МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **“ Розробка системи інформаційної підтримки діяльності поліклініки ”**

Виконав: студент гр. Іт-41
Спеціальності 126 – «Інформаційні системи та технології»
(шифр і назва)

Звірко О.О.
(Прізвище та ініціали)

Керівник: к.е.н., доц. Шувар Б.І.
(Прізвище та ініціали)

Рецензент: к.т.н., доц. Стукалець І.Г.
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри

_____
(підпис)

д.т.н., професор, Тригуба А.М.
"____" _____ 202_ р.

ЗАВДАННЯ
на кваліфікаційну роботу

Звірку Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка системи інформаційної підтримки діяльності поліклініки »

Керівник роботи *к.е.н., доцент, Шувар Богдан Іванович*

Затверджені наказом по університету від *від 27.11.2023 року № 641/к-с*

2. Строк подання студентом роботи *10.06.2024 р.*

3. Вихідні дані до роботи: *технічна документація С++*

4. Зміст розрахунково-пояснювальної записки: (перелік питань, які потрібно розробити)

Вступ
1. Постановка завдання
2. Проєктування
3. Програмна реалізація проєкту
4. Охорона праці
Висновки
Бібліографічний список

5. Перелік графічного матеріалу: *Графічний матеріал подається у вигляді презентації*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3	*Шувар Б. І., доцент кафедри ІТ*		
4	*Городецький І.М., доцент кафедри управління проектами та безпеки виробництва*		

7. Дата видачі завдання __.__.202_ р.

Календарний план

№ з/п	Назва етапів дипломного проєкту	Терміни виконання етапів роботи	Примітка
1.	*Написання першого розділу*	*01.02.2024 – 01.03.2024*	
2.	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*02.03.2024 – 02.04.2024*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*03.04.2024 – 25.04.2024*	
4.	*Написання розділу «Охорона праці»*	*26.04.2024 – 28.04.2024*	
5.	*Завершення оформлення розрахункового-пояснювальної записки та аркушів ілюстраційного матеріалу*	*29.04.2024 – 29.05.2024*	
6.	*Завершення роботи цілому*	*30.05.2024 – 07.06.2024*	

Студент _____, _____Звірко О.О._____
(підпис) (Прізвище, ініціали)

Керівник роботи _____, _____Шувар Б. І._____
(підпис) (Прізвище, ініціали)

УДК: 614.2:004.6

Розробка системи інформаційної підтримки діяльності поліклініки.

Звірко О.О. Кафедра ІТ. - Дубляни, Львівський НУП, 2024.

Кваліфікаційна робота: 52 ст. текст. част., 15 рис., 3 табл., 13 літ. джерел.

Зроблено аналіз існуючих інформаційних систем у медичній сфері. Проаналізовано основні аналоги інформаційних систем, які використовуються в медичній сфері, їх функціональність, переваги та недоліки. Розглянуто етапи створення програми для інформаційної підтримки діяльності поліклініки.

Проаналізовано різні методи розробки. Досліджено принцип роботи та створено аналог системи для інформаційної підтримки діяльності поліклініки.

Проєктовано функціональне моделювання програми. Побудовано на основі ER діаграми діаграму концептуальних схем з використанням узагальнених конструкцій блоків. Розроблено та описано код програми з можливістю модернізації останньої. Запропоновано рекомендації з охорони праці під час роботи з комп’ютерною технікою.

Ключові слова: поліклініка, С++, комп'ютерні техніки, ООП, інформаційна підтримка.

Key word: polyclinic, C++, computer technics, object-oriented programming, informational support.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНИХ ТЕРМІНІВ

ІПД Інформаційна підтрика діяльності

МІС – Медичні інформаційні системи

UI – user interface (інтерфейс користувача)

UX – user experience (взаємодія користувача)

DFD – діаграма потоків даних

ІТ – інформаційні технології

ООП – Об'єктно-орієнтоване програмування

ЕМК – Електронні медичні картки

ВСТУП

Інформаційні технології у наш з вами час займають провідне місце в усіх сферах діяльності людини, зокрема, у медичній галузі, яка активно впроваджує новітні технології для оптимізації роботи, покращення якості обслуговування пацієнтів та підвищення ефективності управлінських процесів. Інформаційні системи дозволяють медичним закладам автоматизувати процеси збору, обробки та зберігання медичної інформації, що суттєво зменшує ймовірність людських помилок та сприяє покращенню якості медичних послуг.

Поліклініки постійно стикаються з низкою проблем, пов'язаних із веденням медичної документації, обліком пацієнтів, плануванням прийомів та контролем за виконанням медичних процедур. Традиційні методи управління інформацією часто є застарілими, що призводить до низької ефективності роботи медичного персоналу та зниження якості надання медичних послуг. Тому розробка програмного забезпечення для інформаційної підтримки діяльності поліклініки є надзвичайно актуальною темою.

Метою нашої роботи є розробка та впровадження системи, що дозволить отримати просту, але якісну інформаційну підтримку діяльності поліклініки. Це програмне забезпечення повинно автоматизувати основні процеси медичного закладу, зокрема ведення електронних медичних карток пацієнтів, облік прийомів, управління розкладом лікарів та моніторинг виконання медичних процедур.

Для досягнення поставленої мети, необхідно вирішити наступні задачі:

- провести аналіз існуючих інформаційних систем у медичній сфері та виявити їх сильні та слабкі сторони;
- визначити вимоги до програмного забезпечення для поліклініки, враховуючи специфіку роботи медичного закладу;
- розробити архітектуру системи, що забезпечить надійність, масштабованість та зручність використання;

Практична значущість роботи полягає у можливості впровадження розробленого програмного забезпечення в реальні умови роботи поліклініки.

Використання даної системи дозволить автоматизувати основні процеси медичного закладу, зменшити кількість помилок, пов'язаних з людським фактором, підвищити ефективність роботи медичного персоналу та покращити якість надання медичних послуг. Це, в свою чергу, сприятиме підвищенню задоволеності пацієнтів та навіть вплине на покращення загального стану здоров'я населення.

РОЗДІЛ 1. ПОСТАНОВКА ЗАВДАННЯ ТА ВІДБІР АРХІТЕКТУРНИХ ІНСТРУМЕНТІВ

1.1. Опис предметної області для інформаційної підтримки діяльності поліклініки

Предметна область даної кваліфікаційної роботи охоплює розробку програмного забезпечення для інформаційної підтримки діяльності поліклініки. Основними аспектами предметної області є медичні інформаційні системи, управління медичними даними, автоматизація процесів у медичних закладах, а також забезпечення ефективної взаємодії між пацієнтами та медичним персоналом. Поліклініка – це заклад, що надає амбулаторнополіклінічну допомогу населенню. Основні функції поліклініки включають:

Профілактичні заходи: проведення профілактичних оглядів, вакцинацій та інших заходів, спрямованих на попередження захворювань.

Діагностика та лікування: здійснення діагностичних процедур, встановлення діагнозів, призначення та проведення лікування.

Консультації спеціалістів: надання консультацій лікарями різних спеціальностей [7].

Лабораторні та інструментальні дослідження: проведення аналізів, рентгенографії, ультразвукових досліджень та інших діагностичних процедур.

Моніторинг стану здоров'я пацієнтів: відстеження стану здоров'я хронічних хворих та пацієнтів, які потребують постійного медичного нагляду.

Ефективна робота поліклініки вимагає відстеження великого обсягу інформації, зокрема:

дані про пацієнтів: особисті дані, медична історія, результати попередніх обстежень, призначення та рекомендації лікарів.

Результати лабораторних та інструментальних досліджень: своєчасне отримання та обробка результатів для швидкого прийняття медичних рішень.

Облік медичних послуг: реєстрація наданих медичних послуг, їх вартість.

В останні роки в медичній сфері активно використовуються інформаційні технології для покращення якості медичних послуг та підвищення ефективності роботи медичних закладів. До основних сучасних підходів належать: Електронні медичні картки (ЕМК): забезпечують централізоване зберігання медичної інформації про пацієнтів, що дозволяє лікарям швидко отримувати доступ до необхідних даних.

Телекомунікаційні технології - використання телемедицини для дистанційних консультацій, моніторингу стану здоров'я пацієнтів.

Інформаційні системи управління - комплексні рішення для автоматизації адміністративних та медичних процесів у закладі.

Системи підтримки прийняття рішень - допомагають лікарям у діагностиці та виборі оптимального лікування на основі аналізу медичних даних.

1.2. Аналіз існуючих інформаційних систем у медичній сфері

Для написання кваліфікаційної роботи, з програмних засобів, використали Visual Studio 2019 (через її зручність), Notepad++ (через його можливості підсвічування синтаксису та підтримки великої кількості мов програмування, у тому числі C++) і браузер Google Chrome, в зв’язку з його швидкістю та чудовою оптимізацією.

З апаратних засобів нами було використано персональний комп’ютер з такими характеристиками:

- Процессор Intel i5 11400 (6 ядер 12 потоків);
- Відеокарта RTX 2070 super 8gb;
- 16 gb озу, DDR4 частотою 3200 MHz, 2x8gb;
- SSD M2 на 512 gb;

У світі існує багато різних інформаційних систем, які використовуються в медичній сфері. Розглянемо основні з них [13], а також їх сильні та слабкі сторони.

