

ТИТУЛЬНИЙ

Зміст

Вступ.....	3
1 ЗАГАЛЬНИЙ РОЗДІЛ	4
1.1 Аналіз предметної області	4
1.1.1 Опис бізнес-процесів в предметній області	4
1.1.2 Вихідні дані.....	8
1.1.3 Результати	9
1.1.4. Зразки документів, облік яких автоматизується.....	9
1.2 Постановка задачі	10
1.2.1. Цілі і призначення системи.....	10
1.2.2. Функції системи	12
2 СПЕЦІАЛЬНИЙ РОЗДІЛ	14
2.1 Розробка внутрішніх структур даних	14
2.2 Проектування структури програми і взаємодії модулів	17
2.2.1 Специфікація підзадач і способів їх взаємодії.....	18
2.2.2 Узагальнений алгоритм логічної структури програми	20
2.3 Опис засобів програмування.....	20
2.3.1 Структурне програмування.....	20
2.3.2 Мова програмування C++ та інструментальне середовище програмування	21
2.4 Опис програми	24
3. ЗАКЛЮЧНА ЧАСТИНА.....	27
3.1. Тестування програми і аналіз отриманих результатів	27
3.2. Інструкція користувача	30
Висновки	35
Список використаної літератури	36
Додаток А. Код програми.....	37

Вступ

У сучасному світі, де ключовими принципами є ефективність, швидкість та якість, відсутність часу має критичне значення. Отримати бажаний товар або послугу з максимально ефективним використанням свого часу та зусиль, стає все більш актуальним запитом людини. У відповідь на це попит, на ринку послуг з'являються автоматизовані пошукові системи, які забезпечують максимальну швидкість та точність знаходження необхідної інформації.

Однією з таких систем може стати інформаційно-пошукова система "Оренда катерів", яка дозволяє допомогти користувачам знайти найбільш підходяще рішення при пошуку катерів для оренди.

Залишаючись у порозумінні зі стрімким розвитком цієї галузі, ця курсова робота проводитиме аналіз предметної області та надасть постановку задачі описуючі технічні вимоги до розробки програмного забезпечення.

Мета цього проекту – створити зручну та легку в управлінні платформу для пошуку та бронювання катерів у зручному режимі для користувачів.

Для досягнення цієї мети потрібно скласти детальний огляд ринку оренди катерів і виявити ключові процеси, що потребують автоматизації, щоб дати користувачам можливість легко знайти необхідний катер та здійснити бронювання. Також ми маємо розробити вимоги до інформаційної системи та вибрати найкращі програмні засоби для реалізації проекту. Основним завданням буде створення зручного та ефективного інтерфейсу, що надасть користувачам можливість легко та швидко виконувати всі необхідні дії. Такий проект є практично значущим, оскільки прискорить процес пошуку та бронювання катерів, забезпечивши високу ефективність та зручність користування.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналіз предметної області

1.1.1 Опис бізнес-процесів в предметній області

Автоматизована інформаційно-пошукова система "Оренда катерів" є інноваційним рішенням, спрямованим на полегшення процесу оренди катерів для розваг та відпочинку. Вона надає зручність та ефективність у виборі, бронюванні та оплаті послуг оренди катерів.

Таким чином, оренда катерів є комерційно успішним підприємством.

Головними цілями надання послуг в поліклініці є:

1. Пошук катерів для оренди:

- Користувач відкриває автоматизовану інформаційно-пошукову систему "Оренда катерів".
- Користувач вводить параметри пошуку, такі як ім'я, тривалість оренди та інші вимоги.
- Система аналізує базу даних доступних катерів та проводить пошук відповідно до введених параметрів.
- Результати пошуку відображаються користувачу, включаючи інформацію про доступні катери, їх характеристики та ціни.

2. Бронювання катерів на потрібну дату та час:

- Користувач обирає певний катер з результатів пошуку та вказує дату та час оренди.
- Система перевіряє доступність катера на вказаний період та відображає користувачу інформацію про можливість бронювання.
- Користувач підтверджує бронювання катера та вводить необхідну особисту інформацію, таку як ім'я, контактні дані і можливо оплату передоплати.
- Система резервує катер на вказаний період та надсилає підтвердження бронювання користувачеві з усією необхідною інформацією.

3. Оплата послуги оренди катерів:

- Користувач отримує підтвердження бронювання з інформацією про вартість оренди та способи оплати.
- Користувач вносить необхідну суму грошей або здійснює оплату онлайн за допомогою вказаних способів оплати.
- Система підтверджує оплату та надсилає користувачеві підтвердження оплати.
- Катер стає доступним для користувача в зазначений день та час.

Ці бізнес-процеси можуть бути реалізовані за допомогою автоматизованої інформаційно-пошукової системи "Оренда катерів", що дозволяє забезпечити зручну та ефективну організацію оренди катерів, пошук доступних варіантів та здійснення оплати.

Завдяки автоматизованій інформаційно-пошуковій системі "Оренда катерів" процес пошуку, бронювання та оплати послуг оренди катерів стає швидким, зручним та ефективним для користувачів, сприяючи покращенню їхнього відпочинку та задоволенню.

Розглянемо бізнес-процеси роботи поліклініки.

Запис на прийом по телефону.

Пацієнт дзвонить в реєстратуру і записується на прийом до лікаря.

Співробітник реєстратури дивиться графік роботи лікарів, записує пацієнта. Напередодні прийому співробітник реєстратури передає картки записаних на прийом пацієнтів в кабінети лікарів.

Наприклад, в поліклініці ведуть прийом три хірурга, але пацієнт хоче записатися на прийом до конкретного лікаря. У реєстратурі намагаються навантаження лікарів розподіляти рівномірно або до цього лікаря вже закритий прийом на цей день (за кількістю пацієнтів), тому прохання пацієнта не задоволене.

Запис на прийом в реєстратурі.

Пацієнт приходить в поліклініку, звертається в реєстратуру, співробітник реєстратури звіряється із записом заявок і виписує талон до лікаря, звіряючись зі списком кабінетів – ставить на талоні номер кабінету.

Пацієнт приходить в поліклініку, а прийом скасували. Потрібно починати процедуру запису на наступний день, але все талони можуть бути вже зайняті.

Прийом у лікаря.

Лікар оглядає пацієнта, ставить діагноз, робить записи в карті пацієнта. Виписує направлення і призначення. Видає талон на повторний прийом. В кінці робочого дня лікар передає карти пацієнтів в реєстратуру.

Під час прийому лікар сумнівається в діагнозі і хоче проконсультуватися з колегою, але той веде прийом і не може приділити на це час. Пацієнту пропонують почекати або призначають кілька обстежень, іноді абсолютно непотрібних (рисунок 1.1).

Для того, щоб позбутися від недоліків такої організації роботи, необхідно створення автоматизованої інформаційної системи, яка полегшить роботу реєстратури і лікаря, а також збереже час і нерви пацієнтів.

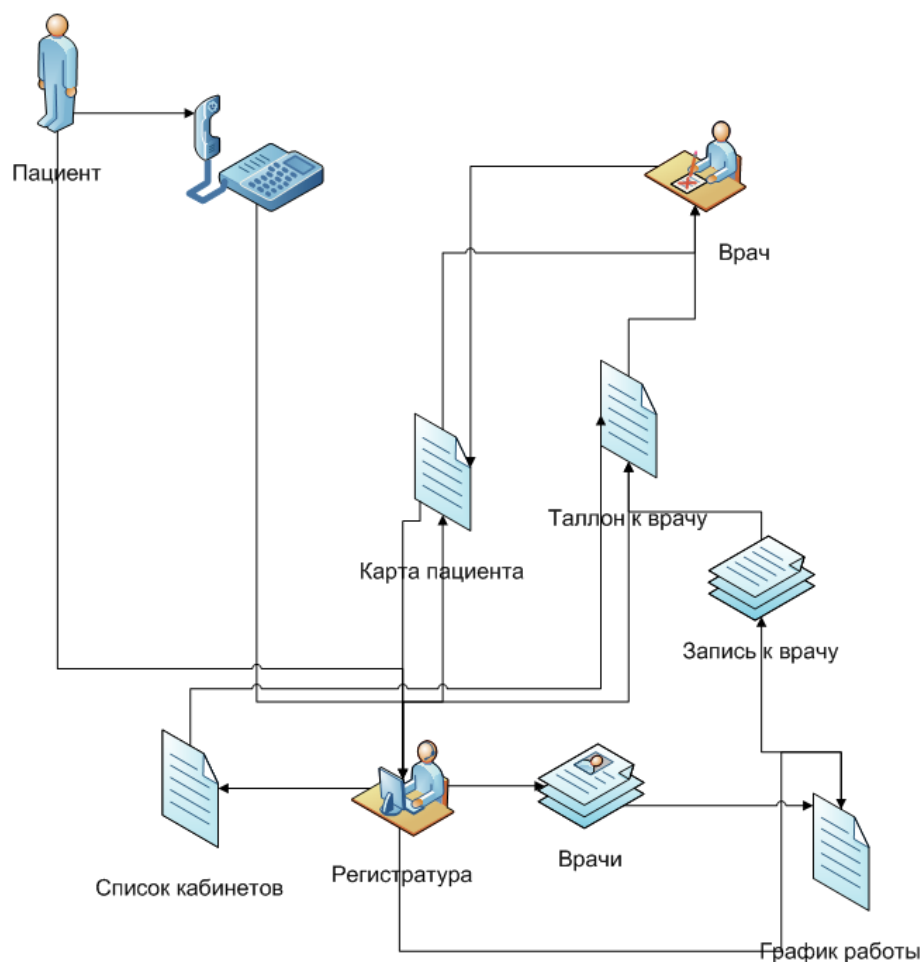


Рисунок 1.1 – Мнемосхема AS-IS, яка відображає принцип роботи поліклініки та її недоліки

Необхідно створити інформаційну систему, яка буде мати базу даних з розподіленим входом в неї в залежності від прав користувача: пацієнт, лікар, працівник реєстратури.

Для пацієнта створюється електронний кабінет на сервері медичного центру, в який пацієнт входить за своїм логіном і паролем. На сервері розміщується інформація про напрями, за якими ведеться прийом в медичному центрі, інформація про лікарів. Також є графік прийому лікарями медичного центру, є електронна форма запису до лікаря і інша довідкова інформація.

Для лікаря на сервері медичного центру створюється доступ до бази даних, де у нього є доступ до електронних карт пацієнтів, ведеться облік захворюваності, диспансеризації і т.д.

