МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра фундаментальної та прикладної математики

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ СЛАБОСТУКТУРОВАНИХ СКЛАДНИХ СИСТЕМ»**

Виконала: студентка 2 курсу, групи 8.1132
спеціальності 113 Прикладна математика
(шифр і назва спеціальності)

освітньої програми Прикладна математика
(назва освітньої програми)

А.О. Зінченко
(ініціали та прізвище)

Керівник доцент кафедри фундаментальної та прикладної математики, к.ф.-м.н., Кондрат'єва Н. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к. т. н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра фундаментальної та прикладної математики

Рівень вищої освіти магістр

Спеціальність 113 Прикладна математика
(шифр і назва)

Освітня програма Прикладна математика

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Гребенюк С.М..
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ СТУДЕНТЦІ

Зінченко Алісі Олександрівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Математичне моделювання слабоструктурованих складних систем

керівник роботи Кондрат’єва Наталія Олександрівна, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 27.11.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Збір даних.
3. Математичне моделювання слабоструктурованих складних систем.
4. Перевірка працездатності програмного продукту.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.05.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.05.2023	
2.	Збір вихідних даних.	15.05.2023	
3.	Обробка методичних та теоретичних	05.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	09.10.2023	
5.	Розробка третього розділу.	24.10.2021	
6.	Оформлення та нормоконтроль	10.11.2021	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	12.12.2023	

Студент _____ А.О. Зінченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.О. Кондрат’єва
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О. Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Математичне моделювання слабоструктурованих складних систем»: 70 с., 21 рис., 27 джерел, 1 додаток.

СКЛАДНА СИСТЕМА, СЛАБОСТРУКТУРОВАНА СИСТЕМА, МОДЕЛЬ, МЕТАСИСТЕМА, СИСТЕМНА ЗАДАЧА.

Об’єктом дослідження є слабоструктуровані складні системи.

Мета роботи – побудова та дослідження моделі метасистеми слабоструктурованої складної системи.

Метод дослідження – аналітичний.

Кваліфікаційна робота нараховує 3 розділи. У першому розділі викладено загальний контекст і проблематику дослідження. Наведений розвиток системної методології дослідження складних систем. Надані основні характеристики складних систем та слабоструктурованих систем. Представлені напрямки системних досліджень. Проведено аналіз підходів до математичного моделювання складних систем. У другому розділі надано теоретичний огляд основних концепцій і підходів, пов'язаних з досліджуваною темою. Проаналізовано наукову літературу та публікації, що стосуються досліджуваної проблематики. Наведена концептуальна схема дослідження складних слабоструктурованих систем за Дж. Кліром. У третьому розділі проведено алгоритмізацію та автоматизаціію процесу дослідження слабоструктурованих складних систем. Описано етапи розробки програмного продукту, наведено інструкцію для кінцевого користувача з використання програми, а також проведено обчислювальний експеремент за розробленим програмним продуктом на конкретному прикладі.

SUMMARY

Master’s qualificftion theses « Mathematical Modeling of the Weakly Stuctured Complex Systems »: 70 pages, 21 figures, 27 references, 1 supplements.

COMPLEX SYSTEM, UNSTRUCTURED SYSTEM, MODEL, METASYSTEM, SYSTEM TASK.

Object of the study is weakly-structured complex systems.

Aim of the study is to build and investigate the metasystem model of a weakly-structured complex system.

Method of research is analytical.

The qualifying work consists of 3 sections. The first section presents the general context and research issues. The development of the systemic methodology for studying complex systems is outlined. The main characteristics of complex systems and weakly-structured systems are provided. Directions of systemic research are presented. An analysis of approaches to mathematical modeling of complex systems is conducted. The second section provides a theoretical overview of key concepts and approaches related to the researched topic. Scientific literature and publications related to the research issues are analyzed. A conceptual scheme for studying complex weakly-structured systems by George Jiri Klir is presented. The third section involves the algorithmization and automation of the research process of weakly-structured complex systems. The stages of software development are described, instructions for end-users on using the program are provided, and a computational experiment is conducted with the developed software product on a specific example.