Розглянемо першу аналогову інформаційну систему (рис. 1.1):

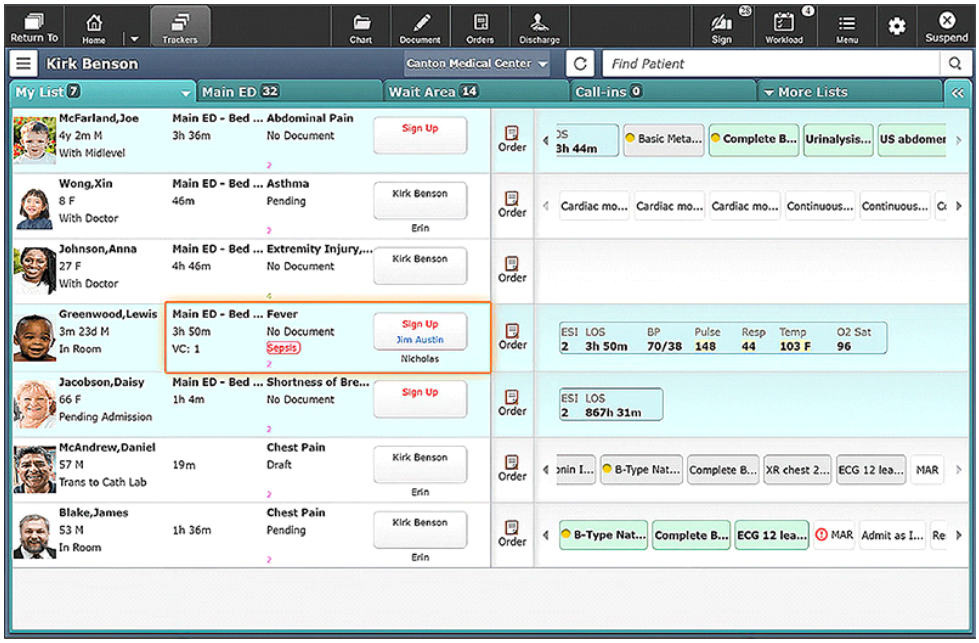

Рисунок 1.1 – Приклад систем: Meditech, Epic System

Сильні сторони: Комплексність: Ці системи охоплюють всі аспекти управління медичним закладом, включаючи електронні медичні картки, управління ресурсами, планування прийомів пацієнтів тощо. Інтеграція: MIC забезпечують інтеграцію різних модулів та підсистем, що дозволяє обмінюватися даними між різними відділами. Аналітика: Системи надають потужні інструменти для аналізу даних та створення звітів, що допомагає приймати обґрунтовані управлінські рішення. Слабкі сторони: Висока вартість: Впровадження та підтримка MIC вимагає значних фінансових витрат. Складність впровадження: Процес впровадження може бути тривалим та складним, вимагаючи спеціальних знань та навичок. Навчання персоналу - для ефективного використання системи необхідне навчання медичного персоналу, що може бути додатковим витратним фактором [3].

Розглянемо другу аналогову інформаційну систему (рис. 1.2):

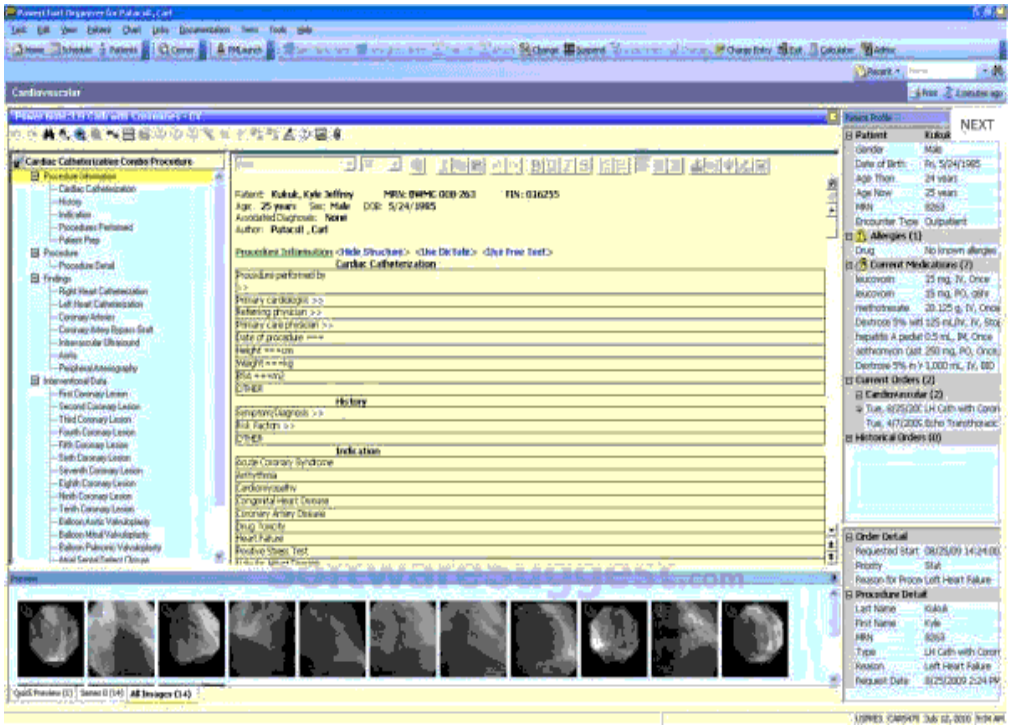

Рисунок 1.2 – Приклад систем: Allscripts, Cerner

Сильні сторони: Доступність інформації: ЕМК забезпечують швидкий доступ до медичних даних пацієнтів, що підвищує оперативність та точність діагностики і лікування. Зручність використання: Інтерфейси таких систем зазвичай інтуїтивно зрозумілі, що спрощує їх використання медичним персоналом. Покращення якості обслуговування: Завдяки точному обліку медичних даних, лікарі можуть надавати більш якісні медичні послуги. Слабкі сторони: Конфіденційність даних: Існує ризик порушення конфіденційності медичних даних, якщо система не має належного захисту. Інтероперабельність: Проблеми з інтеграцією ЕМК з іншими системами можуть ускладнити обмін інформацією між різними медичними закладами. Розглянемо третю аналогову інформаційну систему (рис. 1.3):

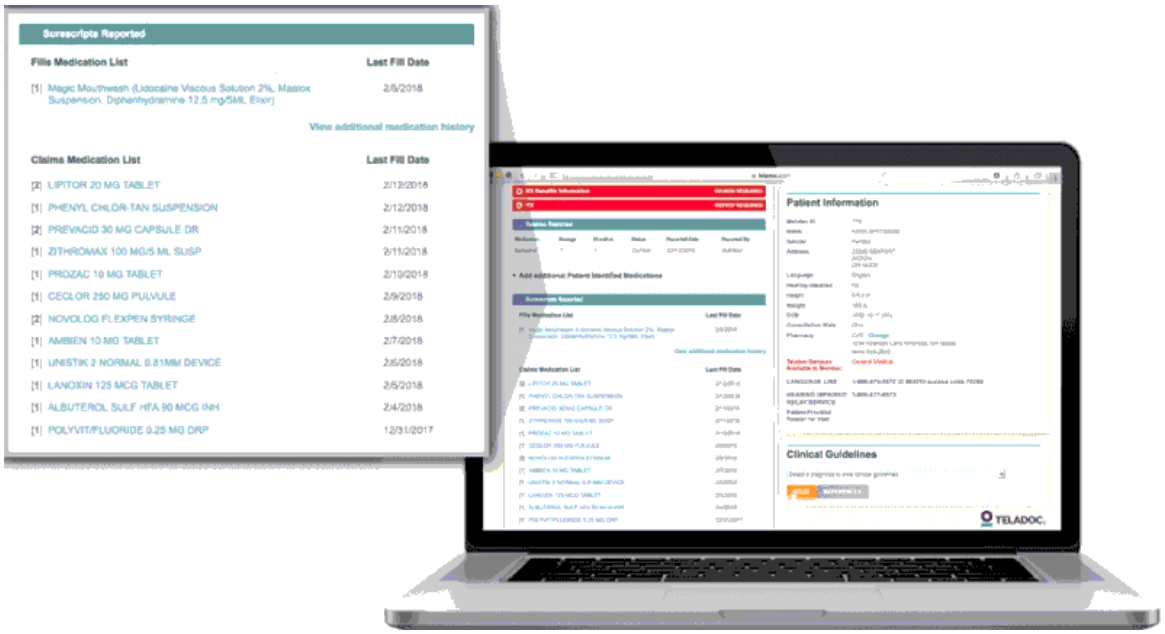

Рисунок 1.3 – Приклад систем: Teladoc, Amwell

Сильні сторони: Дистанційний доступ до медичних послуг: Телемедичні системи дозволяють пацієнтам отримувати консультації від лікарів на відстані, що особливо актуально у віддалених регіонах або під час пандемій. Зниження витрат: Використання телемедичних систем може зменшити витрати на поїздки та організацію прийомів у медичних закладах. Зручність для пацієнтів: Пацієнти можуть отримувати медичні послуги у зручний для них час без необхідності відвідувати поліклініку. Слабкі сторони: Технічні обмеження: Для використання телемедичних систем необхідно мати доступ до інтернету та відповідне обладнання, що може бути проблемою у деяких регіонах. Обмеженість фізичних обстежень: Дистанційний формат обмежує можливості фізичного огляду пацієнта, що може знижувати точність діагностики [8].

Отже, розробка нової інформаційної системи для підтримки діяльності поліклініки має враховувати ці аспекти, поєднуючи переваги існуючих рішень та мінімізуючи їх недоліки. Це дозволить створити ефективне, зручне та безпечне програмне забезпечення, яке підвищить якість медичних послуг та задоволеність пацієнтів.

1.3. Функціональність інформаційної системи, її переваги та недоліки

Програмне забезпечення для інформаційної підтримки діяльності поліклініки повинно мати широкий спектр функцій для забезпечення комплексного управління медичним закладом. Основні функції включають: Управління даними пацієнтів: Електронні медичні картки: зберігання особистих даних пацієнтів, історії хвороб, результати обстежень та лікування. Доступ до інформації: швидкий доступ для медичного персоналу до електронних карток пацієнтів. Аналіз та обробка даних: можливість аналізу медичних даних для покращення лікувального процесу. Управління прийомами та розкладом роботи лікарів: Онлайн-запис на прийом: пацієнти можуть самостійно записуватись на прийом через інтернет або мобільний додаток. Автоматичне формування розкладу: система автоматично створює розклад прийомів на основі записів. Нагадування про прийоми: автоматичні повідомлення пацієнтам про майбутні візити через SMS або електронну пошту [13].