Для працівника реєстратури створюється база даних, де зібрані відомості про лікарів, пацієнтів, кабінетах, складається графік прийому, ведеться картка пацієнта або історія хвороби.

Мнемосхема AS-TO-BE пропонованого проектного рішення представлена на рисунку 1.2.

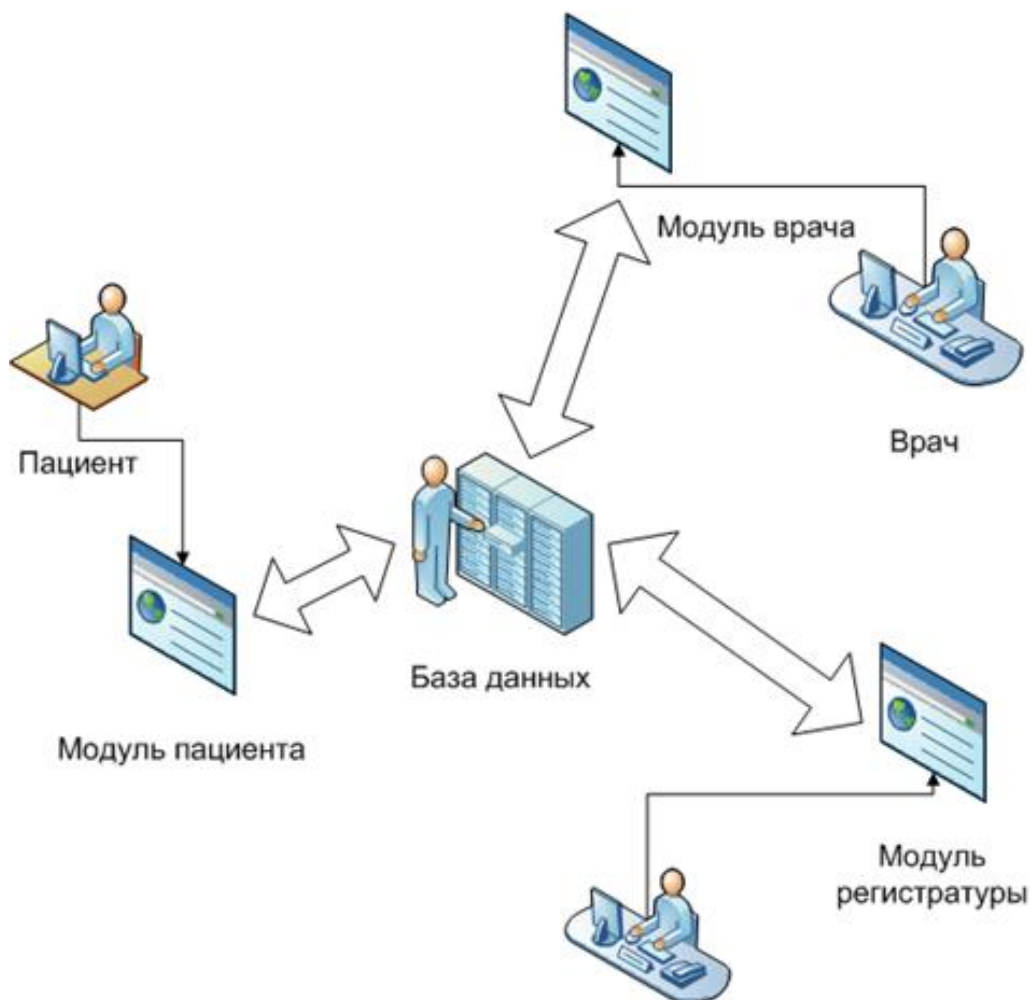


Рисунок 1.2 – Мнемосхема AS-TO-BE пропонованого проектного рішення, яка на концептуальному рівні відображає постановку завдання на створення АІС.

1.1.2 Вихідні дані

До нормативно-довідкової вхідної інформації відносяться довідники хвороб, кабінетів, типів прийому, докторів. Довідкова інформація вноситься відразу після впровадження проекту і в майбутньому редагується в міру надходження нових відомостей.

Вихідними даними є:

- дані лікарів;
- дані пацієнтів;
- дані про хвороби;
- дані про спеціалізацію лікарів;
- дані про прийом пацієнтів.

1.1.3 Результати

Результатною інформацією є:

- виведення інформації про кількість викликів додому у вказаний день;
- виведення інформації про кількість хворих вказаною хворобою;
- визначала назву хвороби, якою хворіють найчастіше.

1.1.4. Зразки документів, облік яких автоматизується

Розглянемо типові документи, облік яких необхідно автоматизувати. Типовий штатний розпис є документом, яким визначається структура закладу і чисельність посад за кожним найменуванням у конкретних підрозділах і в цілому по установі.

Штатний розпис відповідно до діючого порядку затверджується вищою організацією. Це відображається на документі «Типовий штатний розпис», в правому верхньому кутку якого під грифом «Затверджую» ставиться підпис відповідальної посадової особи та дата затвердження.

Типовий штатний розпис складається за встановленою формою всіма закладами охорони здоров'я та затверджується щорічно в строки, визначені для затвердження кошторисів доходів і видатків.

Порядок розташування структурних підрозділів і посад у поліклініці визначається керівником медичного закладу з урахуванням рекомендованої схеми для поліклініки.

Відповідно до наказу Міністерства охорони здоров'я України, у кожному амбулаторному закладі охорони здоров'я на пацієнта заводиться медична карта амбулаторного хворого за формою № 025/о.

На кожного хворого в поліклініці ведеться одна карта незалежно від того, лікується він в одного чи декількох лікарів (рисунок 1.3).

ЗАТВЕРДЖЕНО
 Наказ Міністерства охорони здоров'я України
 14 лютого 2012 року № 110

Найменування міністерства, іншого органу виконавчої влади, підприємства, установи, організації, до сфери управління якого належить заклад охорони здоров'я _____ Найменування та місцезнаходження (повна поштова адреса) закладу охорони здоров'я, де заповнюється форма _____ Код за ЄДРПОУ _____	МЕДИЧНА ДОКУМЕНТАЦІЯ Форма первинної облікової документації № 025/о ЗАТВЕРДЖЕНО Наказ МОЗ України № _____
--	---

МЕДИЧНА КАРТА АМБУЛАТОРНОГО ХВОРОГО № _____

Код хворого _____
Дата заповнення карти _____
(число, місяць, рік)

1. Прізвища, ім'я, по батькові _____
2. Стать: чоловіча – 1, жіноча – 2 ☐ 3. Дата народження _____
(число, місяць, рік) 4. Телефон: дом. _____, робочий _____
5. Місце проживання хворого _____ 6. Місце роботи, посада _____
7. Диспансерна група (так – 1, ні – 2) ☐
8. Контингент: інваліди війни – 1; учасники війни – 2; учасники бойових дій – 3; інваліди – 4; учасники ліквідації наслідків аварії на Чорнобильській АЕС – 5; евакуйовані – 6; особи, які проживають на території зонітрадіоекологічного контролю – 7; діти, які народились від батьків, які віднесені до 1, 2, 3 категорій осіб, що постраждали внаслідок Чорнобильської катастрофи, із зони відчуження, а також віднесені із зони безумовного (обов'язкового) і гарантованого добровільного відселення – 8; інші пільгові категорії – 9 ☐
9. Номер пільгового посвідчення _____
10. Вазитий(а) на облік: _____ з приводу _____
 (число, місяць, рік) (число, місяць, рік)
 _____ з приводу _____
 (число, місяць, рік) (число, місяць, рік)
11. Знятий(а) з обліку _____ (причина) _____
 (число, місяць, рік) (число, місяць, рік)
 _____ (причина) _____
 (число, місяць, рік) (число, місяць, рік)

Рисунок 1.3 – Медична карта амбулаторного хворого

Внесені дані засвідчуються підписом лікаря і печаткою.

1.2 Постановка задачі

1.2.1. Цілі і призначення системи

Базу даних використовує для роботи колектив лікарів. У таблиці мають бути занесена інформація про пацієнта: ім'я, стать, дата народження і домашня адреса. Всякий раз, коли лікар оглядає хворого, який з'явився до нього на прийом, або коли лікар сам приходить до нього додому, він записує дату і місце, де проводиться огляд, симптоми, діагноз.

На кожного *пацієнта* в базу даних заносяться наступні відомості: код пацієнта, стать, ПІБ (Прізвище, Ім'я, По батькові), дата народження, адреса, телефон.

На кожного *лікаря* в базу даних заносяться наступні відомості: код лікаря, стать, ПІБ (Прізвище, Ім'я, По батькові), дата народження, адреса, телефон, спеціалізація, стаж роботи.

Кожний *прийом* характеризується наступними параметрами: код лікаря, код пацієнта, дата прийому, місце прийому (місце проживання пацієнта або лікарня), діагноз.

Існуючий на сьогодні спосіб роботи поліклініки з відвідувачами пов'язаний з великою трудомісткістю і розрізненістю відомостей, що з великою ймовірністю може призвести до втрати важливої інформації або неправильної її інтерпретації.

У разі використання обчислювальної техніки цей процес зводиться до введення даних користувача в інформаційну систему, включаючи дату прийому, а також призначені лікувальні заходи. На підставі цих даних система визначатиме вільного лікаря, буде планувати дату і час прийому, надавати дані про пацієнта лікаря і формувати графік прийому лікаря в електронному вигляді, за запитом готувати звіти [1].

Автоматизація цього процесу дозволить:

- планувати завантаження поліклініки «Довголіття».
- здійснювати планування прийому лікарями і пацієнтами.
- вести облік медичних послуг.
- планувати завантаження лікарів і медичних кабінетів.
- формувати звіти медичної статистики.

В результаті впровадження системи планується досягти:

- збільшення швидкості обслуговування пацієнтів – збільшення потоку виручки;
- оптимізації планування завантаження кабінетів і персоналу – підвищення ефективності використання ресурсів підприємства;
- можливість формування різних видів запланованих звітів;

- оперативний і стратегічний аналіз роботи кабінетів і фахівців;
- надійність зберігання і швидкість обробки даних;
- підвищення ефективності управлінських рішень.

1.2.2. Функції системи

Варіанти використання (прецеденти) допомагають визначити функціональні вимоги до системи [4].

Прецеденти описують взаємодії між користувачами системи та самою системою за допомогою сценаріїв, тобто певних дій.

Користувач у цій нотації називається актором (actor) і визначає роль, що він грає стосовно системи. Розробимо перелік прецедентів для побудови UML-діаграми «Поліклініка» [8].