ЗМІСТ

ВСТУП

В процесі розвитку науки, техніки і інформаційних технологій в різних напрямках професійної та наукової діяльності все актуальнішими і активнішими стають спроби вдосконалити використання методів і технологій математичного моделювання складних систем при автоматизації процесу розв’язання поставлених завдань. А саме, розглядаючи цілий ряд проблем, які стосуються, на сам перед, стабільності функціонування, проєктування, управління та прогнозування поведінки складних об’єктів різної фізичної природи, необхідно уніфікувати процес математичного моделювання систем та рішення системних задач, що виникають при їх дослідженні [1-6].

При цьому потрібно зауважити, що традиційні парадигми не можуть адекватно відобразити універсальні емерджентні властивості, притаманні складним системам. Саме тому дедалі частіше застосовують для аналізу останніх методи й моделі фундаментальних наук. Їх поєднання із сучасними інформаційними технологіями та місткими базами даних сприяє глибокому розумінню природи згаданих систем. Можна стверджувати, що потреба у моделюванні складних систем значною мірою зумовила прогрес у галузі інформаційних технологій та програмного забезпечення. Ґрунтуючись на теорії систем і системному підході, математичне моделювання істотно підвищує ефективність процесів моделювання складних об'єктів [3, 5-10]. Математичне моделювання складних систем передбачає розмежування системи, як об'єкта на деякому часовому інтервалі і тих з його властивостей, які становлять підвищений інтерес і будуть включені в процес дослідження.

Математичне моделювання систем та подальше використання моделей, отриманих за допомогою тих чи інших математичних методів досягається при визначенні конкретного класу, до якого відносяться досліджувані системи.

Виділення специфічних ознак досліджуваних систем, є характерним для класичних методів дослідження та необхідним для вибору відповідних засобів моделювання, рішення задачі ідентифікації та системних задач. Поняття слабо структурованої системи базується на понятті складної системи. Складність полягає в багатозначності станів системи, слабкої передбачуваності розвитку, труднощів розуміння процесів, що протікають у них та характеризуються відсутністю чіткої структури, наявністю великої кількості різнорідних елементів та прихованих взаємозв'язків між ними. Вирішення деяких задач, пов'язаних з наявністю якісних характеристик слабо структурованих систем традиційними методами моделювання оказуються не достатньо ефективними, а іноді і не можливими [1,3,6-10].

Отже, виникає необхідність розвитку та впровадження математичних моделей та методів їх дослідження, загальних для слабоструктурованих систем, а також створення програмних продуктів математичного моделювання на основі міждисциплінарного підходу, що ставить своїм завданням виявлення та теоретичний опис закономірностей будови, поведінки, функціонування та розвитку систем.

Об’єктом дослідження в даній роботі є слабоструктуровані складні системи.

Метою роботи є алгоритмізація та автоматизація побудови моделі метасистеми слабоструктурованої складної системи.

Для досягнення поставленої мети необхідно вирішити ряд завдань дослідження:

- провести аналіз сучасні теоретико-методологічні підходи до використання математичних методів до математичного моделювання складних слабоструктурованих систем;
- побудовати математичниі моделі визначених ієрархічних рівнів;
- обґрунтувати необхідність та ефективність розробляємого алгоритму моделі метасистеми;

- розробити комп'ютерні алгоритми, що дозволять проводити автоматизацію процесу математичного моделювання слабоструктурованих систем;
- створити комплекс програм, що дозволяє здійснювати математичне моделювання складних систем за рахунок автоматизації вирішення системних задач;
- згенерувати та видати досліднику рішення природною мовою в досліджуваній ним предметній галузі.

Робота складається з з трьох розділів.

У першому розділі висвітлюються методологічні аспекти дослідження. Розглядається актуальність теми дослідження, розвиток системної методології для аналізу складних систем, аналіз базових понять, таких як поняття та основні характеристики складних систем, слабоструктуровані системи, моделювання та емерджентність. Також розглянуто основні напрямки системних досліджень та огляд підходів до математичного моделювання складних систем.