1. Управління чергами:
    - Електронна система черг. Ефективне керування чергами для зменшення часу очікування пацієнтів.
    - Інформування в реальному часі. Пацієнти можуть відстежувати поточний стан черги через інформаційні екрани або мобільний додаток.
    - Інтеграція з лабораторними та діагностичними системами.
    - Автоматичний обмін даними. Автоматичне отримання та збереження результатів лабораторних і діагностичних досліджень.
    - Швидкий доступ до результатів. Лікарі можуть оперативно переглядати результати обстежень для прийняття медичних рішень.
    - Фінансовий облік та управління ресурсами. Облік наданих послуг, реєстрація та облік медичних послуг, їх вартості та оплат.
    - Формування звітів. Створення звітів про фінансову діяльність поліклініки, витрати та доходи.

2. Захист даних та конфіденційність:
    - Шифрування даних. Використання сучасних методів шифрування для захисту медичної інформації.
    - Контроль доступу. Визначення прав доступу до інформації на основі ролей користувачів.
    - Звіти та аналітика. Аналітичні інструменти: аналіз медичних даних для оцінки ефективності лікування та прийняття управлінських рішень.
    - Створення звітів. Можливість створення різноманітних звітів про діяльність поліклініки.
3. Переваги системи:
    - Підвищення ефективності роботи. Автоматизація рутинних процесів зменшує навантаження на медичний персонал, дозволяючи їм зосередитися на основних обов'язках.
    - Покращення якості медичних послуг. Швидкий доступ до актуальної медичної інформації дозволяє лікарям приймати більш обґрунтовані рішення та надавати високоякісні послуги.
    - Зручність для пацієнтів. Онлайн-запис на прийом та нагадування про візити знижують час очікування та покращують взаємодію між пацієнтами та медичним закладом [9].
    - Централізоване управління даними. Ведення електронних медичних карток дозволяє централізовано зберігати та обробляти всю медичну інформацію, забезпечуючи її доступність та цілісність.
    - Зниження паперової роботи. Перехід до електронної документації зменшує потребу в паперових носіях, що знижує витрати на їх зберігання та обробку.
4. Недоліки – високі витрати на впровадження:
    - Вартість розробки, впровадження та підтримки програмного забезпечення може бути значною, що може бути проблемою для деяких поліклінік з обмеженим бюджетом.

- Необхідність навчання персоналу. Медичний персонал потребує навчання для ефективного використання нових технологій, що потребує додаткових ресурсів і часу.
- Інтеграційні проблеми. Інтеграція нової системи з існуючими може бути складною і вимагати значних зусиль для забезпечення безперебійної роботи всіх компонентів.
- Ризики кібербезпеки. Використання електронних систем підвищує ризики несанкціонованого доступу та кібератак, що вимагає впровадження ефективних заходів безпеки.
- Проблеми з технічною підтримкою. Підтримка та обслуговування програмного забезпечення вимагає наявності кваліфікованих IT-фахівців, що може бути викликом для деяких медичних закладів [1].

Таким чином, розробка програми для інформаційної підтримки діяльності поліклініки має значні переваги, які сприяють покращенню якості медичних послуг та підвищенню ефективності роботи медичного закладу. Водночас існують певні виклики та недоліки, які потребують уваги та обліку при впровадженні таких систем.

1.4. Етапи створення програми для інформаційної підтримки діяльності поліклініки

Розробка програми для ІПД поліклініки – це важливий крок до підвищення ефективності роботи медичного закладу, покращення якості медичних послуг та зручності для пацієнтів [5].

Для досягнення цих цілей процес розробки включає наступні етапи:

1. Планування визначення обсягу проєкту та складання графіку робіт. Оцінка поточних потреб поліклініки.

- Формування технічного завдання. Встановлення термінів для кожного етапу розробки, визначення відповідальних осіб та ресурсів.

3. Аналіз, збір та документування вимог. Виявлення функціональних та нефункціональних вимог, вивчення існуючих систем та процесів в поліклініці. Створення детального технічного завдання.

4. Дизайн інтерфейсу. Розробка дизайну UI та UX) з використанням певних інструментів.

5. Розробка функціональності. Реалізація основних функцій, таких як: управління даними пацієнтів, онлайн-запис на прийом, управління розкладом лікарів та чергами.

5. Інтеграція платіжних систем. Інтеграція платіжних систем для обробки фінансових операцій, забезпечення безпечного процесу оплати медичних послуг.

6. Оптимізація для користувачів та безпеки. Оптимізація продуктивності (бази даних та серверних ресурсів), забезпечення швидкої роботи системи.

7. Захист даних та кібербезпека. Впровадження заходів для захисту конфіденційних медичних даних, регулярне оновлення безпекових політик.

8. Тестування. Перевірка всіх функцій системи на відповідність вимогам, виявлення та виправлення помилок.

9. Запуск проєкту. Підготовка до запуску та старт проєкту.

10. Підтримка продукту. Технічна підтримка, регулярне оновлення та покращення системи, оновлення та розвиток, постійне вдосконалення системи для відповідності сучасним стандартам [6].

Етапи створення програми для інформаційної підтримки діяльності поліклініки забезпечують комплексний підхід до розробки та впровадження програмного забезпечення, що сприяє ефективному управлінню медичним закладом та покращенню якості надання медичних послуг.

1.5. Постановка завдання, опис функціональних та нефункціональних вимог

Метою даної роботи є створення програмного забезпечення для інформаційної підтримки діяльності поліклініки, яке забезпечить автоматизацію ключових процесів, включаючи ведення електронних медичних карток пацієнтів, управління розкладом лікарів, облік прийомів та моніторинг.

Основні задачі, що необхідно вирішити для досягнення поставленої мети:

- Проведення аналізу потреб поліклініки у автоматизації основних процесів;
- Визначення вимог до функціональності програмного забезпечення;
- Розробка простої архітектури системи, що забезпечить надійність і зручність використання;
- Відбір та застосування сучасних архітектурних інструментів для розробки програмного забезпечення;
- Проведення тестування системи;

Система повинна відповідати наступним вимогам. Функціональні вимоги:

- Ведення електронних медичних карток пацієнтів;
- Запис пацієнтів на прийом;
- Облік медичних послуг та процедур;
- Збереження та обробка медичних даних;

Та нефункціональні вимоги:

- Висока продуктивність і швидкість роботи;
- Надійність та відмовостійкість системи;
- Зручний та інтуїтивно зрозумілий принцип роботи;

1.6. Відбір архітектурних інструментів

Для реалізації програмного забезпечення інформаційної підтримки діяльності поліклініки необхідно обрати відповідні архітектурні інструменти. Розглянемо основні з них [10].

Для розробки системи доцільно обрати архітектуру, яка забезпечить гнучкість, масштабованість та надійність системи. Основні переваги мікросервісної архітектури:

1. Масштабованість: Можливість незалежного масштабування окремих компонентів системи.
2. Модульність: Легкість у зміні та розширенні функціональності системи.
3. Відмовостійкість: Збій одного сервісу не впливає на роботу інших.

Технологічний стек - це набір мов програмування, фреймворків і інструментів, які використовуються у розробці. Для реалізації архітектури необхідно обрати відповідний технологічний стек.

- Мова програмування: C++, добре підходить для нашої розробки.
- Фреймворк: Visual Studio 2019 (або Code::Blocks), Notepad++ – забезпечують швидкий старт і велику кількість готових рішень [13].

Для роботи будуть використані бібліотеки, зазначені в таблиці 1.1.

Таблиця 1.1 – Основні бібліотеки

Бібліотеки	Опис
<iostream>	Бібліотека для введення/виведення в C++. Вона забезпечує роботу з потоками вводу-виводу, такими як cin, cout, cerr та clog.
<string>	Бібліотека для роботи з рядками в C++. Вона забезпечує клас std::string, який є зручнішим і більш функціональним для роботи з рядками в порівнянні з масивами символів.

Продовження таблиці 1.1

<conio.h>	Ця бібліотека використовується для роботи з консольним вводом-виводом, включаючи такі функції, як getch(), getche(), kbhit(), clrscr(), cputs(), putch() і т.д. Ця бібліотека є специфічною для компіляторів, таких як Turbo C/C++, і не є частиною стандарту C++

Якщо програма потребуватиме додаткових бібліотек або модулів для роботи з базами даних, мережами, графічним інтерфейсом користувача, то ці бібліотеки можуть бути додані відповідно до вимог вашого проєкту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ ДІЯЛЬНОСТІ ПОЛІКЛІНІКИ

2.1. Структурно-функціональне моделювання процесу

Для нашої програми, яка підтримує діяльність поліклініки, можемо визначити ключові процеси та їх взаємодії. Використаємо діаграму потоків даних

DFD Рівень 0 (Контекстна діаграма) (рис.2.1):

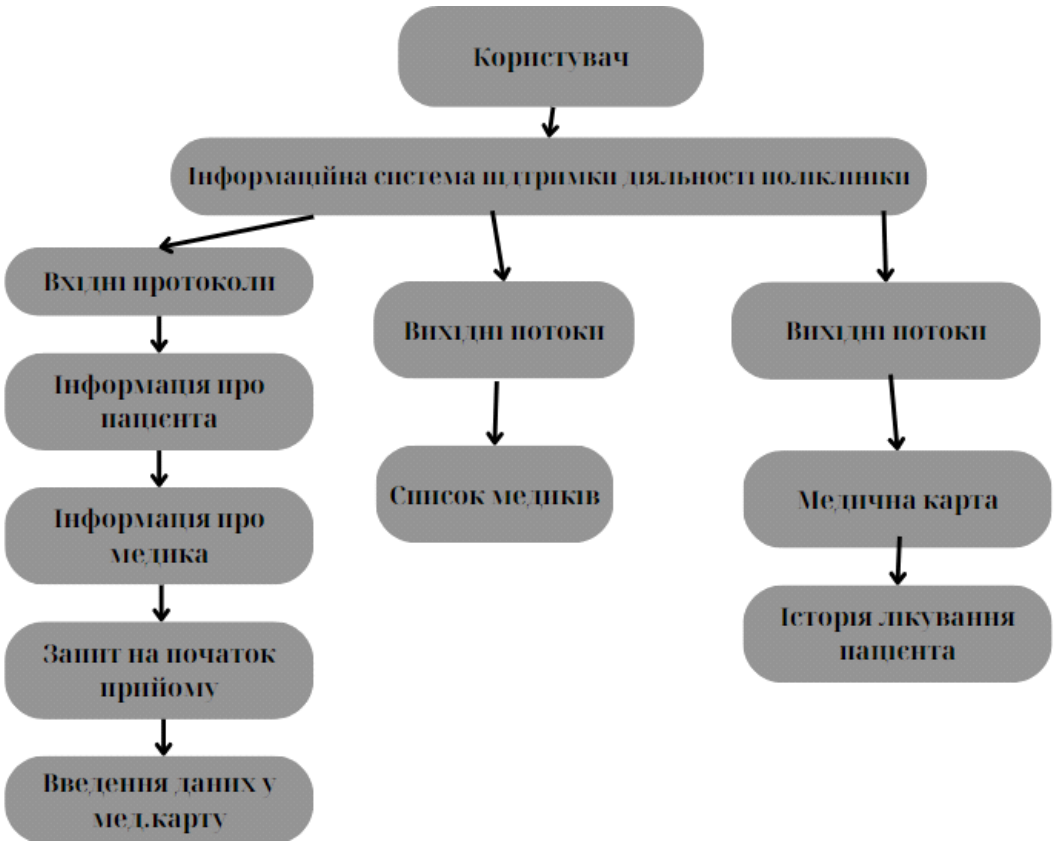

Рисунок 2.1 – Показує загальний вигляд нашої системи, включаючи основні вхідні та вихідні потоки даних