Таблиця 1.1 – Список акторів та прецедентів

Найменування акторів	Прецедент
Медсестра приймального відділення	Заносить до бази даних дані: про відділення, про пацієнтів, про хвороби, про спеціалізації лікарів, про лікарів
Лікар	Вносить дані історії хвороби пацієнта
	Формує умови для запиту
	Отримує інформацію на запит
	Отримує звіти

В результаті аналізу предметної області було складено наступну діаграму прецедентів (варіантів використання) (рисунок 1.4).

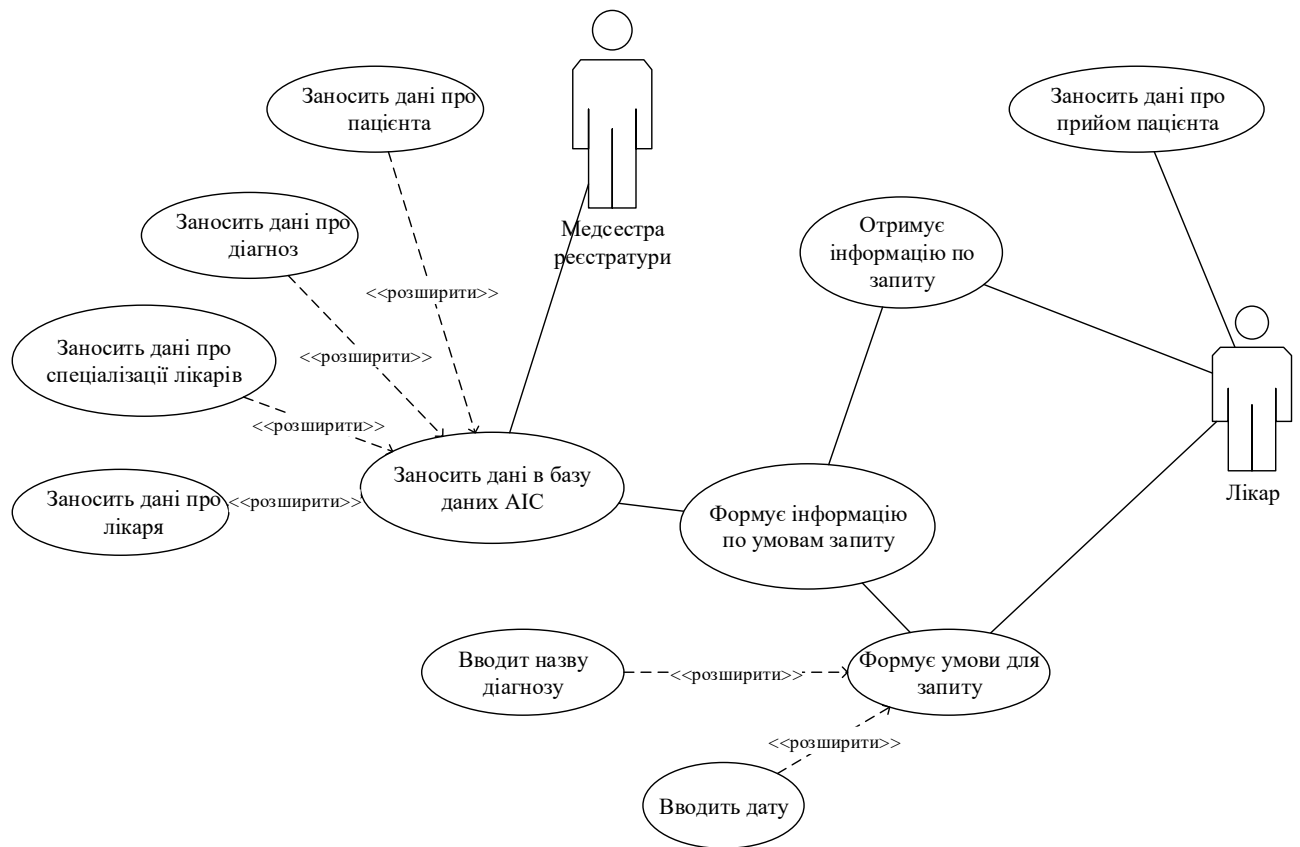


Рисунок 1.4 – Діаграма варіантів використання (прецедентів)

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Розробка внутрішніх структур даних

У розглянутої предметної області «Поліклініка» виділені наступні інформаційні об'єкти – сутності [7] (таблиця 2.1).

Таблиця 2.1 – Сутності предметної області

Сутність	Опис
пацієнт	дані пацієнта
лікар	дані про лікарів
прийом	дані про прийом пацієнта лікарем
діагноз	список хвороб
спеціалізація	дані про спеціалізацію доктора

Зв'язки між сутностями подано в таблиці 2.2.

Таблиця 2.2 – Зв'язки між сутностями

Сутність батьківська	Сутність дочірня	Тип зв'язку
пацієнт	прийом	один-до-багатьох
лікар	прийом	один-до-багатьох
діагноз	прийом	один-до-багатьох
спеціалізація	лікар	один-до-багатьох
спеціалізація	діагноз	один-до-багатьох

Сутності та зв'язки між сутностями предметної області «Поліклініка» представлені на рисунку 2.1.

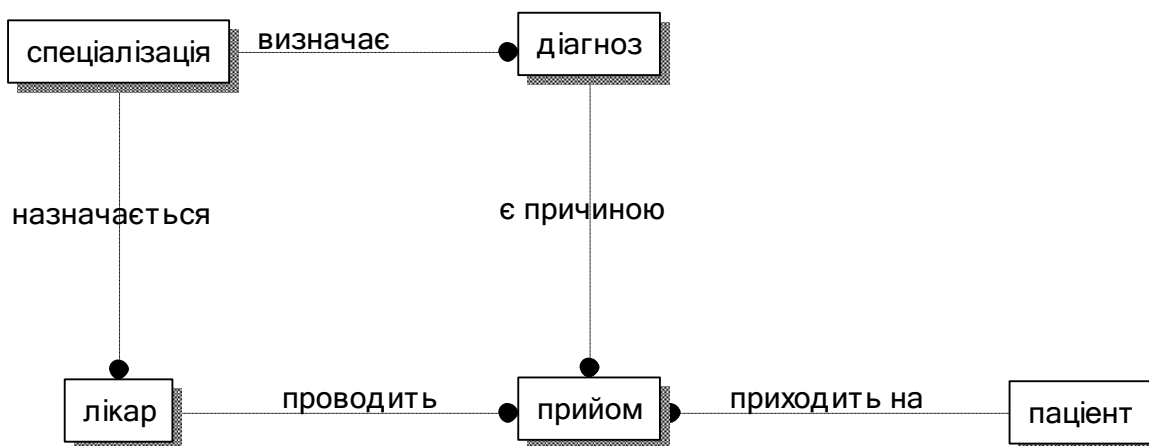


Рисунок 2.1 – Сутності та зв'язки між сутностями

Атрибути сутностей предметної області «Поліклініка» представлені в таблиці 2.3.

Таблиця 2.3 – Атрибути сутностей предметної області «Поліклініка»

Сутність	Атрибути	Ключі
Пацієнт	КодПацієнта	Первинний ключ
	Стать	
	ПІБПацієнта	
	ДатаНародження	
	Адреса	
	Телефон	
Спеціалізація	КодСпеціалізації	Первинний ключ
	Спеціалізація	
Діагноз	КодДіагнозу	Первинний ключ
	Діагноз	
	КодСпеціалізації	Зовнішній ключ
Лікар	КодЛікаря	Первинний ключ
	Стать	
	ПІБЛікаря	
	ДатаНародження	
	Адреса	
	Телефон	
	КодСпеціалізації	Зовнішній ключ
	СтажРоботи	
	НазваТипПрийому	
Прийом	КодПрийому	Первинний ключ
	КодПацієнта	Зовнішній ключ
	КодЛікаря	Зовнішній ключ
	ДатаЗвернення	
	МісцеПрийому	
	Діагноз	Зовнішній ключ

На рисунку 2.2 представлена інфологічна модель предметної області.

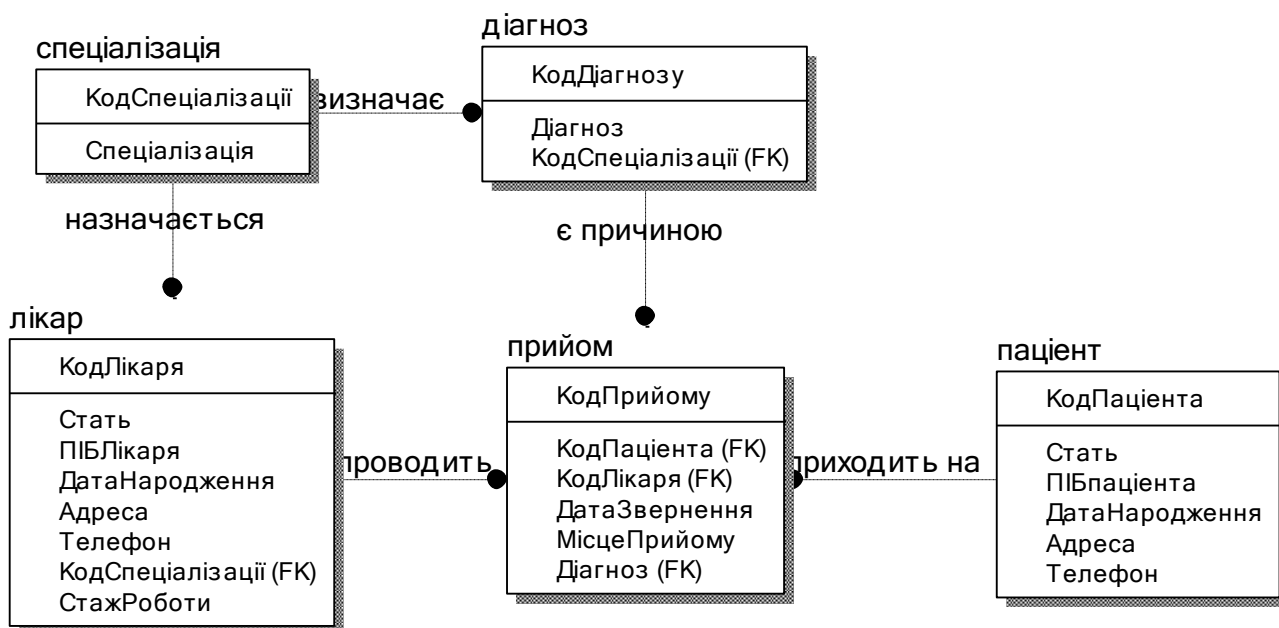


Рисунок 2.2 – Інфологічна модель предметної області «Поліклініка» – FA-діаграма

Далі визначаємо типи даних для фізичної моделі [8]. Ця модель даних містить 5 таблиць, і в даній моделі немає зв'язків «багато-до-багатьох», тому модель даних нормалізована, приведена до 3 НФ (таблиця 2.4-2.8).