У другому розділі розкривається концептуальна схема дослідження складних слабоструктурованих систем за методикою Дж. Кліра. Детально описуються етапи формування початкової системи, системи даних, породжувальної системи, структурованої системи та мета-системи.

В третьому розділі розглядається алгоритмізація та автоматизація процесу дослідження слабоструктурованих складних систем. Обґрунтовуються вибір та основні інструменти розробки, включаючи мову програмування та програмне середовище. Також описуються основні етапи створення автоматизованого програмного продукту, архітектура програми. Наведено інструкцію для кінцевого користувача з використання розробленої в роботі програми та проводиться обчислювальний експеремент за розробленим програмним продуктом на конкретному прикладі.

1 МЕТОДОЛОГІЧНІ АСПЕКТИ ДОСЛІДЖЕННЯ

1.1 Обґрунтування актуальності теми

Системний підхід в математичному моделюванні є ключовим для моделювання процесів і явищ. Таку практику орієнтовано на аналіз взаємодій між частинами досліджуваного об’єкта, його організацію й прогнозування поведінки. Незважаючи на широкий спектр моделей досліджуваних процесів і явищ, які є слабоструктурованими системами, автоматизація процесу математичного моделювання потребує доопрацювання в технічних галузях не менше, ніж у гуманітарних та природничих. Ця проблема, безперечно, стає все більш актуальною в сучасному світі, де складні слабоструктуровані системи вирішують значущі задачі у багатьох сферах, включаючи науку, технології, бізнес і багато інших. Недостатня ефективність прийнятих рішень може призводити до серйозних наслідків, включаючи втрати ресурсів, часу і можливостей [1].

Важливо розуміти, що складні слабоструктуровані системи є унікальними у своїй природі, тому традиційні методи математичного моделювання, аналізу й прогнозування поведінки можуть бути непридатними для ефективного вирішення виникаючих питань.

Одиною із ключових проблем можливість збирати і аналізувати дані з цих систем, особливо якщо інформація про принципи роботи і взаємозв'язки елементів в системі відсутня або обмежена.

Забезпечення ефективної роботи і неперервного функціонування таких систем є пріоритетною метою, що вимагає безперервних інновацій та міжгалузевої співпраці. Лише шляхом поєднання зусиль та розробки нових

підходів до моделювання дозволить досягти значущого прогресу у вирішенні складних системних задач.

Використання математичних методів моделювання і аналізу для вирішення одного й того ж завдання може призводити до розходження результатів через різні припущення і набори даних. Це призводить до необхідності розробки загальних методів моделювання, які можуть бути застосовні до складних систем різної фізичної природи і, таким чином, не обмежуються конкретними предметними областями.

Міждисциплінарний підхід до математичного моделювання і застосування обчислювальних методів є важливим, оскільки дозволяє вирішивати задачі, що виникають при моделюванні слабоструктурованих систем і. таким чином, розширяє застосовність отриманих рішень. Завданням такого підходу є розробка ефективних методів та підходів до математичного моделювання, аналізу та прогнозування поведінки складних систем. Отримувані математичні моделі слабоструктурованих систем знаходять застосування в різних областях, від природничих наук і технічних дисциплін до соціальних і гуманітарних досліджень.

Отже, міждисциплінарний підхід включи в себе інтеграцію знань з різних предметних галузей, з використанням широкого спектру математичних і обчислювальних інструментів, а також розробку програмного забезпечення може бути використаним до математичного моделювання об’єктів різної фізичної природи [2].

1.2 Розвиток системної методології дослідження складних систем

Розглянемо історічной розвиток системної методології дослідження складних систем.

Великий внесок в міждисциплінарний підхід дослідження складних систем внесли вчені, такі як Л. Берталанфі, А.А. Богданов, А. Рапопорт, В.Р. Ешбі, К. Боулдінг і Н. Вінер. Їх інтерес до міждисциплінарних аспектів структуральних відношень між елементами систем був значущим у створенні системології Дж. Кліра. Завдяки їх зусиллям сформовані нові підходи та методики до вивчення складних систем, які сприяли виникненню інтегрованого підходу до наукового дослідження об’єктів довільної природи. [2].

