта
- Можливість додати інформацію про лікаря
- Можливість відредагувати інформацію про лікаря
- Можливість додати запис у журнал реєстратури
- Можливість проглянути записи у журналі реєстратури

2) Нефункціональні вимоги:

- Можливість запуску програми на платформі Windows
- Безпека особистих даних користувачів
- Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:  
ГОСТ 29.401- 78 - Текст програми. Вимоги до змісту та оформлення.  
ГОСТ 19.106 - 78 - вимоги до програмної документації.  
ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

5. *Стадії та етапи розробки:*

- 1) Об'єктно-орієнтований аналіз предметної області задачі (до 04.12.2018)
- 2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 07.12.2018)
- 3) Розробка програмного забезпечення (до 18.12.2018)
- 4) Тестування розробленої програми (до 21.12.2018)

- 5) Розробка пояснювальної записки (до 23.12.2018)
- 6) Захист курсової роботи (до 31.12.2018)
- 6. *Порядок контролю та приймання.* Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки впливає на оцінку за КР відповідно до критеріїв оцінювання.

## ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду програмного забезпечення

Програмної системи підтримки обліку пацієнтів для реєстратури поліклініки  
(Найменування програми (документа))

CD-RW

(Вид носія даних)

17 арк, 1.22 МБ

(Обсяг програми (документа), арк., Кб)

студента II курсу групи ІІІ-71  
Фоміна Владислава Віталійовича

**disease.h**

```

#include <iostream>
#include <string>
#include <ctime>

using namespace std;

class Disease {
private:
    string date;
    string doctorName;
    string diseaseName;
    string comment;
public:
    void addNewObject(string date, string doctorName, string diseaseName) {
        this->date = date;
        this->doctorName = doctorName;
        this->diseaseName = diseaseName;
        this->comment = " ";
    }
    void addNewObject(string date, string doctorName, string diseaseName, string
comment) {
        this->date = date;
        this->doctorName = doctorName;
        this->diseaseName = diseaseName;
        this->comment = comment;
    }

    void setDate(string date) {
        this->date = date;
    }
    void setDoctorName(string doctorName) {
        this->doctorName = doctorName;
    }
    void setDiseaseName(string diseaseName) {
        this->diseaseName = diseaseName;
    }
    void setComment(string comment) {
        this->comment = comment;
    }

    string getDate() {
        return date;
    }
    string getDoctorName() {
        return doctorName;
    }
    string getDiseaseName() {
        return diseaseName;
    }
    string getComment() {
        return comment;
    }
};

```

## doctor.h

```
#include <iostream>
#include "people.h"
#include <fstream>
#include <string>
#include <cstdlib>
#include <list>
#include <cstdio>
#include "disease.h"

using namespace std;

class Doctor:public People {
private:
    //----Информация о докторе----
    string surname;
    string name;
    string patronymic;
    string speciality;
    string cabinet;
    string phoneNumber;
public:
    Doctor() {} // Конструктор для ввода информации с клавиатуры

    void addInfoKeyboard(string surname, string name, string patronymic,
                        string speciality, string cabinet, string phoneNumber)
    {
        if (surname == "" || name == "" || patronymic == "" || speciality == ""
        || cabinet == "" || phoneNumber == "") {
            throw exception("Вы оставили пустое поле.");
        }
        this->surname = surname;
        this->name = name;
        this->patronymic = patronymic;
        this->speciality = speciality;
        this->cabinet = cabinet;
        this->phoneNumber = phoneNumber;
    }

    void addInfoFile(string filename) {
        filename = "doctors/" + filename + ".txt";
        ifstream file;
        file.open(filename);
        if (!file.is_open()) {
            throw exception("Вы ввели неверную фамилию или специальность!");
        }
        else {
            list<string> tmp;
            string tmpStr;
            while (!file.eof()) {
                file >> tmpStr;
                tmp.push_back(tmpStr);
            }
            file.close();
            auto it = tmp.begin();
            surname = *it;
            it++;
            name = *it;
            it++;
```

```

        patronymic = *it;
        it++;
        speciality = *it;
        it++;
        cabinet = *it;
        it++;
        phoneNumber = *it;
    }
}

void addNewObject() {
    ifstream file;
    string idFile = "doctors/" + this->surname + this->speciality + ".txt";
    file.open(idFile);
    if (!file.is_open()) {
        ofstream file;
        file.open(idFile);

        if (!file.is_open()) {
            throw exception("Error opening file! (addNewObject)");
        }
        else {
            file << this->surname + "\n";
            file << this->name + "\n";
            file << this->patronymic + "\n";
            file << this->speciality + "\n";
            file << this->cabinet + "\n";
            file << this->phoneNumber + "\n";
        }
    }
    else {
        throw exception("Доктор с такой фамилией и специальностью уже существует.");
    }
}

void redactObject(int num, string redactArg) {
    if (redactArg == "") {
        throw exception("Вы оставили пустое выбранное поле.");
    }
    string filename = "doctors/" + surname + speciality + ".txt";
    const char * cFilename = filename.c_str();

    switch (num) {
        case 1: {
            this->surname = redactArg;
            break;
        }
        case 2: {
            this->name = redactArg;
            break;
        }
        case 3: {
            this->patronymic = redactArg;
            break;
        }
        case 4: {
            this->speciality = redactArg;
            break;
        }
    }
}