Таблиця 2.4 – Структура таблиці «Пацієнт»

Ідентифікатор	Тип	Примітка
КодПацієнта	Лічильник	Первинний ключ
Стать	Текстовий	
ПІБПацієнта	Текстовий	
ДатаНародження	Дата / час	
Адреса	Текстовий	
Телефон	Текстовий	

Таблиця 2.5 – Структура таблиці «Прийом»

Ідентифікатор	Тип	Примітка
КодПрийому	Лічильник	Первинний ключ
КодПацієнта	Числовий	Зовнішній ключ
КодЛікаря	Числовий	Зовнішній ключ
ДатаЗвернення	Дата / час	
МісцеПрийому	Текстовий	
Діагноз	Числовий	Зовнішній ключ

Таблиця 2.6 – Структура таблиці «Діагноз»

Ідентифікатор	Тип	Примітка
КодДіагнозу	Лічильник	Первинний ключ
Діагноз	Текстовий	
КодСпеціалізації	Числовий	Зовнішній ключ

Таблиця 2.7 – Структура таблиці «Лікар»

Ідентифікатор	Тип	Примітка
КодЛікаря	Лічильник	Первинний ключ
Стать	Текстовий	
ПІБЛікаря	Текстовий	
ДатаНародження	Дата / час	
Адреса	Текстовий	
Телефон	Текстовий	
КодСпеціалізації	Числовий	Зовнішній ключ
СтажРоботи	Числовий	

Таблиця 2.8 – Структура таблиці «Спеціалізація»

Ідентифікатор	Тип	Примітка
КодСпеціалізації	Лічильник	Первинний ключ
Спеціалізація	Текстовий	

Діаграма фізичного рівня представлена на рисунку 2.3.

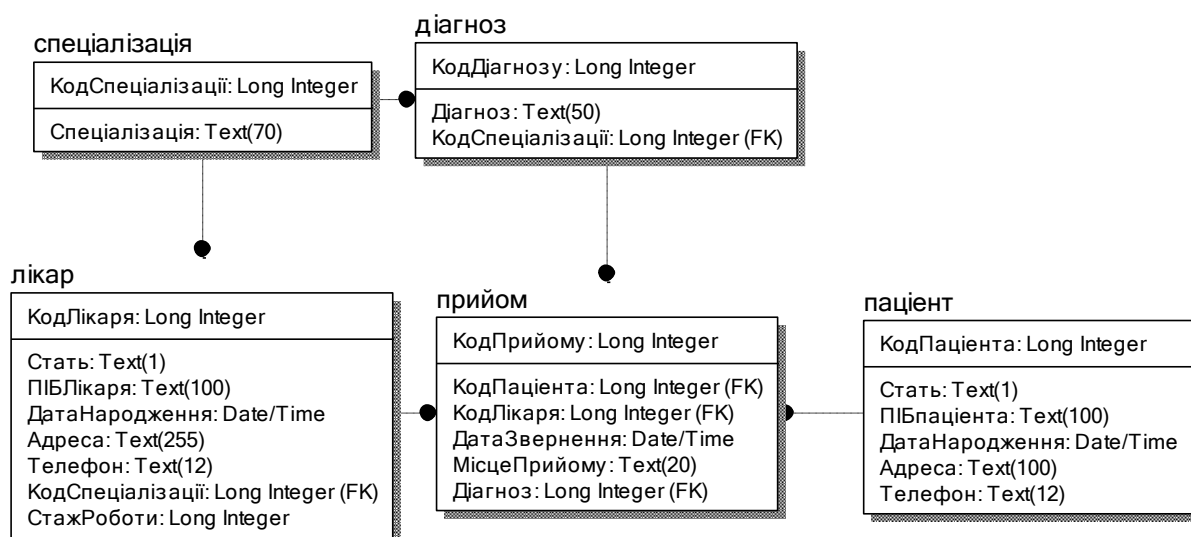


Рисунок 2.3 – Діаграма фізичного рівня

2.2 Проектування структури програми і взаємодії модулів

2.2.1 Специфікація підзадач і способів їх взаємодії

Робота з даними в режимі таблиці має істотний недолік: якщо полів занадто багато, вони не вміщаються на екрані і доводиться вдаватися до різних маніпуляцій, щоб оптимізувати їх вигляд: наприклад, прибирати деякі стовпці, змінювати їх положення [9].

Після створення бази даних необхідно створити інтерфейс системи у вигляді форм для перегляду даних. Форма може служити засобом захисту бази даних від некваліфікованих користувачів, а також ширмою, що затуляє від цікавих очей конфіденційну інформацію.

Перед створенням форм необхідно визначити основні функції системи.

Розглянемо наступні задачі та підзадачі системи:

- ведення довідників:
 - а) пацієнт;
 - б) лікар;
 - в) прийом;
 - г) діагноз;
 - д) спеціалізація;
- отримання інформації:
 - а) про кількість викликів додому у вказаний день;
 - б) про кількість хворих вказаною хворобою;
 - в) про хворобу, якою хворіють найчастіше.
- Вихід.

На рисунку 2.4 представлено дерево функцій програмних модулів, які використовуються в АІС обліку роботи поліклініки.

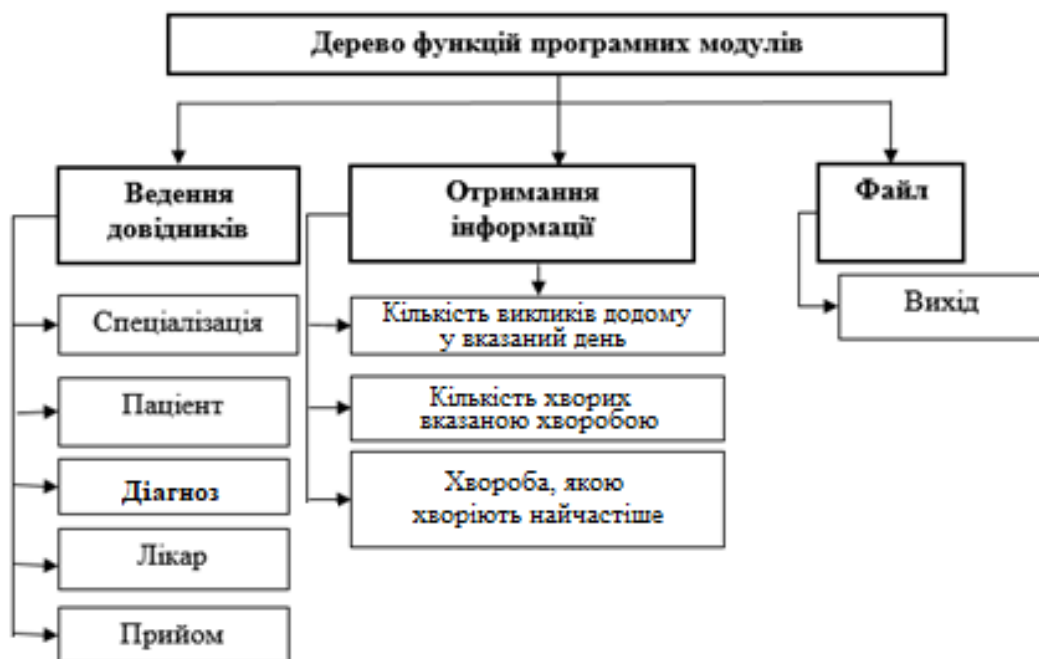


Рисунок 2.4 – Функціональна схема програми

Аналізуючи функціональну схему програми, розробимо структуру сценарію діалогу (рисунок 2.5).

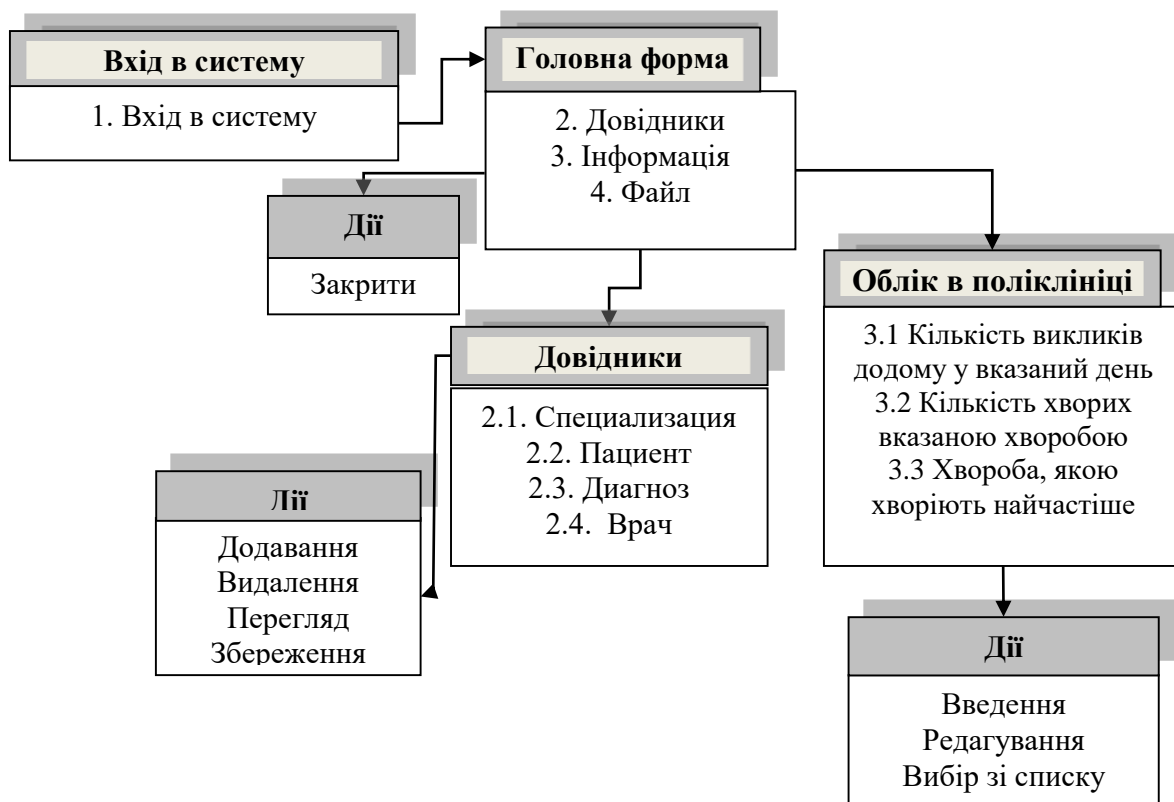


Рисунок 2.5 – Сценарій діалогу

На основі даної схеми формують меню системи.