DFD Рівень 1 (рис. 2.2):

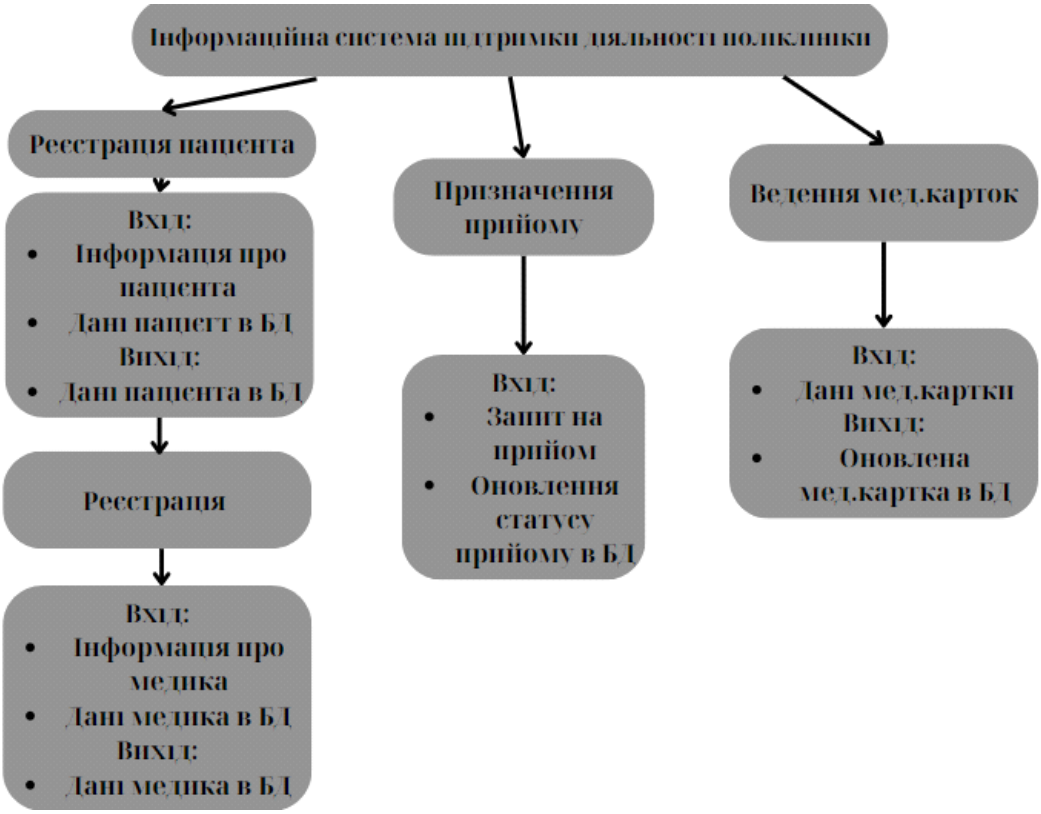

Рисунок 2.2 – Деталізує основні процеси системи, їх вхідні та вихідні потоки

Пояснення до DFD-діаграми (рис. 2.1, рис. 2.2): модель проектування показує, звідки надходять дані, які види діяльності обробляють дані, і чи зберігаються або використовуються вихідні результати іншою діяльністю або зовнішнім суб'єктом.

1. Реєстрація пацієнта:
   - Вхід: Інформація про пацієнта.
   - Вихід: Дані пацієнта в базі даних.
2. Реєстрація медика:
   - Вхід: Інформація про медика.
   - Вихід: Дані медика в базі даних.
3. Призначення прийому:
   - Вхід: Запит на прийом.
   - Вихід: Оновлення статусу прийому в базі даних.

4. Ведення медичних карток:
    - Вхід: Дані медичної картки.
    - Вихід: Оновлена медична картка в базі даних.

Ця діаграма допомагає візуалізувати основні процеси, що відбуваються в системі, та їх взаємодії з користувачами і базою даних.

2.2. Проектування моделі бази даних

Опис сутностей:

1. Пацієнт (Patient):
    - PatientID (ідентифікаційний номер пацієнта);
    - Name (ім'я пацієнта);
    - DateOfBirth (дата народження);
    - Age (вік);
    - ContactInfo (контактна інформація).
2. Медик (Medic):
    - MedicID (ідентифікаційний номер медика);
    - Name (ім'я медика);
    - Specialization (спеціалізація);
    - QueuePosition (позиція в черзі).

3.Медична картка (MedicalRecord):

- RecordID (ідентифікаційний номер запису);
- PatientID (ідентифікаційний номер пацієнта);
- MedicID (ідентифікаційний номер медика);
- Date (дата запису);
- Symptoms (симптоми);
- Diagnosis (діагноз);
- TreatmentStatus (статус лікування: завершено/почато);
- RecordCount (кількість записів у медичній картці);
- TreatmentCost (ціна лікування);
- MedicName (ім'я медика);

- PatientName (ім'я пацієнта).

4. Прийом (Appointment):
    - AppointmentID (ідентифікаційний номер прийому);
    - PatientID (ідентифікаційний номер пацієнта);
    - MedicID (ідентифікаційний номер медика);
    - AppointmentDate (дата прийому);
    - Status (статус прийому: завершено/почато).
5. Зв'язки між сутностями:
    - Пацієнт - Медична картка: Один пацієнт може мати багато медичних карток;
    - Медик - Медична картка: Один медик може обслуговувати багато пацієнтів через медичні картки;
    - Пацієнт - Прийом: Один пацієнт може мати багато прийомів;
    - Медик - Прийом: Один медик може мати багато прийомів [4].

Пояснення до ER-діаграми(рис.2.3): модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків.

1. Пацієнт (Patient):
    - Унікально ідентифікується за PatientID;
    - Має такі атрибути, як ім'я, дата народження, вік і контактна інформація.
2. Медик (Medic):
    - Унікально ідентифікується за MedicID;
    - Має такі атрибути, як ім'я, спеціалізація та позиція в черзі.
3. Медична картка (MedicalRecord):
    - Унікально ідентифікується за RecordID;
    - Містить зовнішні ключі PatientID та MedicID, які посилаються на відповідні сутності;
    - Має такі атрибути, як дата запису, симптоми, діагноз, статус лікування, кількість записів, ціна лікування, ім'я медика та ім'я пацієнта.

4. Прийом (Appointment):
    - Унікально ідентифікується за AppointmentID;
    - Містить зовнішні ключі PatientID та MedicID, які посилаються на відповідні сутності;
    - Має такі атрибути, як дата прийому та статус [9].

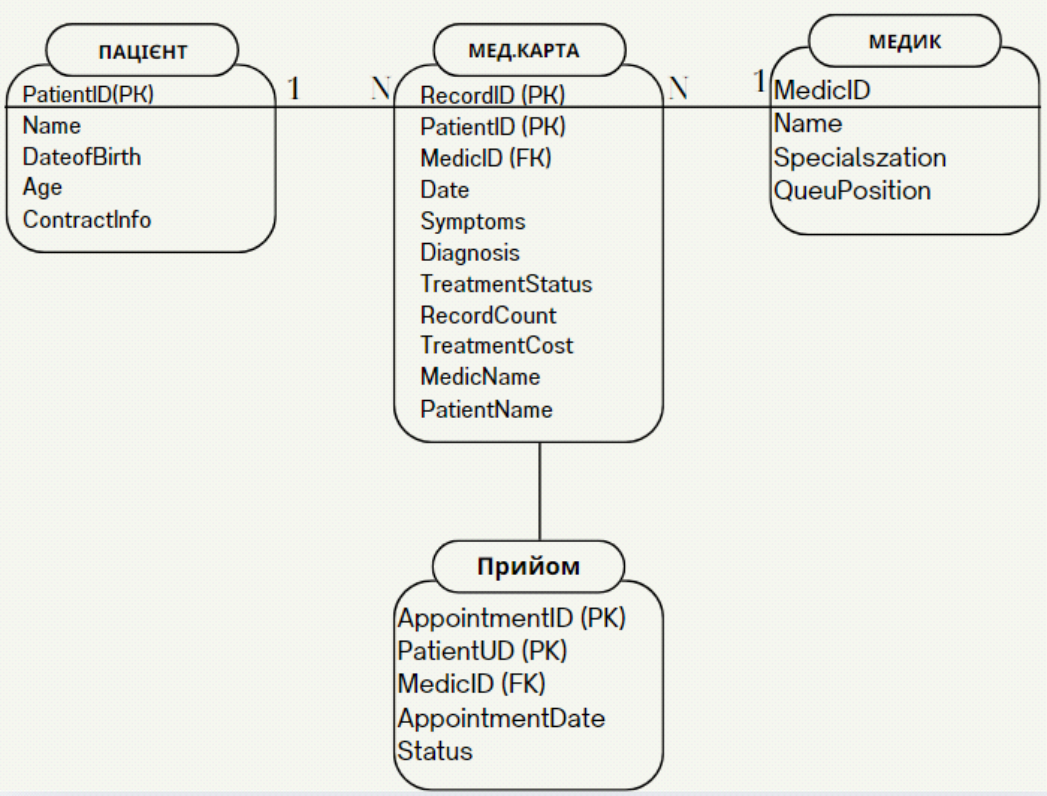

Рисунок 2.3 – ER-діаграма процесів розробленої системи

Ця модель бази даних забезпечує зручне зберігання та доступ до інформації про пацієнтів, медиків, медичні картки та прийоми, необхідні для підтримки діяльності поліклініки.