```

```

        case 5: {
            this->cabinet = redactArg;
            break;
        }
        case 6: {
            this->phoneNumber = redactArg;
            break;
        }
        default: {
            throw exception("Error. Undefined key.");
            break;
        }
    }

    if (num == 1 || num == 4) {
        if (remove(cFilename) != 0) {
            perror("File deletion failed");
            throw exception("File deletion failed!");
        }
        addNewObject();
    }
    else {
        string idFile = "doctors/" + this->surname + this->speciality +
".txt";

        ofstream file;
        file.open(idFile);

        if (!file.is_open()) {
            throw exception("Вы ввели неверную фамилию или специальность!");
        }
        else {
            file << this->surname + "\n";
            file << this->name + "\n";
            file << this->patronymic + "\n";
            file << this->speciality + "\n";
            file << this->cabinet + "\n";
            file << this->phoneNumber + "\n";
        }
    }
}

void printObject() {
    cout << "Surname:" << this->surname << endl <<
        "Name: " << this->name << endl <<
        "Patronymic: " << this->patronymic << endl <<
        "Ident. number: " << this->speciality << endl <<
        "Date of birth: " << this->cabinet << endl <<
        "Phone number: " << this->phoneNumber << endl;
}

void addDisease() {}

void addInfoKeyboard(string surname, string name, string patronymic,
                    string identNum, string dateOfBirth, string
registration,
                    string phoneNumber, string id) { }
void addDisease(string date, string doctorName, string diseaseName) {}
void addDisease(string date, string doctorName, string diseaseName, string
comment) {}
string getSurname() { return surname; }

```



```

string getName() { return name; }
string getPatronymic() { return patronymic; }
string getIdentNum() { return ""; }
string getDateOfBirth() { return ""; }
string getRegistration() { return ""; }
string getPhoneNumber() { return phoneNumber; }
string getSpeciality() { return speciality; }
string getCabinet() { return cabinet; }
int getDiseasesCount() { return 0; }
list<Disease> getDiseases() {
    list <Disease> tmp;
    return tmp;
}
};

```

## patient.h

```
#include <iostream>
#include "people.h"
#include "disease.h"
#include <fstream>
#include <string>
#include <list>
#include <cstdlib>
#include <vector>
#include <QDebug>
#include <QDir>

using namespace std;

class Patient:public People {
private:
    //----Информация о пациенте-----

    string surname;
    string name;
    string patronymic;
    string identNum;
    string dateOfBirth;
    string registration; // Прописка
    string id; // Номер медицинской книги
    string phoneNumber;

    //----Хранение болезней---
    list <Disease> disease;

public:
    void addInfoKeyboard(string surname, string name, string patronymic,
                        string identNum, string dateOfBirth, string
registration,
                        string phoneNumber, string id) {
        if (surname == "" || name == "" || patronymic == "" || identNum == "" ||
            dateOfBirth == "" || registration == "" || phoneNumber == "" ||
id == "") {
            throw exception("Вы оставили пустое поле.");
        }
        this->surname = surname;
        this->name = name;
        this->patronymic = patronymic;
        this->identNum = identNum;
        this->dateOfBirth = dateOfBirth;
        this->registration = registration;
        this->id = id;
        this->phoneNumber = phoneNumber;
    }

    void addInfoFile(string filename) {
        filename = "patients\\" + filename + ".txt";
        ifstream file;
        file.open(filename);
        if (!file.is_open()) {
            throw exception("Вы ввели неверный номер мед. книги пациента!");
            //cout << "Error opening file!\n";
        }
        else {
```

```

list <string> tmp;
string tmpStr;
while (!file.eof()) {
    getline(file, tmpStr, '\n');
    tmp.push_back(tmpStr);
}
file.close();
auto it = tmp.begin();
surname = *it;
it++;
name = *it;
it++;
patronymic = *it;
it++;
identNum = *it;
it++;
dateOfBirth = *it;
it++;
registration = *it;
it++;
phoneNumber = *it;
it++;
id = *it;
it++;

int tmpLen = tmp.size() - 8;
for (int i = 0; i < tmpLen; i++) {
    vector <string> tmpVec;

    string s = *it;
    it++;

    string delimiter = "|";

    size_t pos = 0;
    string token;
    while ((pos = s.find(delimiter)) != string::npos) {
        token = s.substr(0, pos);
        tmpVec.push_back(token);
        s.erase(0, pos + delimiter.length());
    }
    tmpVec.push_back(s);

    Disease tmpDis;
    auto iter = tmpVec.begin();
    tmpDis.setDate(*iter);
    iter++;
    tmpDis.setDoctorName(*iter);
    iter++;
    tmpDis.setDiseaseName(*iter);
    iter++;
    tmpDis.setComment(*iter);

    disease.push_back(tmpDis);
}
}

void addNewObject() {