2.2.2 Узагальнений алгоритм логічної структури програми

Модулі програми відповідають створюваним формам, код модуля генерується автоматично при створенні форми, а потім, при необхідності доповнюється процедурами і функціями за бажанням розробника [14].

Аналізуючи функціональну схему програми, виконаємо опис програмних модулів, що відображають структуру форм інтерфейсу (таблиця 2.9). Форми інтерфейсу подано в інструкції користувача.

Таблиця 2.9 – Опис функцій програмних модулів – логічної структури програми

Найменування модуля	Функції модуля
Модуль Unit1	Головний модуль програми. Містить процедури і функції для виклику форм через систему меню.
Модуль Unit2	Містить зумовлені процедури і функції, які необхідно виконати при роботі з довідником «Прийом»
Модуль Unit3	Містить процедури і функції, які необхідно виконати при роботі з довідником «Діагноз»
Модуль Unit4	Містить процедури і функції, які необхідно виконати при роботі з довідником «Пацієнти»
Модуль Unit5	Містить процедури і функції, які необхідно виконати при роботі з довідником «Спеціалізація»
Модуль Unit6	Містить процедури і функції, які необхідно виконати при роботі з довідником «Лікарі»
Модуль Unit7	Містить процедури і функції, які необхідно виконати при роботі з результатами запитів

2.3 Опис засобів програмування

2.3.1 Структурне програмування

Для розробки програмного продукту, який вирішує проблеми реального світу, застосовуються моделі його об'єктів та об'єктно-орієнтована розробка, тобто розробка з формальними структурами, які представляють ці об'єкти у програмній системі.

В об'єктно-орієнтованій технології розробляється ПЗ представляється у вигляді трьох взаємопов'язаних моделей:

- 1) об'єктної моделі (представляє статичні, структурні аспекти системи);
- 2) динамічної моделі (описує роботу окремих елементів системи);
- 3) функціональної моделі (розглядає взаємодію окремих частин системи у процесі її роботи).

Об'єктно-орієнтована технологія передбачає побудову програми як набору взаємодіючих між собою та незалежних об'єктів (класів), які обробляють інформацію за допомогою передачі повідомлень один одному.

Об'єктна модель містить ті ознаки та властивості об'єкта, які є суттєвими для програмного продукту, що розробляється. У її основі лежать абстрагування, модульність, інкапсуляція, ієрархія.

Створення об'єктно-орієнтованої програми починається з об'єктно-орієнтованого аналізу. Найбільшого поширення серед методів аналізу ПО отримав метод, що враховує інформаційну модель системи [2].

Відповідно до результатів об'єктно-орієнтованого аналізу формуються моделі, об'єктно-орієнтований підхід допомагає зменшити складність, але, водночас, підвищити надійність ПЗ. Крім того, ООП забезпечує можливість модифікації та повторного використання окремих програмних модулів, не торкаючись інших компонентів.

До найбільш істотних недоліків об'єктно-орієнтованих методологій відносять відсутність методу, що однаково добре реалізує етапи аналізу вимог та проектування, а також відсутність стандартизації для подання об'єктів та взаємодії між ними.

У подоланні виділених недоліків використається уніфікована методологія UML.

2.3.2 Мова програмування C++ та інструментальне середовище програмування

Розглянемо програмні засоби розробки інтерфейсу інформаційної системи [12].

Borland C++ Builder – це розроблений компанією Borland інструмент швидкої розробки додатків, який дозволяє створювати програми на C++, використовуючи середовище розробки та бібліотеку компонентів [13].

C++ Builder – це додаток SDI, головне вікно якого містить панель інструментів, що настроюється, і палітру компонентів. Крім того, за замовчуванням під час запуску C++ Builder відображаються вікно «Інспектор об'єктів» та форма нової програми. Під вікном форми заявки знаходиться вікно редактора коду.

Форми є основою програм C++ Builder. Створення інтерфейсу програми складається з додавання елементів об'єктів C++ Builder, званих компонентами, у вікно форми. Компоненти C++ Builder розташовані на палітрі компонентів, виконаної у вигляді багатосторінкового блокнота. Важливою особливістю C++ Builder є те, що він дозволяє створювати власні компоненти і налаштовувати палітру компонентів, а також створювати різні версії палітри компонентів для різних проектів.

Компоненти поділяються на видимі (візуальні) та невидимі (не візуальні). Візуальні компоненти з'являються під час виконання та розробки. Прикладами є кнопки та редаговані поля. Невізуальні компоненти з'являються під час розробки як піктограм на формі. Вони ніколи не видно під час виконання, але мають певні функції (наприклад, вони надають доступ до даних, викликають стандартні діалоги Windows і т.д.)

Кожен компонент C++ Builder має три типи характеристик: властивості, події та методи.

При виборі компонента з палітри та додаванні його у форму інспектор об'єктів автоматично покаже властивості та події, які можна використовувати з цим компонентом.

Властивості компонента – це атрибути компонента, які визначають його зовнішній вигляд та поведінку. Багато властивостей компонента в стовпці властивостей мають значення за промовчанням (наприклад, висоту кнопок).

Властивості компонента відображаються на сторінці властивостей. Інспектор об'єктів відображає опубліковані властивості компонентів [14].

При визначенні властивостей компонента під час розробки необхідно вибрати компонент у формі, відкрити сторінку властивостей в інспекторі об'єктів, вибрати певну властивість і змінити його за допомогою редактора властивостей (це може бути текст або число, що залишився поле, список, що розкривається, список, що розкривається, діалогове вікно і т.д. д.).

Сторінка «Події» Інспектора об'єктів показує список подій, що розпізнаються компонентом (програмування для операційних систем з графічним інтерфейсом користувача, зокрема, для Windows і включає опис реакції програми на певні події та самої операційної системи). Кожен компонент має власний набір обробників подій.

Щоб додати обробник подій, використовують мишу, щоб вибрати компонент у формі, який потребує обробника подій, потім C++ Builder повинен згенерувати прототип обробника подій і показати його в редакторі коду. Це автоматично генерує текст порожньої функції, і редактор відкривається там, де має бути введений код. Курсор розташований усередині дужок оператора {...}.

Далі необхідно ввести код, який має бути виконаний у разі виникнення події. Обробник подій може мати параметри, які вказуються після імені функції у дужках.

Метод – це функція, що з компонентом і яка оголошена як частина об'єкта. Створюючи обробники подій можна викликати методи.

При створенні форми модуль і файл заголовка з розширенням *.h обов'язково генеруються, тоді як під час створення нового модуля не потрібно зв'язуватися з формою (наприклад, якщо він містить процедури розрахунку).

Файли, з яких складається програма – форми та модулі – зібрані в проект. Менеджер проектів відображає списки файлів та модулів програми та дозволяє переміщатися між ними. За замовчуванням новий проект називається Project1.cpp.

За замовчуванням проект спочатку містить файли для однієї форми та вихідний код одного модуля. Однак більшість проектів містять кілька форм та модулів.

C++ Builder має вбудовану контекстно-залежну довідкову систему, яка доступна для будь-якого елемента інтерфейсу і є широким джерелом довідкової інформації про C++ Builder.

2.4 Опис програми

Відповідно до правил перетворення схеми об'єкт-відношення в реляційну модель об'єкти перетворюються на таблиці. Ім'я таблиці відповідає назві об'єкта. Властивості об'єкта (атрибути) стають полями таблиць бази даних, кожна таблиця має мати ключове поле – первинний ключ.

Отримуємо наступну схему даних (рисунок 2.6).

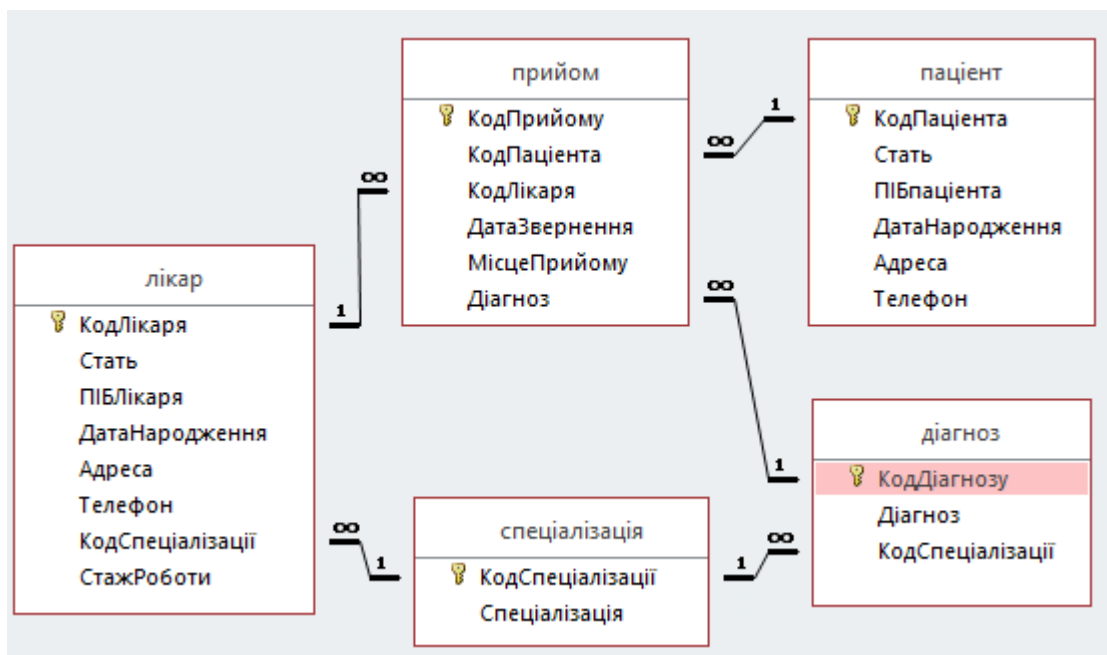


Рисунок 2.6 – Схема таблиць бази даних MSAccess

Перевіряємо зв'язки, у всіх зв'язках є забезпечення цілісності даних, додаємо також каскадне оновлення.