РОЗДІЛ 3. ПРЕДСТАВЛЕННЯ РЕЗУЛЬТАТІВ СИСТЕМИ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ ДІЯЛЬНОСТІ ПОЛІКЛІНІКИ

3.1. Основні поняття ООП

Клас - визначає абстрактні характеристики деякої сутності, включаючи її атрибути (властивості) та дії (методи). Наприклад, клас "Пацієнт" може мати атрибути, такі як ім'я, вік та медична картка, і методи, як-от запис діагнозу або додавання запису до медичної картки. Класи додають модульність та структурованість в об'єктно-орієнтовані програми і мають бути зрозумілими для медичних працівників. Код класу має бути самодостатнім, а його атрибути та методи разом називаються членами класу [5].

Об'єкт - це конкретний екземпляр класу, створений після запуску програми та ініціалізації полів класу. Наприклад, клас "Пацієнт" описує загальні характеристики пацієнтів, тоді як об'єкт "Сергій" представляє конкретного пацієнта з певними значеннями атрибутів, такими як вік та медична історія. Об'єкти мають стан, що визначається значеннями їх атрибутів. На основі класу "Пацієнт" можна створити інші об'єкти, як-от "Артем", які будуть відрізнятися своїм станом (наприклад, діагнозом).

Метод - можливості об'єкта. Оскільки Сергій - пацієнт, тому він може "отримати діагноз". Тому отримати діагноз є одним із методів цього об’єкта. Він може мати й інші методи, зокрема: отримати запис, або отримати вартість лікування. В межах програми, використання методу має впливати лише на один об'єкт; всі Пацієнти можуть отримати діагноз, але треба щоб діагноз отримав лише певний пацієнт [11].

Обмін повідомленнями - передача даних від одного процесу іншому, або надсилання викликів методів. Наприклад, лікар може викликати метод "отримати діагноз" об'єкта "Пацієнт".

Наслідування - клас може мати підкласи, які є спеціалізованими версіями надкласу. Наприклад, клас "Медичний працівник" може мати підкласи "Лікар" та "Медсестра". Підкласи успадковують атрибути та поведінку свого

батьківського класу і можуть додавати свої власні. Наприклад, підклас "Лікар" може мати додаткові методи, як-от "поставити діагноз".

Інкапсуляція - приховування деталей реалізації класів від об'єктів, що їх використовують. Наприклад, клас "Пацієнт" має метод "отримати діагноз". Реалізація цього методу може включати складні логічні обчислення, які приховані від користувачів класу. Інкапсуляція досягається шляхом визначення доступу до членів класу. Це дозволяє змінювати реалізацію без впливу на інші частини програми. Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт надає кожному іншому класу певний інтерфейс члени, доступні іншим класам.

Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дасть змогу полегшити внесення змін без потреби змінювати і користувачів інтерфейсу.

Індекси двомірних масивів записуються в квадратних дужках і нумерація індексів починається з нуля.

Вказівник - тип даних, який використовується для зберігання адреси змінних і об'єктів, тобто значення вказівника посилається на інше значення, що записане будь-де в пам'яті комп'ютера (фактично містить його адресу).

Вказівники - це ті ж самі змінні, однак їх відмінність від простих змінних полягає в тому, що замість фактичних даних вказівник містить адресу комірки пам'яті, де знаходиться інформація [5].

Наприклад, наступних два рядки створюють два

вказівники на ціле число:

int* pNumberOne;

int* pNumberTwo;

Префікс "p" в обох іменах змінних використовується для того, щоб показати, що змінна є вказівником. Потрібно щоб вказівники

вказували на щось:

pNumberone=&some_number;

pNumberTwo =&some other number;

Знак амперсанд (&) слід розуміти як "адреса чого-небудь", він означає, що буде отримана саме адреса змінної у пам'яті, а не значення самої змінної. Таким чином, у цьому прикладі, у pNumberone зберігається адреса змінної some_number, тому pNumberOne тепер вказує на some_number [9].

Тепер, якщо потрібно звернутися за адресою змінної some_number, можна використовувати pNumberOne. Для такого звертання слід написати *pNumberOne. Кажуть, що оператор (*) розіменовує вказівник. Це значить, що він повертає значення в комірці пам'яті, на яку вказує цей вказівник. За винятком самого оголошення вказівника int * pNumber

3.2. Результати роботи

Продемонструємо роботу створеної програми. Так на рис.3.1 виглядає головне меню:

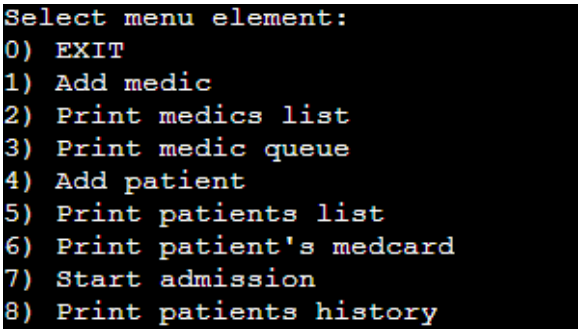

Рисунок 3.1 – Головне меню програми

Додаємо 2 медика та вказуємо потрібну інформацію, у нашому випадку імена та вік. Робимо це у функції Add medic (рис.3.2):

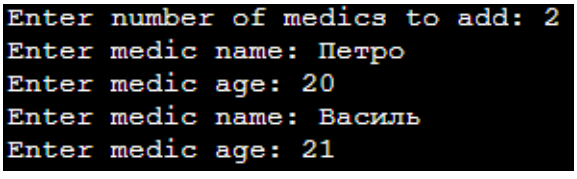

Рисунок 3.2 – Демонстрація доданих даних до функції

Активовуємо вивід списку медиків функцією Print medics list (рис.3.3):

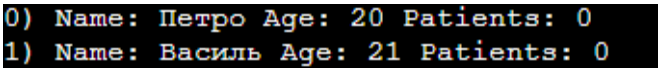

Рисунок 3.3 – вивід списку медиків функцією Print medics list

Додаємо 3х пацієнтів функцією Add patient (рис.3.4):

```
Enter number of patients to add: 3
Enter patient name: Сергій
Enter patient age: 12
```

Рисунок 3.4 – Результат додавання 3х пацієнтів функцією Add patient

У процесі додавання, одразу призначаю, до якого лікаря пацієнт буде записаний (рис.3.5):

```
Fill medcard:
Enter date: 26.06.2024
Enter symptom: нежить
0) Name: Петро Age: 20 Patients: 0
1) Name: Василь Age: 21 Patients: 0
Enter medic id: 0
```

Рисунок 3.5 – Дані запису програми

Виводимо список пацієнтів функцією Print patients medcard (рис.3.6):

```
Name: Сергій  Age: 12 Entries: 1
Name: Артем  Age: 15 Entries: 1
Name: Тарас Age: 20 Entries: 1
Enter patient id: 0
Date: 27.06.2024 Symptom: Нежить Patient: Сергій Medic id: (0)
```

Рисунок 3.6 – Вивід списку пацієнтів на екран

Виводимо чергу записів функцією Print medic queue (рис.3.7):

```
0) Name: Петро Age: 20  Have 0 entry's in journal  Patients: 1
1) Name: Василь Age: 21  Have 0 entry's in journal  Patients: 1
Enter medic id: 1
0) Name: Сергій  Age: 12   Have 1 entry's in medic card Heals in medic (id): 0)
```

Рисунок 3.7 – Демонстрація результату виведення черги записів функцією Print medic queue

Вказуємо діагноз та вартість лікування функцією Start admission (рис.3.8):

```
Enter medic id: 1
Date: 26.07.2024 Symptom: сип Patient: Артем Medic id: (1)
Medic Василь set diagnosis: алергія
Medic Василь set healing cost: 1200
```

Рисунок 3.8 – Результат використання функції Start admission

Після призначення прийому пацієнтів функцією, пацієнт видаляється з масиву пацієнтів у лікаря і додається в масив «історія пацієнтів», який знаходиться у функції Print patients history. Програма працює правильно та стабільно.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1. Поняття й завдання техніки безпеки

Небезпека – це явища, процеси, об'єкти, що за певних умов можуть завдавати шкоди здоров'ю чи життю людини як одразу, так і в майбутньому, тобто викликати небажані наслідки. Джерелами небезпеки є знаряддя праці (інструмент, спеціальні пристрої, машини), сам предмет праці або виробниче середовище. Оточуюче виробниче середовище, зокрема соціальне оточення, можуть стати джерелом психічної травми.

На виробництві, спеціально виділяються роботи з підвищеною небезпекою (із піднімальними кранами, балонами великого тиску, із електромережами високої напруги тощо). Більшість об'єктів підвищеної небезпеки зосереджена в атомній енергетиці, у нафтогазовому, хімічному та нафтохімічному комплексах.

У небезпечних зонах діють або періодично виникають фактори, небезпечні для життя та здоров'я людини. При цьому стан умов праці, за якого виключена дія на працівників небезпечних і шкідливих виробничих факторів, називається безпекою праці.

Об'єктами вивчення техніки безпеки є [10]:

- Технологічний і трудовий процеси;
- Особливості обладнання, інструментів і пристроїв із точки зору безпеки праці;
- Виробниче середовище в цілому, а також його складники (технічні, організаційні, соціальні), які можуть бути причиною виробничих травм чи сприяти їх виникненню й посиленню їхньою дії.