```

```

ifstream file;
id = this->id;
string idFile = "patients\\" + id + ".txt";
file.open(idFile);
if (!file.is_open()) {
    ofstream file;
    file.open(idFile);

    if (!file.is_open()) {
        throw exception("Error opening file (Patient::addNewObject)");
        //qDebug() << "Error opening file (Patient::addNewObject)\n";
    }
    else {
        file << this->surname + "\n";
        file << this->name + "\n";
        file << this->patronymic + "\n";
        file << this->identNum + "\n";
        file << this->dateOfBirth + "\n";
        file << this->registration + "\n";
        file << this->phoneNumber + "\n";
        file << this->id; // + "\n";  !!!!!!!!!!!!!
    }
}
else {
    throw exception("Пациент с таким номером мед. книги уже
зарегистрирован!");
    //qDebug() << "The patient is already registered.\n";
}
}

void addDisease(string date, string doctorName, string diseaseName) {
    if (date == "" || doctorName == "" || diseaseName == "") {
        throw exception("Вы оставили пустое поле.");
    }
    ofstream file;
    id = "patients\\" + id + ".txt";
    file.open(id, ofstream::app);

    if (!file.is_open()) {
        throw exception("Вы ввели неверный номер мед. книги пациента!");
        //qDebug() << "Error opening file!!!!\n";
    }
    else {
        Disease newDisease;
        newDisease.addNewObject(date, doctorName, diseaseName);
        file << "\n" + newDisease.getDate() + "|" +
newDisease.getDoctorName() +
        "|" + newDisease.getDiseaseName() + "|" +
newDisease.getComment();
    }
}

void addDisease(string date, string doctorName, string diseaseName, string
comment) {
    if (date == "" || doctorName == "" || diseaseName == "") {
        throw exception("Вы оставили пустое поле.");
    }
    ofstream file;
    id = "patients\\" + id + ".txt";
    file.open(id, ofstream::app);

```

```

    if (!file.is_open()) {
        throw exception("Вы ввели неверный номер мед. книги пациента!");
        //qDebug() << "Error opening file!!!!\n";
    }
    else {
        Disease newDisease;
        newDisease.addObject(date, doctorName, diseaseName, comment);
        file << "\n" + newDisease.getDate() + "|" +
newDisease.getDoctorName() +
        "|" + newDisease.getDiseaseName() + "|" +
newDisease.getComment();
    }
}

void printObject() {
    cout << "Surname:" << this->surname << endl <<
        "Name: " << this->name << endl <<
        "Patronymic: " << this->patronymic << endl <<
        "Ident. number: " << this->identNum << endl <<
        "Date of birth: " << this->dateOfBirth << endl <<
        "Registration: " << this->registration << endl <<
        "Phone number: " << this->phoneNumber << endl <<
        "ID: " << this->id << endl;

    auto it = disease.begin();
    Disease tmp;
    for (int i = 0; i < disease.size(); i++) {
        tmp = *it;
        cout << tmp.getDate() << " | " << tmp.getDoctorName() << " | " <<
tmp.getDiseaseName()
        << " | " << tmp.getComment() << endl;
        it++;
    }
}

void redactObject(int num, string redactArg) {
    if (redactArg == "") {
        throw exception("Вы оставили пустое выбранное поле.");
    }
    switch (num) {
        case 1: {
            this->surname = redactArg;
            break;
        }
        case 2: {
            this->name = redactArg;
            break;
        }
        case 3: {
            this->patronymic = redactArg;
            break;
        }
        case 4: {
            this->identNum = redactArg;
            break;
        }
        case 5: {
            this->dateOfBirth = redactArg;
            break;
        }
    }
}