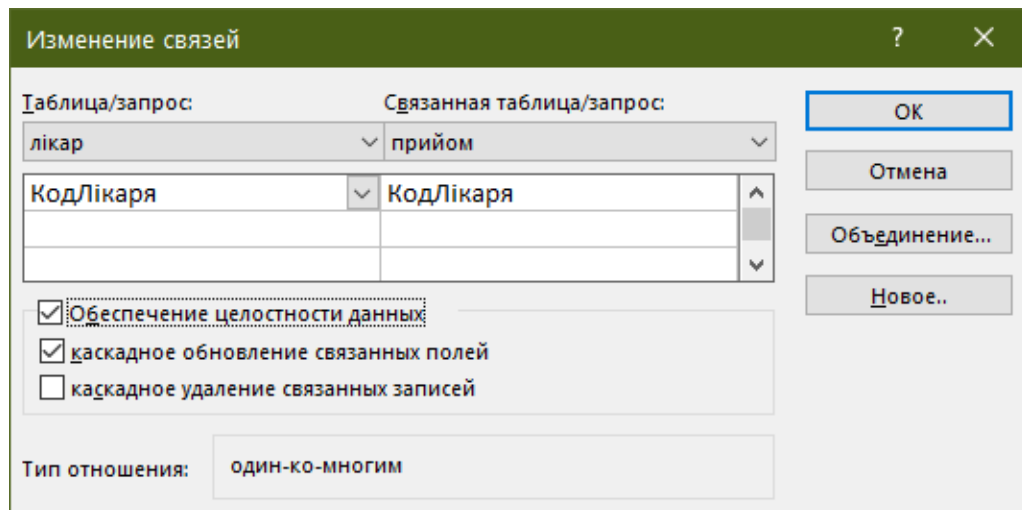


Рисунок 2.7 – Забезпечення цілісності даних

Для вибірки даних із бази даних на основі створених таблиць використовуються запити до бази даних. Запити були створені у базі даних мовою SQL. Усього було створено три запити до бази даних.

Запит про кількість викликів додому у вказаний день.

```
SELECT прийом.ДатаЗвернення, прийом.МісцеПрийому,
Count(прийом.ДатаЗвернення) AS ВсьогоВикликів
FROM прийом
GROUP BY прийом.ДатаЗвернення, прийом.МісцеПрийому
HAVING (((прийом.МісцеПрийому)=«у пацієнта»));
```

Запит про кількість хворих вказаною хворобою.

```
SELECT діагноз.Діагноз, Count(прийом.Діагноз) AS КількістьХворих
FROM діагноз INNER JOIN прийом ON діагноз.КодДіагнозу = прийом.Діагноз
GROUP BY діагноз.Діагноз;
```

Запит про хворобу, якою хворіють найчастіше.

```
SELECT Діагноз, КількістьХворих
FROM ЗапитКількістьХворихВказаноюХворобою
WHERE КількістьХворих=(
SELECT Max(ЗапитКількістьХворихВказаноюХворобою.КількістьХворих) AS
НайчастішеХворіють
FROM ЗапитКількістьХворихВказаноюХворобою);
```

Додаткові умови фільтрації даних створено при розробці інтерфейсу.

Для зручності користування інформаційною системою було створено головну форму. Вона містить меню для відкриття інших форм для введення та редагування даних по всіх перерахованих вище об'єктах. При зміні інформації у формах автоматично оновлюється інформація в таблицях бази даних, при додаванні записів у форми автоматично додаються дані до таблиць бази даних.

Розроблена програма виконує такі функції:

- керування програмою за допомогою меню головної кнопкової форми;
- введення, форматування та видалення даних з БД за допомогою відповідних форм;
- виконання запитів.

Програма працює в діалоговому режимі, який надає користувачеві обмежену можливість взаємодіяти з інформацією, що зберігається в системі в режимі реального часу, отримуючи при цьому всю необхідну інформацію для вирішення функціональних завдань.

Форми – це засіб представлення інформації для перегляду, зміни чи друкування даних у вигляді, зручному для сприйняття користувачами. Використання форм суттєво полегшує введення та контроль даних. Крім того, форми є основною частиною інтерфейсу прикладної програми навколо форм і будується весь алгоритм роботи програми, так як кінцевий користувач не бачить нічого, крім набору форм. Події, що виникають під час роботи з формами, визначають логіку роботи програми.

Вибравши необхідну кнопку або елемент меню на формі, користувач переходить до наступної форми. У цих формах виконуються введення даних, пошук та виведення інформації у певні компоненти на формі. На головній формі є пункт меню виходу із програми.

При запуску програми відкривається головна форма. У головній формі необхідно вибрати потрібний режим роботи, вибравши меню:

- Файл;
- Довідники;
- Облік в поліклініці.

3. ЗАКЛЮЧНА ЧАСТИНА

3.1. Тестування програми і аналіз отриманих результатів

Дуже важливим етапом розроблення є проведення випробувань програми. Проведення випробувань програми призначене для аналізу відповідності розробленого програмного засобу вимогам, що висуваються до нього [24].

Для контролю за правильністю вирішення завдання та роботи програми, як метод проведення випробувань виберемо тестування роботи системи за методом «Чорної скриньки».

Тестування за методом «Чорної скриньки» – це тестування програмного забезпечення, при якому функціональні можливості програмних додатків перевіряються без знання внутрішньої структури коду, деталей реалізації та внутрішніх шляхів. Тестування за методом «Чорної скриньки» здебільшого фокусується на введенні та виведенні програмних додатків і повністю ґрунтується на вимогах та специфікаціях програмного забезпечення (рисунок 3.1).

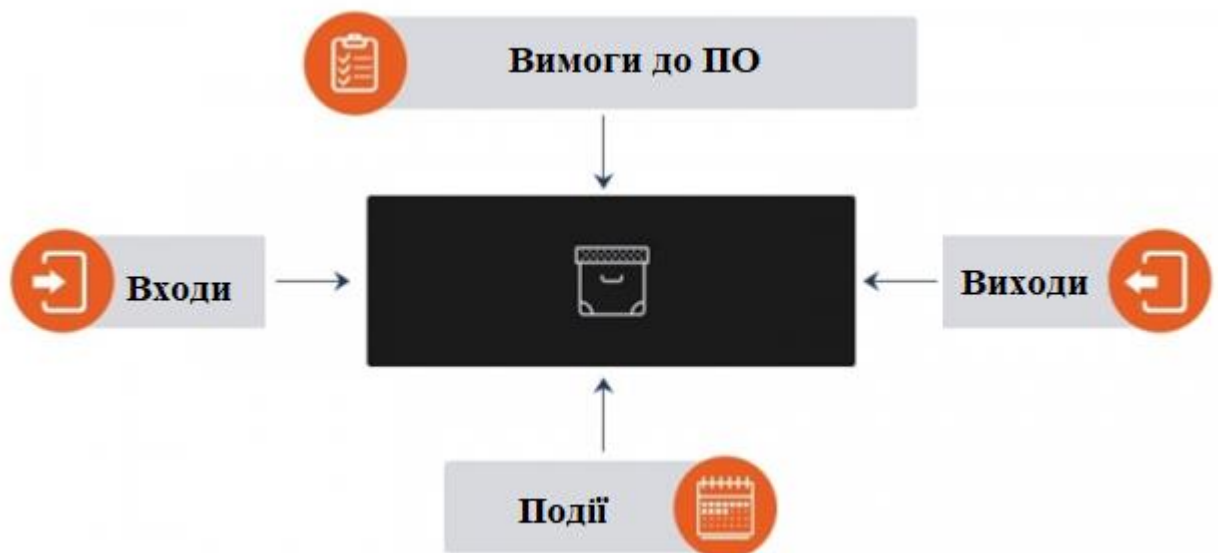


Рисунок 3.1 – Схема тестування за методом «Чорної скриньки»

Тестування за методом «Чорної скриньки» включає наступні етапи:

- вивчення вихідних вимог до програмного забезпечення та специфікацій проекту;
- вибір допустимих вхідних даних для позитивних тестових сценаріїв та недопустимих вхідних даних для негативних.;
- визначення очікуваних результатів як для позитивних, так і для негативних тестових сценаріїв;
- підготовка тестових випадків з усіма певними вхідними даними;
- виконання готових тест-кейсів;
- порівняння фактичних результатів з очікуваними;
- повторне тестування після виправлення помилок, якщо такі є.

Для тестування програми за методом «Чорної скриньки» необхідно виділити правильні та неправильні класи еквівалентності та виконати дії, як для правильних, так і для неправильних класів еквівалентності та щодо реакції системи визначити, чи правильно працює програма.

Виділимо правильні класи еквівалентності:

1. Файл бази даних розташований у папці з програмою.
2. Під час введення коректних значень полів таблиць довідників виконується перевірка тип даних.
3. При натисканні на кнопки виконуються запрограмовані дії.

Виділимо неправильні класи еквівалентності:

4. У папці програмою немає файлу бази даних.
5. У разі введення некоректних значень полів таблиць довідників виконується перевірка на тип даних.
6. Введення неправильних логіна або паролю у вікні з'єднання з базою даних.

Результати тестування програми на відповідність класам еквівалентності наведено в таблиці 3.1.

Таблиця 3.1 – Результати тестування програми

№	Клас еквівалентності	Прогнозований результат	Результат роботи програми	Висновок
Правильні класи еквівалентності				
1	Файл бази даних розташований у папці з програмою	Повинна відкриватися головна форма	Відкривається головна форма	Програма працює правильно
2	При введенні коректних значень полів таблиць довідників виконується перевірка на тип даних	При записі в базу коректних даних не повинно бути попереджувальних повідомлень	При записі в базу коректних даних немає попереджувальних повідомлень	Програма працює правильно
3	Дії кнопок запрограмовані відповідно їх призначенню	Повинні виконуватись запрограмовані дії для кнопок	Виконуються запрограмовані дії для кнопок	Програма працює правильно
Неправильні класи еквівалентності				
4	У папці програмою немає файлу бази даних	Повинне видаватися попередження про відсутність бази даних	Видається попередження «Не знайдено файл бази даних»	Програма працює правильно
5	При введенні некоректних значень полів таблиць довідників виконується перевірка на тип даних	При розбіжності типів дані в поле не можуть вводитися	При розбіжності типів дані в поле не вводяться	Програма працює правильно
6	Введення неправильних логіна або паролю у вікні з'єднання з базою даних	Повинно виводитися відповідне повідомлення про невідповідність логіна або паролю	Виводиться відповідне повідомлення про невідповідність логіна або паролю	Програма працює правильно

Робота програми була перевірена на різних вхідних даних, які перевіряли правильні та неправильні класи еквівалентності. При цьому програма працювала, як і очікувалося.