Виділяють наступні завдання техніки безпеки [10]:

- Виявлення причин травматизму, профзахворювань і потенційних небезпек;
- Визначення заходів і технічних засобів, що підтримують безпеку обладнання, а також технологічного і трудового процесу;

- Підготовку й обґрунтування матеріалів для законодавства з техніки безпеки, правил і норм технічних умов, інструкцій із дотримання безпеки експлуатації будівель, споруд, обладнання;
- Проведення повного обліку виробничих травм та аналіз причин їхнього виникнення;
- Вивчення й дослідження наявних технологічних процесів і впровадження нових, більш досконалих, які підтримують безпеку, а також механізацію тяжких і шкідливих робіт;
- Розробку матеріалів та організацію роботи з інструктажу та навчання працівників безпечним прийомам праці.

До необхідності постійно підвищувати рівень безпеки праці керівників підприємств (власників), керівників виробничих підрозділів спонукають три причин:

- Природний людський обов'язок стосовно працівників;
- Мотиви економічного характеру;
- Вимоги відповідних правих норм.

Крім того, нещасні випадки завдають чималих матеріальних збитків, негативно позначаються на добробуті окремої людини, знижують ефективність економіки країни. Від нещасного випадку потерпають люди й виробничий капітал не лише безпосередньо на місці аварії. Аварії та нещасні випадки тягнуть за собою безліч витрат, пов'язаних із відновленням ділянки виробництва, що вийшла з ладу [10].

Для підприємств, які усвідомили значення зусиль із дотримання правил техніки безпеки, спрямованих на гарантування безпечності праці, крім усього іншого, ці зусилля обертаються доволі відчутною економічною вигодою. Багато підприємств підрахували, що витрати на утримання штатних спеціалістів із безпеки праці, навіть фінансово вигідніше, адже все це сприяє помітному зменшенню аварій, нещасних випадків і пов’язаних із ними витрат [10].

4.2. Комп'ютерна техніка та її вплив на організм людини

Комп'ютерна техніка широко використовується в усіх галузях людської діяльності. Людина, яка працює з комп'ютером, постійно перебуває під впливом небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ЗЧ, УВЧ, СВЧ), інфрачервоного й іонізуючого випромінювань, шуму й вібрації, статистичної електрики тощо. Крім цього, оператор піддається значному розумову і психоемоційному навантаженню, високій напрузі зорової та м'язової (робота з клавіатурою ЕОМ) діяльності [11].

Аналіз скарг операторів комп'ютерного набору, редакторів і коректорів, які під час роботи використовують комп'ютер, наведені в таблиці 4.1 [11].

Таблиця 4.1 – Характеристика скарг операторів комп'ютерного набору

Симптоми впливу комп'ютера	Кількість працівників, що повідомили про симптоми, від загальної кількості опитаних, %		
	Стаж роботи		
	До 1 року	1 – 3 роки	3 – 5 років
Біль і різь в очах	58,8	67,5	88,7
Біль у голові	17,6	23,3	42,5
Біль в області спини та шиї	18,5	21,8	32,2
Загальна втома	29,4	25,7	42,6
Утома пам'ять рук	15,1	22,3	38,7
Підвищена роздратованість	11,7	21,6	35,3
Порушення нічного сну	8,3	15,5	20,6
Порушення пам'яті	7,2	12,3	17,1

На функціональний стан користувача комп'ютера впливають: виробниче середовище, трудовий процес, внутрішні та зовнішні засоби діяльності, а також соціально-психологічні фактори. Тому для зменшення ризику захворювань необхідно провадити комплекс медико-гігієнічних, адміністративно-технічних та ергономічних заходів, яких передусім, мають входити [11]:

- Контроль за конструкцією, хорошим станом і функціонуванням комп'ютера;
- Відповідальність місця роботи рекомендацією ергономіки та гігієни;
- Створення оптимальних умов праці у виробничому приміщені (мікроклімат, освітлення, захист від опромінення комп'ютера, іонізація повітря, вентиляція, кондиціонування повітря);
- Раціональний режим праці;
- Підвищувати опір організму користувачів комп'ютерів до дії несприятливих факторів;
- Особиста участь працівників у догляді за своїм здоров'ям.

На превеликий жаль, у багатьох випадках здійснення названих заходів фірмами й індивідуальними користувачами ігноруються, а тому важливо, аби й адміністрація (роботодавець) і користувачі комп'ютерів зрозуміли важливість і необхідність докладання зусиль кожною сторінкою для створення умов, що гарантують працівникам фізичний і духовний комфорт, високу розумову і творчу працездатність, збереження та зміцнення здоров'я [11].

4.3. Шляхи оптимізації технічних, середовищних та ергономічних факторів

Відеодистанційний термінал (ВДТ) – пристрій для візуальної подачі інформації, що зберігається електронним способом. Він складається з дисплейного екрана з виведеними на екран інформацією та клавіатурою [12].

Класифікація ВДТ стосовно впливу на здоров'я базується переважно на конструкторських особливостях та окремих параметрах самого дисплея. Залежно від призначення та сфери застосування ВДТ поділяють на групи [12]:

- Група А – кольорові монітори для демонстрації цілей у навчальному процесі, використання у гральних апаратах, тренажерах, пультах тощо.
- Група Б – кольорові монітори для персональної роботи користувачів у навчальному процесі та господарської діяльності, у якій не потрібна постійна обробка тексту, що містить 80 і більше символів у рядку.

- Група В – кольорові монітори для професійної роботи з текстовими документами й насиченим графічним зображенням у вищих і спеціальних навчальних закладах, ПТУ, спецшколах, на підприємствах, в офісах тощо.
- Група Г – монохромні монітори для забезпечення шкільних комплексів навчальної обчислювальної техніки (старші класи), професійної текстової обробки тощо.

Вимоги до освітлення приміщень, у яких встановленні комп'ютери, насамперед, яким має бути колір стін і підлоги.

Вікна зорієнтовані на південь: стіни зеленувато-голубого кольору; підлога – зелена.

Вікна зорієнтовані на північ: стіни світло-помаранчевого або помаранчево-жовтого кольору; підлога – червонувато-помаранчева.

Вікна зорієнтовані на захід: стіни жовтувато-зеленого або голубувато-зеленого кольору; підлога зеленого або червонувато-помаранчевого.

У приміщені з комп'ютерами необхідно забезпечити наступні величини коефіцієнта відбиття:

- Для стелі – 0,7 – 0,8;
- Для стін – 0,5 – 0,6;
- Для підлоги – 0,3 – 0,5;
- Для інших поверхонь – 0,4 – 0,5.

Доцільно для освітлення використовувати лампи денного (білого) світла разом з лампами теплого білого світла (жовтого, рожевого), що разом імітує колірну гаму, яка відповідає спектральному складу природного світла в сонячний день.

За потреби робочий стіл облаштовують настільною лампою, весь світловий потік якої спрямований на робоче місце, а створене освітлення може бути потік мінімальним для виконуваної праці.

Щоб уникнути потрапляння яскравих променів відбитого світла на екран або в очі користувача, рекомендують, аби кольори апаратури, меблів, одягу персоналу не були світлих тонів (білого, світло-жовтого, світло рожевого).

Відстань від екрану до шкіри обличчя й рцу користувача має бути не меншою 60 см.

Особливо велику небезпеку для здоров’я людей становить підвищена концентрація озону, який уважається не лише подразнюючою, а й канцерогенною речовиною. Відповідно до ДСТУ 12.1 005-88 уміст озону в повітрі робочої зони не повинен перевищувати 0,1 мг/м^3 ; уміст окисів азоту — 5 мг/м^3 , уміст пилу – 4 мг/м^3.

Рівень шуму на робочому місці математиків-програмістів та операторів відеоматеріалів не має перевищувати 50 дБА, а в залах обробки інформації на обчислювальних машинах – 65 дБА.

Рівень вібрації у приміщеннях обчислювальних центрів можна знизити, установивши обладнання на спеціальні фундаменти та віброізолятори.

Максимальна напруженість електричної складової електромагнітного поля на конусі дисплея. Допустимі значення параметрів неіонізованих електромагнітних випромінювань від монітора комп'ютера наведені в табл. 4.1.

Таблиця 4.1 – Допустимі значення параметрів неіонізуючих електромагнітних випромінювань

Назва параметра	**Допустимі норми**
Напруженість електромагнітного поля за електричним складником на відстані 60 см від поверхні відеомонітора	10 кВ/м
Напруженість електромагнітного поля за електромагнітним складником на відстані 50 см від поверхні відеомонітора	0,3 А/м
Напруженість електростатичного поля не має перевищувати для дорослих користувачів. Для дітей дошкільних установ учнів і студентів	20 кВ/м 15 кВ/м

ВДТ – джерело електростатичних зарядів. Тривале перебування в електричному полі, створеному цими зарядами, негативно впливає на здоров'я

працюючих: бронхо-легеневі захворювання, порушення серцево-судинної та нервової систем, ураження шкіри тощо.

Відповідно до ДНАОП 0.00-1.31-99 поверхневий електростатистичний потенціал відеотермінала не має перевищувати 500 В.

Щоб уникнути значного напруження поля та захистити від статичної електрики рекомендують:

- Встановити нейтралізатори статичної електрики.
- Підтримувати у приміщені з ВДТ відносну вологість повітря не нижче 45 – 50% (чим сухіше повітря, тим більше електростатичних зарядів); можна для цього використовувати навіть побутове зволоження.
- Застелити підлогу у приміщенні з ВДТ антистатичних лінолеумом і проводити щоденне вологе прибирання.
- Протирати екран і робоче місце антистатичною серветкою або зволоженою тканиною.
- Користувачам бажано носити одяг, особливо першого шару, із натуральних матеріалів.
- Для зняття статичного заряду бажано кілька разів на день мити руки й обличчя, або від часу торкатися, наприклад, батареї центрального опалення.

Перед початком роботи за комп'ютером необхідно пройти обстеження в лікаря-окуліста і, якщо є потреба, скеровувати рефракцію (окуляри, контактні лінзи). Медичне обстеження зору необхідно періодично повторювати, що не допустити розвиток зниження зору [12].

Під час роботи слід дотримуватися нижченаведених правил.

- Коліна мають бути на достатній відстані одне від одного.
- Необхідно, щоб ступні добре спиралися на підлогу чи підніжку.
- Сидіння не повинні стискати стегон.
- Під час сидіння слід випрямитись, витягнутися вгору, перевірити позу: спина має бути у такому положенні, аби можна було сидіти у спокійній позі без зусиль, тобто не перенапрягаючись.