```

```

    }
    case 6: {
        this->registration = redactArg;
        break;
    }
    case 7: {
        this->phoneNumber = redactArg;
        break;
    }
    default: {
        throw exception("Undefined key! (Patient::redactObject)");
        //qDebug() << "Error. Undefined key.\n";
        break;
    }
}

string idFile = "patients\\" + id + ".txt";
ofstream file;
file.open(idFile);

if (!file.is_open()) {
    throw exception("Вы ввели неверный номер мед. книги пациента!");
    //qDebug() << "Error opening file!!!!\n";
}
else {
    file << this->surname + "\n";
    file << this->name + "\n";
    file << this->patronymic + "\n";
    file << this->identNum + "\n";
    file << this->dateOfBirth + "\n";
    file << this->registration + "\n";
    file << this->phoneNumber + "\n";
    file << this->id; // + "\n"; !!!!!!!!!!!!!
}

auto it = disease.begin();
for (int i = 0; i < disease.size(); i++) {
    Disease tmp = *it;
    file << "\n" + tmp.getDate() << "|" << tmp.getDoctorName() << "|" <<
tmp.getDiseaseName()
        << "|" << tmp.getComment();
    it++;
}
}

string getSurname() {
    return surname;
}
string getName() {
    return name;
}
string getPatronymic() {
    return patronymic;
}
string getIdentNum() {
    return identNum;
}
string getDateOfBirth() {
    return dateOfBirth;
}

```

```

string getRegistration() {
    return registration;
}
string getPhoneNumber() {
    return phoneNumber;
}
int getDiseasesCount() {
    return disease.size();
}
list<Disease> getDiseases() {
    return disease;
}

virtual void addInfoKeyboard(string surname, string name, string patronymic,
                             string speciality, string cabinet, string
phoneNumber) { }
    string getSpeciality() { return ""; }
    string getCabinet() { return ""; }
};

```

## people.h

```
#include <iostream>
#include <string>
#include <list>
#include "disease.h"

using namespace std;

class People {
public:
    People() {}
    virtual ~People() {}
    virtual void addInfoKeyboard(string surname, string name, string patronymic,
                                string identNum, string dateOfBirth, string
registration,
                                string phoneNumber, string id) = 0;
    virtual void addInfoFile(string filename) = 0;
    virtual void addNewObject() = 0;
    virtual void redactObject(int num, string redactArg) = 0;
    virtual void printObject() = 0;
    virtual void addDisease(string date, string doctorName, string diseaseName)
= 0;
    virtual void addDisease(string date, string doctorName, string diseaseName,
string comment) = 0;
    virtual string getSurname() = 0;
    virtual string getName() = 0;
    virtual string getPatronymic() = 0;
    virtual string getIdentNum() = 0;
    virtual string getDateOfBirth() = 0;
    virtual string getRegistration() = 0;
    virtual string getPhoneNumber() = 0;
    virtual int getDiseasesCount() = 0;
    virtual list<Disease> getDiseases() = 0;
    virtual void addInfoKeyboard(string surname, string name, string patronymic,
                                string speciality, string cabinet, string
phoneNumber) = 0;
    virtual string getSpeciality() = 0;
    virtual string getCabinet() = 0;
};
```



## registry.h

```

#include <iostream>
#include <string>
#include <list>
#include <fstream>
#include <vector>
#include <ctime>
#include <QDebug>
#include <Windows.h>

using namespace std;

struct Note {
private:
    string date; // Дата записи
    string patientName; // Фамилия и инициалы
    string bookNumber; // номер книжки пациента
    string doctorName; // Фамилия и инициалы
public:
    void setDate(string date) {
        this->date = date;
    }
    void setPatientName(string patientName) {
        this->patientName = patientName;
    }
    void setBookNumber(string bookNumber) {
        this->bookNumber = bookNumber;
    }
    void setDoctorName(string doctorName) {
        this->doctorName = doctorName;
    }
    string getDate() {
        return date;
    }
    string getPatientName() {
        return patientName;
    }
    string getBookNumber() {
        return bookNumber;
    }
    string getDoctorName() {
        return doctorName;
    }
};

class Registry {
private:
    list <Note> notes;
    string fileName = "registry.txt";
    list <Note> doctorNotesList; // список искомых записей определённого доктора

    Registry() = default;
    ~Registry() = default;

    Registry(const Registry&) = delete;
    Registry& operator=(const Registry&) = delete;

    void* operator new(std::size_t) = delete;
    void* operator new[](std::size_t) = delete;