Розглянемо більш детально вклад вказаних вчених в розвиток міждисціплинарних досліджень.

Людвіг фон Берталанфі, австрійський біолог і філософ, розглядав системи як взаємопов'язані живі та неживі групи, що дало поштовх для розвитку теорії систем. Його праця "Загальна теорія систем" (1968) визначила основні принципи та підходи до вивчення систем як загальної наукового напрямку.

Александр Александрович Богданов вніс вагомий внесок у розробку концепції "Тектології" – науки про організацію та управління складними системами. Вчений визнавав важливість інтеграції різноманітних наукових дисциплін у єдину системну теорію.

Анатоль Рапопорт спрямовував свої дослідження на розвиток теорії системних понять, вперше введено поняття "зворотного зв'язку" у моделювання складних динамічних систем. Його внесок в галузь системного аналізу в соціальних системах був надзвичайно значущим.

Вільям Росс Эшбі ввів термін "кібернетика" і вніс великий внесок в розвиток концепції "структурної керованої системи", що стала фундаментом для подальшого розвитку кібернетики та теорії керування.

Кеннет Боулдінг вивчав проблеми соціальної динаміки та динаміки систем. Він впроваджував концепцію "трьох світів", яка розділяє соціальні системи на світ фізичних об'єктів, світ символів та світ соціальних систем.

Норберт Вінер розробив ідеї зворотного зв'язку та кібернетичних систем, що стали фундаментом для розуміння автоматичного управління та процесів прийняття рішень. Вчений вніс вагомий внесок у галузь стохастичної математики, сприяючи розвитку моделювання випадкових подій і процесів в складних системах.

Методологія Дж.Кліра [3] відіграє ключову роль у системології, надаючи методологію для аналізу та моделювання складних систем. Основна ідея методології Кліра полягає в ієрархічному підході до математичного моделювання слабоструктурованих систем.

Однією з ключових концепцій методології Дж.Кліра є "структурний аналіз", що допомагає розкрити взаємозв'язки та залежності між різними елементами системи. Цей науковий напрямок, спрямований на формалізацію семантики та логіки загальносистемних понять, визначає ієрархічну класифікацію систем. Важливість системології полягає у здатності створювати класифікації системних задач та методів їх вирішення на комп'ютері, що є актуальним та цінним в контексті сучасних вимог математичного моделювання та прийняття рішень.

Роботи Дж. Кліра [3] зосереджені на дослідженні систем зі структурованим та поведінковим характером. Шляхом абстрагування між епістемологічними рівнями визначеними Кліром, можна моделювати системи реального світу для прогнозування їхнього розвитку та ефективного управління факторами, які на нього впливають.

Методологія Кліра також акцентує увагу на міждисциплінарному підході, оскільки вона може бути застосована до систем різної природи та галузей. Вона надає загальні принципи та методи, які можуть бути корисними при розв'язанні різноманітних задач, від проєктування систем до аналізу їх динамічної поведінки.

Розвиток системних концепцій продовжується і в теперішній час в роботах вітчизняних та зарубіжних вчених [17-24].

В них розглядаються: типи відношень між елементами систем, а не типи складових їх елементів; властивості структури систем, а не властивості їх функцій. Розвиток досліджень у цьому напрямі призвело до формування відповідної методології. Системний підхід дозволяє звести численні специфічні задачі до відносно невеликого класу системних задач з кінцевим числом стандартних методів їх вирішення, орієнтованих використання нових інформаційних технологій

Отже, ідея моделювання та вирішення складних системних задач схожа на побудову маленького світу, де кожен елемент взаємодіє з іншими. Замість того, щоб розглядати окремі об'єкти, необхідно аналізувати їх як цілісність.