- Голову слід тримати прямо й нахилено трохи до низу; верхня лінія екрана має бути трохи нижче рівня очей; екран не повинен бути у бліках.
- Ліктьовий суглоб має бути на тій же висоті, що і клавіатура: лікті мають щільно прилягати до тулубу (або підтримуватися підлокітниками крісла) при цьому зап'ястя випрямлене.
- Рекомендується під час роботи згадувати про позу й корегувати її: при відчутті м'язового напруження роблять вправи 4 – 9 разів.
- Кожні 50 хвилин роботи (доцільно використовувати будильник, таймер, тощо) улаштовувати 10-хвилинну перерву, під час якої також відповідними вправами знімають хоча б частину м’язової втоми (тіла й очей), покращуючи самопочуття за допомогою спеціального масажу та фізичних вправ.
- За дві години напруженої безперервної роботи відпочинок слід продовжити до 15 хв.; після 4-годинної роботи – до 1 год.; роботу на комп'ютері бажано чергувати з іншими видами діяльності.
- У післяробочий час необхідно, крім загальних фізичних вправ, проводити специфічні вправи для очей.

Велике значення для збереження здоров'я користувачів комп'ютерів має дотримання медичних рекомендацій щодо раціонального харчування й додаткової вітамінізації організму [12]. Із метою розширення функціональних можливостей м'язів ока необхідно застосовувати спеціальні вправи для очей, масаж «життєвих» точок, а також ті загальні фізичні вправи і спортивні ігри, які особливо рекомендовані для підвищення функціональних можливостей очей.

ВИСНОВОК

Метою кваліфікаційної роботи є дослідження різних альтернативних розробок програм для інформаційної підтримки діяльності поліклініки, створення функціоналу для ефективного управління медичними даними та вибір оптимальних програмних технологій.

Виконуючи дану роботу, використовувались принципи ООП, мова С++ та її особливості, як об’єктно орієнтованої мови: розподіл на класи, використання одинарного та множинного наслідування, інкапсуляції, поліморфізму та абстрагування. Проведено детальний огляд аналогічних систем для поліклінік, розглянуто три альтернативних програмних рішення. Описано функціональність кожної системи, їхні переваги та недоліки, а також етапи створення програмного забезпечення для медичних установ.

Порівняно різні технології для розробки даної програми, оцінено їх основні характеристики, вибрано одну з перелічених і порівняних технологій. Детально проаналізовано та описано.

Розроблено детальне функціональне моделювання, яке включає базу даних, кроки процесу реєстрації пацієнтів, управління медичними записами та іншими можливостями. Проєктовано моделі бази даних. Побудовано діаграми для візуалізації процесів реєстрації пацієнтів та управління їхніми даними.

На основі вимог та специфікацій реалізовано функції програми, включаючи реєстрацію та аутентифікацію користувачів, управління медичними записами, чергу прийомів лікарів, облік наданих медичних послуг тощо.

Розроблено простий інтерфейс користувача, який забезпечує зручну й інтуїтивно зрозумілу взаємодію з програмою для медичного персоналу та, при потребі, пацієнтів.

Проведено ручне тестування програми для виявлення помилок та суперечностей, залучаючи 4-ох користувачів. Після тестування внесені корективи, спрощення та вдосконалення для підвищення якості та надійності системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Systematic Literature Rewiew of Heath Information Systems for Healthcre [Електронний ресурс]. Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10094672/ (дата звернення: 06.03.2024).

2. Armstrong, Deborah J. (February 2006).

3. Основні визначення та принципи ООП [Електронний ресурс]. Режим доступу: https://training.epam.ua/ua/blog/275 (дата звернення: 10.04.2024).

4. C++ refence [Електронний ресурс]. Режим доступу : https://en.cppreference.com/w/ (дата звернення: 10.04.2024).

5. "C++ Primer" від Stanley B. Lippman, Josée Lajoie, Barbara E. Moo

6. Larman C. Use-case model: Drawing system sequence diagrams : *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition) :* 2021. 118 с.

7. Методичні матеріали, призначені для дистанційного навчання, з предмету “Об’єктно-орієнтоване програмування”.

8. Stefan R. Davis. C++. 7-ме вид. 2020. 44-49 с.

9. Василюк А. Інтелектуальна система: *вісник Національного університету «Львівська політехніка» Комп’ютерні науки та інформаційні технології №751*: 2013. 373-381 с.

10. Пашков В. І., Жовтяк Ж. Г., Бодня З.К. Поняття й завдання техніки безпеки : *конспект лекцій з дисципліни "Основи охорони праці" :* Харків: ХНАМГ, 2012. 62 с.

11. Пашков В.І., Жовтяк Ж. Г., Бодня З.К. Комп'ютерна техніка та її вплив на організм людини : *конспект лекцій з дисципліни "Основи охорони праці"* : Харків: ХНАМГ, 2012. 52 с.

12. Пашков В.І., Жовтяк Ж. Г., Бодня З.К. Шляхи оптимізації технічних, середовищних та ергономічних факторів : *конспект лекцій з дисципліни "Основи охорони праці"* : Харків: ХНАМГ, 2012. 53 с.