Результати роботи програми на всіх класах еквівалентності відповідають вимогам до програми.

3.2. Інструкція користувача

При запуску програми відкривається головна форма програми (рисунок 3.2).

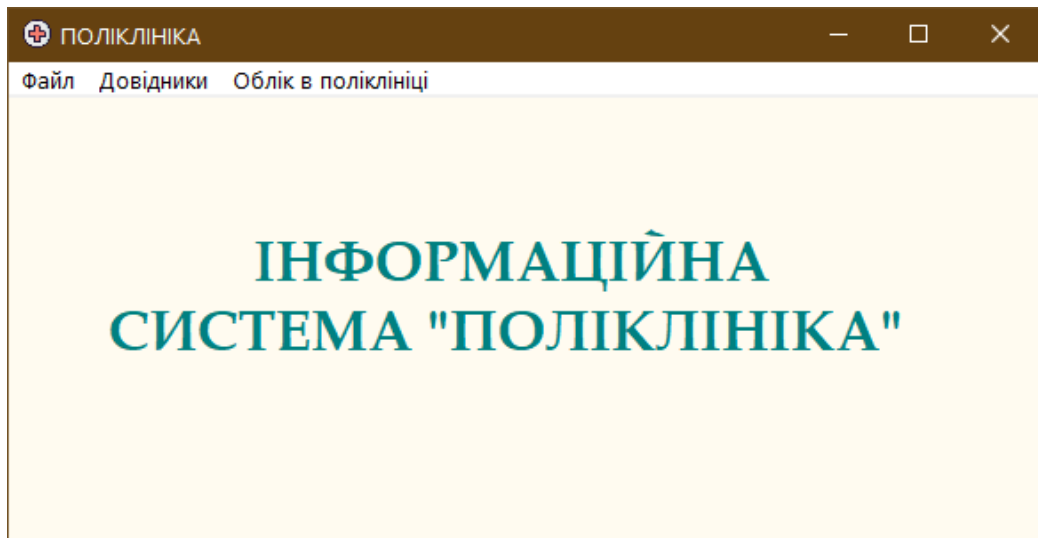
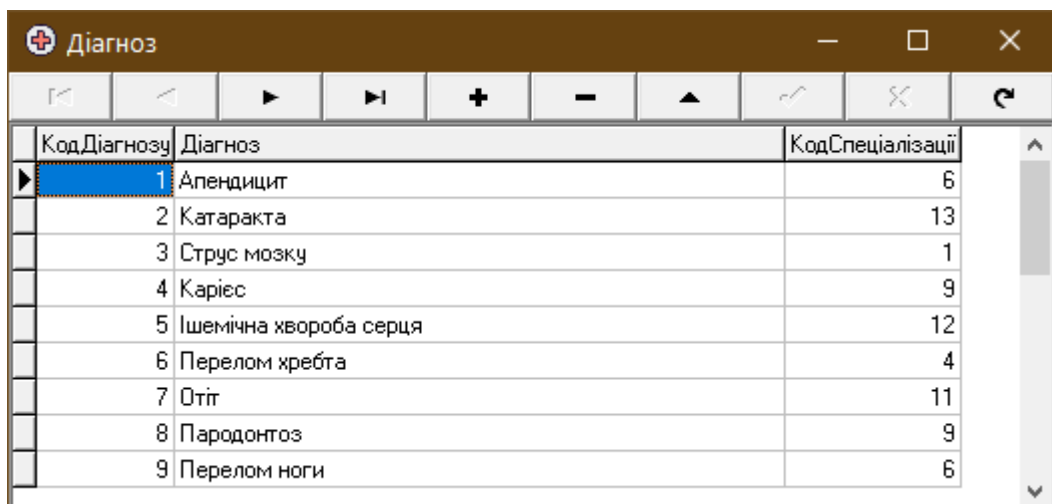


Рисунок 3.2 – Головна форма програми

У головній формі необхідно вибрати потрібний режим роботи, вибравши меню:

- Файл;
- Довідники;
- Облік в поліклініці.

При виборі меню «Довідники» можна вибрати окремі форми відповідних пунктів меню (рисунок 3.3-3.7).



КодДіагнозу	Діагноз	КодСпеціалізації
1	Апендицит	6
2	Катаракта	13
3	Струс мозку	1
4	Карієс	9
5	Ішемічна хвороба серця	12
6	Перелом хребта	4
7	Отіт	11
8	Пародонтоз	9
9	Перелом ноги	6

Рисунок 3.3 – Форма «Діагноз»

КодСпеціалізації	Спеціалізація
1	Невролог
2	Гінеколог
3	Терапевт
4	Травматолог

Рисунок 3.4 – Форма «Спеціалізація»

КодЛікаря	Стать	ПІБЛікаря	ДатаНародження	Адреса	Телефон	КодСпеціалізації	СтажРоботи
1	м	Шукин Павел Генна	26.04.1973	Київ, вул. Бакалинська, 6.64	484548	1	15
2	ж	Гордеева Наталья	26.06.1973	Київ, вул. Маршала Жукова, буд.3/2	459786	2	15
3	м	Корнилов Максим	01.12.1983	Київ, вул. Р.Зорге, буд.17	458976	3	15
4	м	Терентьев Виталий	10.03.1984	Київ, вул. Широка, 267-8	125487	4	10
5	м	Гущин Станислав А	13.06.1994	Київ, вул. Пирогова, 2-3	684512	5	8
6	м	Субботин Александр	01.12.1995	Регіт, вул. Орлова, 34-45	348715	6	8
7	ж	Ершова Екатерина	10.07.1995	Київ, вул.Комарова, буд.15, кв.10	251747	3	8
8	м	Шестаков Дмитрий	02.07.1976	Київ, вул. Аксенова, буд.3, кв.100	958476	7	17
9	ж	Мельникова Анна Е	21.08.1986	Київ, вул.Маркова, буд.50, кв.2	358497	8	8
10	м	Нестеров Алексей	01.06.1997	Київ, вул.Ланіна, буд.5, кв.40	357951	9	9
11	м	Пинцетов Илья Фе	30.09.1977	Київ, вул. Бакалинська, 6.84	852456	10	10
12	м	Архипов Николай М	17.06.1973	Київ, вул. Маршала Жукова, буд.2/2	951753	6	12
13	м	Богданов Анатолий	21.07.1983	Київ, вул. Р.Зорге, буд.27	121582	11	11
14	ж	Осипова Надежда	26.07.1973	Київ, вул. Широка, 28	125205	12	22

Рисунок 3.5 – Форма «Лікарі»

КодПацієнта	Стать	ПІБпацієнта	ДатаНародження	Адреса	Телефон
123	м	Хохлов Николай Денисович	12.09.1975	пр-т А. Камалеева,	73363269
194	м	Тополев Михаил Иванович	18.12.1988	1-я Магістральна в	74160561
210	м	Кузьмин Антон Николаевич	19.11.1985	вул. Пришвіна, 6. 8,	51046276
252	м	Шаров Дмитрий Антонович	21.09.1988	пл. Ланіна / вул. Бу	54131309
297	ж	Доронина Екатерина Дмитриевна	15.08.1952	5-й мікрорайон, кв	61264879
318	м	Захаров Александр Александрович	25.07.1976	вул.Чернишевської	43764746
339	ж	Гришина Ольга Владимировна	16.04.1973	Чланський пров., 6	82145592
359	ж	Блинова Наталья Кирилловна	13.07.1982	Лопухінського пров.	65463969
375	м	Гаврилов Алексей Владимирович	12.06.1990	вул.Амурська, 225	62535400
442	ж	Артемьева Марина Павловна	26.03.1979	вул. Ласточкина б.	166378907
458	ж	Федорова Валентина Ильинична	05.06.1941	вул. Абакумова б.	26739152
504	м	Сидоров Виктор Геннальевич	23.06.1987	вул. Народна, 38	74704713
508	м	Капустин Валерий Семенович	04.10.1965	вул. Першотравнев	54853183
524	м	Горшков Кирилл Федорович	28.10.1964	вул. Пермська, 19	43724767

Рисунок 3.6 – Форма «Пацієнти»

Прийом пацієнтів						
	Г	◀	▶	+	-	▲
	◂	◃	▹	▸	↶	↷
	КодПрийому	КодПацієнта	КодЛікаря	ДатаЗвернення	МісцеПрийому	Діагноз
▶	1	123	6	06.01.2023	поліклініка	1
	2	194	16	07.01.2023	поліклініка	2
	3	194	1	14.01.2023	поліклініка	3
	4	252	10	15.01.2023	у пацієнта	4
	5	297	14	19.01.2023	поліклініка	5
	6	318	4	09.02.2023	поліклініка	6
	7	339	13	10.02.2023	поліклініка	7
	8	359	10	12.02.2023	у пацієнта	8
	9	375	7	20.02.2023	поліклініка	9
	10	442	9	23.02.2023	поліклініка	10
	11	458	11	28.02.2023	поліклініка	11
	12	504	14	02.03.2023	поліклініка	5

Рисунок 3.7 – Форма «Прийом пацієнтів»

За допомогою панелі навігації ці форми можна використовувати для введення нових даних або зміни інформації.

При виборі меню «Інформація по поліклініці» відкривається однойменна форма, в якій можна вибрати потрібну сторінку для виконання запитів та перегляду результатів виконання.

На сторінці «Кількість викликів додому у вказаний день» можна переглянути інформацію про те, скільки викликів додому було в поліклініці (рисунок 3.8).

Інформація по поліклініці			
Кількість викликів додому у вказаний день			Кількість хворих вказаною хворобою
			Хвороба, якою хворіють найчастіше
ДатаЗвернення	МісцеПрийому	ВсьогоВикликів	
▶ 15.01.2023	у пацієнта	1	Виберіть дату <input type="text" value="09.04.2023"/> <input type="button" value="Фільтр"/> <input type="button" value="Скинути"/>
12.02.2023	у пацієнта	1	
16.03.2023	у пацієнта	1	
26.03.2023	у пацієнта	1	
06.04.2023	у пацієнта	2	

Рисунок 3.8 – Форма «Інформація по поліклініці», сторінка «Кількість викликів додому у вказаний день»

За допомогою фільтра можна вибрати конкретну дату (рисунк 3.9).