```

```

void operator delete(void*) = delete;
void operator delete[](void*) = delete;
public:
    static Registry& initialize() {
        static Registry obj;
        return obj;
    }

    Registry& addNote(string patientName, string idMedBook, string doctorName) {
        if (patientName == "" || idMedBook == "" || doctorName == "") {
            throw exception("Вы оставили пустое поле.");
        }
        Note tmp;
        string tmpS;

        string date = __DATE__;
        date += " ";
        date += __TIME__;

        tmp.setDate(date);
        tmp.setPatientName(patientName);
        tmp.setBookNumber(idMedBook);
        tmp.setDoctorName(doctorName);

        ifstream fileTest;
        fileTest.open("patients\\" + idMedBook + ".txt");
        if (!fileTest.is_open()) {
            throw exception("Вы ввели неверный номер мед. книги пациента!");
        }

        ofstream file;
        file.open(fileName, ofstream::app);
        if (!file.is_open()) {
            throw exception("Ошибка открытия файла регистратуры.");
        }
        else {
            file << tmp.getDate() + "|" + tmp.getPatientName() + "|" +
                tmp.getBookNumber() + "|" + tmp.getDoctorName() + "\n";
        }
        return *this;
    }

    Registry& readNoteList() {
        ifstream file;
        file.open(fileName);
        if (!file.is_open()) {
            throw exception("Ошибка открытия файла регистратуры.");
        }
        else {
            notes.clear();
            list<string> tmp;
            string tmpStr;
            while (!file.eof()) {
                getline(file, tmpStr, '\n');
                tmp.push_back(tmpStr);
            }
            file.close();
            auto it = tmp.begin();

```

```

        for (int i = 0; i < tmp.size()-1; i++) { // -1 пото му что в файле
последняя строка пустая
            vector <string> tmpVec;

            string s = *it;
            it++;

            string delimiter = "|";

            size_t pos = 0;
            string token;
            while ((pos = s.find(delimiter)) != string::npos) {
                token = s.substr(0, pos);
                tmpVec.push_back(token);
                s.erase(0, pos + delimiter.length());
            }
            tmpVec.push_back(s);

            Note tmpNote;
            auto iter = tmpVec.begin();
            tmpNote.setDate(*iter);
            iter++;
            tmpNote.setPatientName(*iter);
            iter++;
            tmpNote.setBookNumber(*iter);
            iter++;
            tmpNote.setDoctorName(*iter);

            notes.push_back(tmpNote);
        }
        tmp.clear();
    }
    return *this;
}

void printNoteList() {
    auto it = notes.begin();
    Note tmp;
    for (int i = 0; i < notes.size(); i++) {
        tmp = *it;
        cout << tmp.getDate() << " | " << tmp.getPatientName() << " | " <<
            tmp.getBookNumber() << " | " << tmp.getDoctorName() << endl;
        it++;
    }
}

Registry& search(string doctorName) {
    if (doctorName == "") {
        throw exception("Вы оставили пустое поле.");
    }
    doctorNotesList.clear();
    auto it = notes.begin();
    Note tmp;
    for (int i = 0; i < notes.size(); i++) {
        tmp = *it;
        if (tmp.getDoctorName() == doctorName) {
            doctorNotesList.push_back(tmp);
        }
        it++;
    }
}

```

```

        return *this;
    }

    void printDoctorNotesList() {
        auto it = doctorNotesList.begin();
        Note tmp;
        for (int i = 0; i < doctorNotesList.size(); i++) {
            tmp = *it;
            cout << tmp.getDate() << " | " << tmp.getPatientName() << " | " <<
                tmp.getBookNumber() << " | " << tmp.getDoctorName() << endl;
            it++;
        }
    }

    list <Note> getNotes() {
        return notes;
    }

    list <Note> getDoctorNotes() {
        return doctorNotesList;
    }
};

```