1.3 Аналіз базових понять використовуваних при дослідженняі складних систем

Загалом, моделювання визнається як ефективний інструмент для аналізу та вирішення проблем у науці, технології та багатьох інших галузях, де реальні об'єкти або процеси можуть бути складними для безпосереднього вивчення чи розуміння.

Аналіз понять "складна система" та "слабоструктурована система" є ключовим етапом у розумінні математичного моделювання слабоструктурованих складних систем.

Усвідомлення понять допомагає краще розуміти важливість математичного моделювання та розвитку спеціалізованих методів для аналізу та розв'язання задач, пов'язаних зі слабоструктурованими складними системами. Важливим завданням є розробка методів, які можуть адаптуватися до змін

структури та властивостей цих систем для виділення та вирішення системних задач.

1.3.1 Поняття та основні характеристики складних систем

Поняття "складна система" визначається як система, яка включає в себе значну кількість елементів або компонентів, котрі взаємодіють між собою.

Складні системи володіють потенціалом для виникнення неочікуваних та нелінійних змін у своїй поведінці. Вони часто характеризуються тим, що їхня структура не завжди є чіткою та визначеною, і взаємодії між компонентами можуть бути складними та неочевидними.

Складні системи можуть включати різноманітні елементи, які взаємодіють між собою, утворюючи велику мережу взаємозв'язків. Важливою особливістю є їх потенціал для емерджентності, коли властивості системи виникають не на рівні окремих елементів, а на рівні їх взаємодії.

Одним з основних понять при розгляді систем є поняття емерджентності, яке вказує на явище, коли складові частини системи, лише взаємодіючи між собою, створюють нові властивості та патерни, які неможливо передбачити або пояснити, аналізуючи окремі елементи системи. Врахування емерджентності може зробити аналіз систем більш повним і реалістичним, але водночас це може ускладнити прогнозування та управління системами через їхню складність. Далі наведено особливості цього явища [2]:

- може виникати спонтанно;
- розглядається як необхідна глобальна властивість системи, що виникає з локальної взаємодії складових;
- емерджентні властивості виникають на більш високих рівнях організації, ніж просто сума її складових частин;

- аналізуючи компоненти системи окремо, неможливо передбачити або зрозуміти емерджентні явища на рівні всієї системи;
- системи можуть мати властивість самоорганізації, яка призводить до виникнення нових структур та властивостей без зовнішнього втручання.

1.3.2 Слабоструктуровані системи

Слабоструктуровані системи є підтипом складних систем, які характеризуються тим, що вони не мають чіткої формалізованої структури та правил функціонування. Ці системи можуть бути нелинійними та містити різнородні компоненти зі складними взаємодіями між ними.

В силу своєї складності та нелінійності, слабоструктуровані системи часто не мають визначених математичних моделей, які б точно описували їх функціонування.

Системи можуть мати велику кількість компонентів та нестандартну структуру, що ускладнює визначення чіткого підпорядкування елементів і розбиття системи на підсистеми.

Взаємодія між елементами системи може бути нелінійною та включати в себе складні взаємодії, які можуть призвести до непередбачуваної поведінки системи. При об'єднанні різних компонентів системи може виникнути синергетичний ефект, коли ці компоненти взаємодіють і підсилюють один одного, що може призвести до несподіваних результатів [3].

Для систем даного типу характерна відсутність інформації про принципи роботи, властивості та взаємозв'язки елементів. При цьому інформація про дію системи є даними, отриманими шляхом вимірювання (спостереження або визначення) станів її елементів, а правило відображення $y(t) = f(x(t))$ невідомо [11-18].

З цього випливає, що слабоструктуровані системи мають наступні характеристики:

- відсутність даних про закон функціонування системи;
- наявність як кількісних, так і якісніх змінних, що характеризуоть об’єкт дослідження;
- складна організаційна структура;
- складні форми взаємодії;
- наявність синергетичного ефекту.

Традиційні методи математичного моделювання можуть бути не ефективними для вирішення задач з якісними характеристиками систем.