ДатаЗвернення	МісцеПрийому	ВсьогоВикликів
06.04.2023	у пацієнта	2

Виберіть дату: 06.04.2023

Фільтр Скинути

Рисунок 3.9 – Форма «Інформація по поліклініці», сторінка «Кількість викликів додому у вказаний день» в режимі фільтрації

На сторінці «Кількість хворих вказаною хворобою» можна переглянути відповідну інформацію (рисунк 3.10).

Діагноз	КількістьХворих
Алкоголізм	1
Апендицит	1
Аритмія	1
Бронхіт	1
Геморой	1

Виберіть хворобу: [dropdown]

Фільтр Скинути

Рисунок 3.10 – Форма «Інформація по поліклініці», сторінка «Кількість хворих вказаною хворобою»

За допомогою фільтра можна вибрати конкретну хворобу (рисунк 3.11).

Діагноз	КількістьХворих
Отіт	2

Виберіть хворобу: Отіт

Фільтр Скинути

Рисунок 3.11 – Форма «Інформація по поліклініці», сторінка «Кількість хворих вказаною хворобою» в режимі фільтрації

На сторінці «Хвороба, якою хворіють найчастіше», можна дізнатися відповідну інформацію. Ця інформація буде корисною, наприклад, введення карантину (рисунок 3.12).

The screenshot shows a window titled "Інформація по поліклініці" with a standard Windows interface (minimize, maximize, close buttons). The window contains a tabbed interface. The active tab is "Хвороба, якою хворіють найчастіше". Other tabs include "Кількість викликів додому у вказаний день", "Кількість хворих вказаною хворобою", and "Хвороба, якою хворіють найчастіше". The main content area displays a table with two columns: "Діагноз" and "Кількість хворих". The table lists five diseases, each with a count of 2. The first row, "Ішемічна хвороба серця", is highlighted in blue. A small arrow icon is visible next to the "Кількість хворих" header.

Діагноз	Кількість хворих
Ішемічна хвороба серця	2
Короста	2
Отіт	2
Пародонтоз	2
Перелом ноги	2

Рисунок 3.12 – Форма «Інформація по поліклініці», сторінка «Хвороба, якою хворіють найчастіше»

Для виходу з програми необхідно вибрати пункт меню «Вихід» у меню «Файл» на головній формі програми.

Висновки

Під час виконання курсової роботи були вивчені питання, що стосуються загальної характеристики діяльності поліклініки, а також визначення постановки завдання. Можна зробити наступні висновки:

При вивченні автоматизації роботи поліклініки був виділений і описаний бізнес-процес, який необхідно автоматизувати, побудовані мнемосхеми процесів обліку «як-є» і «як-має-бути».

Поставлено завдання на створення інформаційної системи, виявлені відповідальні особи – користувачі системи, сформульовані функції системи, її вхідні і вихідні дані.

Також в процесі виконання курсової роботи було виконано моделювання бази даних і опис інтерфейсу АІС обліку роботи поліклініки.

Результатом курсової роботи стала АІС обліку роботи поліклініки.

Інформаційна система складається з бази даних MS Access і файлів проекту C++Builder на мові програмування C ++.

Основні відомості про роботу поліклініки міститися в таблицях бази даних. Також були розроблені форми інтерфейсу користувача.

Отримання інформації з бази даних виконується на основі розроблених запитів до бази даних, інформація виводиться в форми додатку.

Список використаної літератури

1. Медичні інформаційні технології в Україні [Електронний ресурс] / Режим доступу: <https://www.medsprava.com.ua/article/855-medichn-nformatsyn-tehnolog-v-ukran> (дата звернення: 05.04.2023)
2. Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н.В. Федоров. – М.: МГИУ, 2018. – 280 с.
3. Медичні інформаційні системи: огляд можливостей і приклади використання [Електронний ресурс] / Режим доступу: <https://evergreens.com.ua/ua/articles/medical-information-systems.html> (дата звернення: 05.04.2023)
4. Трофименко О. Г. Організація баз даних : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. – Одеса : Фенікс, 2019. – 246 с.
5. Базы даних/організація баз даних: методичні рекомендації для виконання практичних занять / уклад.: Ю. Є. Добришин – Київ: Університет економіки та права «КРОК», 2017. – 130 с.
6. Недашківський О.М.. Планування та проектування інформаційних систем. – Київ, 2014. – 215 с.
7. Басюк Т. М. Основи інформаційних технологій : навчальний посібник / Т.М. Басюк Н.О. Думанський О.В. Пасічник ; За ред. В.В. Пасічника. – Львів : Новий світ-2000, 2010. – 390 с.
8. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. – Ужгород : Ужгородський національний університет», 2018. – 118 с.
9. SQL Підручник [Електронний ресурс]. – Режим доступу: <https://w3schoolsua.github.io/sql/index.html> (дата звернення: 02.04.2023)
10. Авраменко В.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.

Додаток А. Код програми

```
//-----
#include <vcl.h>
#pragma hdrstop
//-----
USEFORM("Unit1.cpp", Form1);
USEFORM("Unit3.cpp", Form3);
USEFORM("Unit4.cpp", Form4);
USEFORM("Unit5.cpp", Form5);
USEFORM("Unit6.cpp", Form6);
USEFORM("Unit7.cpp", Form7);
USEFORM("Unit2.cpp", Form2);
//-----
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TForm1), &Form1);
        Application->CreateForm(__classid(TForm3), &Form3);
        Application->CreateForm(__classid(TForm4), &Form4);
        Application->CreateForm(__classid(TForm5), &Form5);
        Application->CreateForm(__classid(TForm6), &Form6);
        Application->CreateForm(__classid(TForm7), &Form7);
        Application->CreateForm(__classid(TForm2), &Form2);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    catch (...)
    {
        try
        {
            throw Exception("");
        }
        catch (Exception &exception)
        {
            Application->ShowException(&exception);
        }
    }
    return 0;
}
//-----
//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "Unit6.h"
#include "Unit7.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```

//-----
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
//-----
//////////
void __fastcall TForm1::FormCloseQuery(TObject *Sender, bool &CanClose)
{
if (MessageDlg("Закрити додаток?", mtConfirmation, TMsgDlgButtons() << mbYes
<< mbNo, 0) == mrNo)
{
CanClose=False;
return ;
} else Application->Terminate();
}
//-----
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Form1->Position = poDesktopCenter;
}
//-----
void __fastcall TForm1::N2Click(TObject *Sender)
{
Close();
}
//-----
void __fastcall TForm1::N13Click(TObject *Sender)
{
Form3->Show();
if (Form3->DataSource1->DataSet->Active==false) Form3->DataSource1-
>DataSet->Active=true;
}
//-----
void __fastcall TForm1::N14Click(TObject *Sender)
{
Form5->Show();
if (Form5->DataSource1->DataSet->Active==false) Form5->DataSource1->DataSet-
>Active=true;
}
//-----
void __fastcall TForm1::N15Click(TObject *Sender)
{
Form6->Show();
if (Form6->DataSource1->DataSet->Active==false) Form6->DataSource1-
>DataSet->Active=true;
}
//-----
void __fastcall TForm1::N16Click(TObject *Sender)
{
Form4->Show();
if (Form4->DataSource1->DataSet->Active==false) Form4->DataSource1->DataSet-
>Active=true;
}
//-----
void __fastcall TForm1::N10Click(TObject *Sender)
{
Form7->Show();
Form7->TabSheet1->Show();
if (Form7->DataSource1->DataSet->Active==false) Form7->DataSource1-
>DataSet->Active=true;
}

```

```

    if (Form7->DataSource2->DataSet->Active==false) Form7->DataSource2->
    DataSet->Active=true;
    if (Form7->DataSource3->DataSet->Active==false) Form7->DataSource3->
    DataSet->Active=true;
    if (Form7->DataSource4->DataSet->Active==false) Form7->DataSource4->
    DataSet->Active=true;
}
//-----
void __fastcall TForm1::N4Click(TObject *Sender)
{
    Form2->Show();
    if (Form2->DataSource1->DataSet->Active==false) Form2->DataSource1->
    DataSet->Active=true;
}
//-----
//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//-----
__fastcall TForm2::TForm2(TComponent* Owner)
: TForm(Owner)
{
}
//-----

//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit3.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//-----
__fastcall TForm3::TForm3(TComponent* Owner)
: TForm(Owner)
{
}
//-----
void __fastcall TForm3::FormCreate(TObject *Sender)
{
    Form3->Position = poDesktopCenter;
}
//-----
//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit4.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//-----

```

```

__fastcall TForm4::TForm4(TComponent* Owner)
: TForm(Owner)
{
}
//-----
void __fastcall TForm4::FormCreate(TObject *Sender)
{
Form4->Position = poDesktopCenter;
}
//-----
//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit5.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//-----
__fastcall TForm5::TForm5(TComponent* Owner)
: TForm(Owner)
{
}
//-----
void __fastcall TForm5::FormCreate(TObject *Sender)
{
Form5->Position = poDesktopCenter;
}
//-----
//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit6.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
//-----
__fastcall TForm6::TForm6(TComponent* Owner)
: TForm(Owner)
{
}
//-----
void __fastcall TForm6::FormCreate(TObject *Sender)
{
Form6->Position = poDesktopCenter;
}
//-----
//-----
#include <vcl.h>
#pragma hdrstop
#include "Unit7.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
//-----
__fastcall TForm7::TForm7(TComponent* Owner)

```



```

: TForm(Owner)
{
}
//-----
void __fastcall TForm7::FormCreate(TObject *Sender)
{
Form7->Position = poDesktopCenter;
}
//-----
void __fastcall TForm7::Button3Click(TObject *Sender)
{
ADOTable2->Filtered=false;
ADOTable2->Filter="Ã³ãîîç Like "+QuotedStr(DBLookupComboBox1->Text);
ADOTable2->Filtered=true;
}
//-----
void __fastcall TForm7::Button4Click(TObject *Sender)
{
ADOTable2->Filtered=false;
}
//-----
void __fastcall TForm7::Button1Click(TObject *Sender)
{
ADOTable1->Filtered=false;
ADOTable1->Filter="ÃààÇâãðíáíÿ='"+DateToStr(DateTimePicker1->Date)+"'";
ADOTable1->Filtered=true;
}
//-----
void __fastcall TForm7::Button2Click(TObject *Sender)
{
ADOTable1->Filtered=false;
}
//-----

```