ДОДАТКИ

Додаток А

Лістинг програми:

```
#include <iostream>
#include <string>
#include <conio.h>
//попереднє оголошення класів
class Person;
class Patient;
class Entry;
class Medic;
 //попереднє оголошення функцій
 void copyPatients (Patient *patients, const int size, Patient *newPatients);
void addPatient(Patient *&patients, int &size, Medic *medics, const int mSize);
void getPatientsList(Patient *patients, const int size);
void deletePatient(Patient *&patients, int &pSize, const int pld, Medic *medics, const int mld);
void copyMedics (Medic *medics, const int size, Medic *newMedics);
void addMedic(Medic *&medics, int &size);
void getMedicsList(Medic *medics, const int size);
void compileMedicPatients (Patient *patients, const int pld, Medic *medics, int const mld);
void startAdmission (Patient *&patients, int &pSize, Patient *&patientsHistory, int &pHSize,
Medic *medics, const int mSize);
void addPatientToHistory(Patient &patient, Patient *&patHistory, int &pHSize);
using namespace std;

                                /*__________CLASSES__________*/

// клас Запис, з якого складається Медкарта пацієнта
class Entry
{
protected:
// дата, симптом, ім'я медика що лікує, ім'я пацієнта, діагноз string date, symptom, medicName, patientName, diagnosis;
//ідентифікаційний номер медика, що лікує
int medicId;
// булева змінна, яка визначає, чи прийом закінчено
bool complete = false;
public:
//встановити інформацію про запис
void setData(string_patName, string_date, string_symptom, int_medicId) {
patientName =_patName;
date=_date;
symptom =_symptom;
medicId=_medicId;
}
//встановити діагноз
void setDiagnosis(string_diagnosis) { diagnosis =_diagnosis; }
//встановити, що прийом закінчено
```

```
        void setComplete (bool value) { complete = value; }
        //вивести інформацію про запис
        string getInfo() {
        if (complete)return "Date: " + date + " Symptom: " + symptom +" Patient: " patientName +" Medic id: (" + to_string(medicld) + ")" + " Diagnosis: "+ diagnosis;
        else return "Date: " + date + " Symptom: " + symptom + " Patient: " + patientName + " Medic id: (" + to_string(medicld) + ")";
        }
        //класи пацієнт та медик мають доступ до приватних полів цього класу
        friend class Patient;
        friend class Medic;
        };
        //базовий клас Персона
        class Person
        {
        //захищені поля, доступ до яких є лише в класі-насліднику
        protected:
        string name;
        int age;
        public:
        //встановити/вивести ім'я, вік
        void setName(string_name)
        {
        name = _name;
        }
        string getName()
        {
        return name;
        }
        void setAge(int_age)
        {
        age = _age;
        }
        int getAge()
        {
        return age;
        }
        };
        //клас пацієнт, який наслідує Персону
        class Patient: public Person
        {
        protected:
        //статична змінна кількості пацієнтів
        static int count;
        //к-сть записів в медкарті, ідентиф. номер, ід. номер медика що лікує, ціна лікування
        int medSize = 0, id, medld, healCost;
        //змінна "пацієнт лікується"
        bool isHealing = true;
        // динамічний масив медичної карти
        Entry *medcard = new Entry[medSize];
        public:
        void setInfo(string _name, int _age)
        {
```

```
name = _name;
age = _age;
id = count;
count++;
}
//встановити діагноз у останній запис медичної карти
void setDiagnosis (string_diagnosis)
{
medcard[medSize - 1].setDiagnosis(_diagnosis);
medcard[medSize - 1].setComplete(true);
}
// додати запис в кінець медкарти
void addEntryToMedCard(Medic *medics, const int size) {
// створюємо новий динам. масив на 1 більший за попередній
Entry *newMedcard = new Entry[medSize + 1];
string_date, _symptom;
cout << "Enter date: ";
cin >> _date;
cout << "Enter symptom: ";
  }
}
};
int Patient::count = 0;
// клас медик, який наслідує персону
class Medic: public Person {
protected:
static int count;
//розмір журналу, черги пацієнтів, ід
int jsize = 0, pSize = 0, id;
//динамічні масиви журнал, і черга пацієнтів
Entry *journal = new Entry[jSize];
Patient *patientsQueue = new Patient[pSize];
public:
// додати пацієнта до черги
void addPatientToQueue(Patient &patient) {
Patient *newPatientsQueue = new Patient [pSize + 1];
//копіювати пацієнтав з оригінального дин. масиву в новостворений
copyPatients(patientsQueue, pSize, newPatientsQueue);
//записати в останній елемент черги пацієнта, який приймає ф-ція
newPatientsQueue[pSize] = patient;
pSize++;
delete[] patientsQueue;
patientsQueue = newPatientsQueue;
}
//видалити пацієнта за черги
void deletePatientFromQueue(int pld) {
Patient *newPatientsQueue = new Patient [pSize - 1];
//пройтись по масиву, якщо ід пацієнта == ід вказаного в аргументі ф-ції ->
пропустити копіювання в новий масив
for (int i = 0, j = 0; i < pSize; i++) {
if (patientsQueue[i].getId() != pld) {
newpatientsQueue[j] = patientsQueue[i];
j++;
}
```

```
        }
        psize--;
        delete[] patientsQueue;
        patientsQueue = newPatientsQueue;
        }
        void setInfo(string_name, int_age) {
        name=_name;
        age=_age;
        id = count;
        count++;
        }
        //встановити пацієнту діагноз
        void setDiagnosis(Patient &patient) {
        string_diagnosis;
        int_healCost;
        //вивести інфо про останній запис у пацієнта
        cout << patient.getEntryFromMedcard(patient.getEntrysCount()-1)<<endl;
        cout << "Medic " << name << " set diagnosis: ";
        cin >>_diagnosis;
        patient.setDiagnosis(_diagnosis);
        cout << "Medic " << name << " set healing cost: ";
        cin >> _healCost;
        patient.setHealCost(_healCost);
        }
        //вивести інфо про пацієнтів з черги по
        void getPatientinfoFromQueue() {
        for (int i = 0; i < pSize; i++)
        cout << patientsQueue[i].getInfo() << endl;
        }
        string getInfo() {
        return to_string(id) + ") Name: " + name + " Age:"+to_string(age)+" Have
"+to_string(jSize)+ "entry's in journal" +" Have" + to_string(pSize) + " patients";
        }
        };
        int Medic::count = 0;

                                /*_____FUNCTIONS_____*/

        //копіювати пацієнтів з одного масиву в інший
        void copyPatients(Patient *patients,const int size, Patient *newPatients) {
        if (size>0){
        for (int i=0; i < size; i++){
        newPatients[i] = patients[i];
            }
          }
        }
        //додати пацієнта в дин. масив в main
        void addPatient(Patient *&patients, int &size, Medic *medics, const int mSize) {
        string name;
        int age;
        patient *newPatients = new Patient[size + 1];
        copyPatients(patients, size, newPatients);
        cout << "Enter patient name:";
        cin >> name;
```

```
cout << "Enter patient age:";
cin >> age;
newPatients[size].setinfo(name, age);
system("cls");
cout << "Fill medcard:\n";
newPatients[size].addEntryToMedCard(medics, mSize);
size++;
delete[] patients;
patients = newPatients;
}
//вивести список пацієнтів з дин. масиву в main
void getPatientsList(Patient *patients, const int size) {
//якщо розмір масиву == 0 - пропустити виведення
if (size == 0)cout << "Patients list is clear" << endl;
else {
for (int i = 0; i < size; i++) {
cout << patients[i].getInfo() << endl;
 }
}
}
//видалити пацієнта з дин масиву в main
//працює як Medic::deletePatientFromQueue()
void deletePatient(Patient *&patients, int &pSize, const int pld, Medic *medics, const int mld)
{
Patient *newPatients = new Patient [pSize - 1];
for (int i = 0, j = 0; i < pSize; i++) {
if (patients[i].getId() != pld) {
newPatients[j] = patients[i];
j++;
}
else {
medics[mld].delete Patient FromQueue(i);
}
}
pSize--;
delete[] patients;
patients = newPatients;
}
//копіювати медиків з одного масиву в інший
void copyMedics (Medic *medics, const int size, Medic *newMedics) {
if (size > 0) {
for (int i = 0; i < size; i++) {
newMedics[i] = medics[i];
 }
}
}
// додати медика в дин. масив в main
void addMedic(Medic *&medics, int &size) {
string name;
int age;
Medic *newMedics = new Medic[size + 1];
copyMedics(medics, size, newMedics);
cout << "Enter medic name: ";
```

```
cin >> name;
cout << "Enter medic age: ";
cin >> age;
newMedics[size].setInfo(name, age);
size++;
delete[] medics;
medics = newMedics;
}
//вивести список медиків
void getMedicsList(Medic *medics, const int size) {
if (size == 0) cout << "Medics list is clear" << endl;
else {
for (int i = 0; i < size; i++) {
cout << medics[i].getInfo() << endl;
   }
  }
}
//додати в чергу медика Medic::patientsQueue пацієнта
void compileMedicPatients(Patient *patients, const int pld, Medic *medics, const int
mid)
{
medics[mld].addPatientToQueue(patients[pld]);
}
//почати прийом пацієнтів лікарями
void startAdmission(Patient *&patients, int &pSize, Patient *&patientsHistory, int
&pHSize,
Medic *medics, const int mSize)
{
if (pSize != 0) {
for (int i = 0; i < pSize; i++) {
system("cls");
//взяти у пацієнта з дин. масиву по індексу і ід медика, що лікує
// у медика з дин. масиву медиків, по індексу з вищевказаної
операції викликати функцію setDiagnosis
//В patientsQueue передати ід теперішнього пацієнта
medics[patients[i].getMedicld()].setDiagnosis(patients[i]);
// додати пацієнта до історії пацієнтів (дин масив в main) addPatient To
History(patients[i], patientsHistory, pHSize);
//видалити пацієнта з дин масиву черги у медика
deletePatient(patients, pSize, i, medics, patients[i].getMedicld());
cout<<<<< endl;
}
}
else {
cout<<"Patients list is clear" << endl;
_getch();
}
}
// додати пацієнта до історії пацієнтів (дин масив в main)
void addPatientToHistory(Patient & patient, Patient *& patHistory, int & pHSize)
{
Patient *newPatHistory = new Patient[pHSize + 1];
for (int i = 0; i < pHSize; i++) {
compileMedicPatients(patientsQueue, patSize - 1, medics,
```

```
patientsQueue[patSize - 1].getMedicId());
}
temp = 0;
goto mark1;
case 5:
system("cls");
getPatientsList(patientsQueue, patSize);
_getch();
goto mark1;
case 6:
system("cls");
getPatientsList(patientsQueue, patSize);
cout << "Enter patient id: ";
cin >> temp;
for (int i = 0; i < patientsQueue[temp].getEntrysCount(); i++)
cout << patientsQueue[temp].getEntry From Medcard(i) << endl;
_getch();
goto mark1;
case 7:
system("cls");
startAdmission(patientsQueue, patSize, patientsHistory, patHisSize, medics, medSize);
goto mark1;
case 8:
system("cls");
cin>>_symptom;
//вивести список медиків
get MedicsList(medics, size);
cout<< "Enter medic id: ";
cin >> medId;
//встановити інформацію у останній запис в новому дин. масиві
newMedcard[medSize].setData(name,_date,_symptom, medId);
//збільшити розмір оригінальної змінної розміру медкарти на 1 medsize++;
//видалити оригінальний масив медкарти з пам'яті
delete[] medcard;
// замінити вказівник на розміщення в пам'яті дин. масиву на вказівник новоствореного масиву newMedcard
medcard = newMedcard;
}
//повернути симптом з останнього запису в медкарті
string getSymptom() { return medcard [medSize].symptom; }
//встановити стан лікування лікується/лікування закінчено
void setHealing(bool value) {isHealing = value;}
//встановити ціну лікування для пацієнта
void setHealCost(float _healCost)
{
healCost=_healCost;
}
//повернути ціну лікування
float getHealCost()
{
return healCost;
}
//повернути ід
int getId() { return id; }
```

```
//повернути ід медика, що лікує цього пацієнта
int getMedicld() { return medld;}
//повернути розмір медкарти
int getEntryscount() { return medSize; }
// повернути інформацію про пацієнта
string getInfo() {
if (isHealing)//якщо лікування не закінчено
return to_string(id) +") Name: " + name +" Age: " + to_string(age) + Have" + to_string(medSize) + " entry's in medic card" + " Heals in medic (id): " + to_string(medld);
else
return to_string(id) + ") Name: " + name + " Age: " + to_string(age) + "
Have diagnosis: " + medcard[medSize - 1].diagnosis +" Healing cost: " + to_string(healCost);
}
//повернути інформація з мед карти по індексу
string getEntryFromMedcard (int i) {
if (i >= 0 && i<medSize) {
return medcard[i].getInfo();
temp = 0;
//перейти в початок вивести меню
goto mark1;
case 2:
system("cls");
getMedicsList(medics, medSize);
// очікувати дії користувача (наприклад натискання клавіші на клавіатурі)
_getch();
goto mark1;
case 3:
system("cls");
getMedicsList(medics, medSize);
cout << "Enter medic id:";
cin >> temp;
medics[temp].getPatientInfoFromQueue();
_getch();
goto mark1;
case 4:
system("cls");
cout << "How many?: ";
cin >> temp;
for (int i = 0; i < temp; i++) {
system("cls");
addPatient(patientsQueue, patSize, medics, medSize);
newPatHistory[i] = patHistory[i];
}
patient.setHealing(false);
newPatHistory[pHSize] = patient;
pHSize++;
delete[] patHistory;
patHistory = newPatHistory;
}
                                        /*_______MAIN_____*/
int main()
{
```

```
//розмір масиву пацієнтів, медиків, історії пацієнтів, змінна вибору в switch, тимчасова змінна
int patSize = 0, medSize = 0, patHisSize = 0, value, temp;
// масиви медиків, пацієнтів, та журналу історії пацієнтів
Medic *medics = new Medic[medSize];
Patient *patientsQueue = new Patient [patSize];
Patient *patientsHistory = new Patient [patHisSize];

//меню
mark1:
system("cls");
cout << "Select menu element: \n"

<< "0) EXIT\n"
<<"1) Add medic\n"
<< "2) Print medics list\n"
<< "3) Print medic queue\n"
<< "4) Add patient\n"
<< "5) Print patients list\n"
<< "6) Print patient's medcard\n"
<< "7) Start admission\n"
<< "8) Print patients history"
<< endl;
cin >> value;
//прийняти значення value
switch (value)
{
//перевірити value на значення, і відповідно до нього виконати частину коду
case 0:
break;
case 1:
//очистити консоль
system("cls");
cout << "How many?: ";
cin >> temp;
for (int i = 0; i < temp; i++) {
addMedic(medics, medSize);
}
getPatientsList(patientsHistory, patHisSize);
_getch();
goto mark1;
}
//видалити масиви з пам'яті комп'ютера
delete[] medics;
delete[] patientsQueue;
delete[] patientsHistory;
return 0;
```