1.3.3 Моделювання як метод дослідження складних слабоструктурованих систем

Моделювання – це процес створення абстракцій або спрощених представлень складних систем чи явищ з метою розуміння, дослідження, аналізу чи вирішення конкретних задач. Цей підхід дозволяє науковцям, інженерам та дослідникам узагальнювати та вивчати реальний світ, використовуючи спрощені моделі, що полегшує їхнє вивчення та взаємодію з ним.

Однією з ключових властивостей моделей є їх адекватність, тобто вони повинні відповідати певним цілям та вимогам, щоб бути корисними у конкретному контексті. Моделі можуть відображати внутрішню структуру об'єктів, взаємодії між їхніми елементами або ж просто їхню поведінку.

Існує розмаїття типів моделей, таких як пізнавальні, які допомагають організувати та представляти знання, або прагматичні, які використовуються для управління та практичних дій. Моделі можуть бути складними та деталізованими, або ж спрощеними, залежно від мети їх застосування [5].

Розуміння моделювання слабоструктурованих складних систем як комплексного поняття є базою для вирішення системних задач та розвитку нових підходів до математичного моделювання.

1.4 Основні напрямки системних досліджень

Слід одразу зазначити, що напрямки системних досліджень створили основи для нового наукового напрямку, відомого як системотехніка, котра визначена як "правила поведінки" для інженерів, що конструюють складні системи, й має за завдання виявлення та описання загальних системних характеристик та розробку методів їх реалізації.

В свою чергу, системотехніка породила системологію, яка об'єднує концепції науки про системи, системну філософії та системотехніку, про які йтиметься далі [4].

Наука про системи виступає як методологічний інструмент, спрямований на усвідомлення та моделювання комплексних взаємодій. Вона фокусується на вивченні та застосуванні системних підходів у фізичних, суспільних і поведінкових науках. Основний акцент полягає на розумінні систем як цілісних сутностей, де вивчається не лише окремі елементи, а й їх взаємодія та взаємозв'язки. Системи розглядаються як комплексні структури. Для аналізу систем та їх взаємодії широко використовуються математичні моделі. Вони дозволяють визначати рівні складності, ізоморфізми та подібності між різними видами систем. Такий підхід дозволяє створювати абстрактні представлення, які допомагають в розумінні загальних принципів. Дослідження в даному напрямку базується на емпіричних даних та спостереженнях. Використовуючи реальні приклади систем з різних областей знань, вчені аналізують, як застосування системного підходу може поліпшити розуміння та розв'язання проблем.

Системна філософія розглядає наукові теорії як частину більшої системи знань. Вона досліджує способи, як різні галузі науки можуть взаємодіяти та взаємоперетинатися, а також вивчає загальні принципи, що лежать в основі різних наукових дисциплін. Ця галузь філософії досліджує загальні принципи та закономірності, що характеризують системи як універсальний клас явищ. Вона спрямована на розуміння та пояснення того, як системи взаємодіють у світі.

Системотехніка ставить перед собою завдання розробки методів та підходів для застосування системних принципів у створенні та ефективному використанні конкретних систем. Важливою частиною цього напряму є розробка експериментів для оцінки та валідації теоретичних концепцій, що, в свою чергу, може включати в себе моделювання, симуляції та практичні експерименти для тестування ідей у реальних умовах.

Прийнято виділяти наступні основні підходи до дослідження систем:

а) підхід, при якому теорія систем розглядається як розширення і узагальнення теорії управління. Авторами цього напряму є такі вчені як М. Месарович, А. Летов, К. Боулдинг;

б) структуралістський підхід - це підхід, в основу якого покладено вивчення структурних характеристик системи, що описують її поведінку при цьому не розглядаються такі функції системи як " лінійність", " стаціонарність", " гладкість". Великий внесок у розвиток цієї теорії внесли такі учені як Норберт Вінер, Уільям Рос Эшбі, Клод Шеннон, А.Н. Колмогоров, Дж. Клір та ін.;

в) лінгвістичний підхід. Суть цього підходу полягає в узагальненні особистого досвіду. Цим підходом займалися такі учені як Р. Якобсон, Р. Барт та ін.

Таким чином, системні дослідження відкривають нові перспективи для розуміння, аналізу та вирішення проблем у різних галузях, сприяючи розвитку

інновацій та вдосконаленню підходів до виділення та розв’язання складних системних задач.

1.5 Огляд підходів до математичного моделювання складних систем

Розглянемо найбільш поширені методи математичного моделювання складних слабоструктурованих систем, надамо переваги та обмеження кожного методу [6].

Найбільш відомими при математичному моделюванні складних слабоструктурованих систем є евристичні методи, що ґрунтуються на експертному досвіді та інтуїції, а не на повній інформації або формальних алгоритмах. Такі підходи використовуються для розв'язання проблем і виявлення розв'язків шляхом аналогій, експертних оцінок та інтуїтивного розуміння ситуації. Наприклад, одним з відомих методів є синтез аналогій, де використовуються аналогії між подібними системами для розв'язання проблем. Однак результати дослідів значною мірою залежать від компетентності експертів, і якщо ті недостатньо кваліфіковані чи допускають помилки, то це може вплинути на результуючу якість. До того ж, для проведення експертних оцінок необхідно виділяти багато часу та ресурсів.

Методи, де використовуються процедури, алгоритми та математичні інструменти визначаються як формалізовані, і включають ряд підоходів. Розглянемо більш детально формалізовані методи, які виукористовують при моделюванні складних слабоструктурованих систем.

Морфологічний аналіз - це метод, який дозволяє досліджувати структуру системи та взаємозв'язки між її складовими частинами, розкладаючи систему на окремі аспекти для отримання більшого розуміння. Може бути корисним інструментом для командної роботи, де різні спеціалісти можуть аналізувати

окремі аспекти однієї задачі. Серед недоліків: суб'єктивність, обмежена точність, неспроможність до врахування динаміки, підвищена складність при обробці великих обсягів даних.

За допомогою комбінаторного аналізу реально виявити існування неймовірної кількості можливих варіацій, що дозволяє враховувати широкий спектр сценаріїв. Окрім того, він є корисним інструментом для аналізу ризиків і альтернативних варіантів. Незважаючи на ці переваги, головним недоліком є те, що може бути складно обрати оптимальний варіант. Нарешті, комбінаторний аналіз може бути надто трудомістким завданням, коли вимагається аналізувати велику кількість параметрів і варіабельних факторів.

В аналізі ієрархій використовується деревоподібна структура для розробки більш складних моделей прийняття рішень. Однією з його переваг є здатність до систематизації та ієрархічного оцінювання критеріїв, що дозволяє враховувати їх важливість на різних рівнях, спрощуючи сам процес. Проте, відповідність оцінок та ваг параметрів може бути суб'єктивною. Це може впливати на об'єктивність результатів. Незважаючи на недоліки, даний метод залишається корисним інструментом у ситуаціях, коли важливо враховувати ієрархічну структуру і вагомість різних критеріїв.

За допомогою статистичних методів доцільно визначати взаємозв'язок між змінними, встановлювати статистичну значущість та проводити кількісну оцінку взаємозв'язку в даних.

Дисперсійний аналіз застосовується для порівняння середніх значень між 3 чи більше групами для виявлення статистично значущих відмінностей. Перевагами є визначення статистичних відмінностей, можливість множинних порівнянь, можливість визначення взаємодії між факторами. Недоліками є вимоги до дизайну дослідження, однорідності дисперсій, нормального розподілу та незалежності даних.

Кореляційний аналіз використовується для виявлення взаємозв'язку між двома або більше змінними. Перевагами кореляційного аналізу є простота використання, зручність інтерпретації результатів, можливість виявити наявність та силу зв'язку між змінними. Недоліки: відсутність висновків щодо причинно-наслідкових зв'язків, вразливість до викидів та великих значень, обмеження на виявлення лінійних зв'язків, не підтверджує причинність зв'язку між змінними.

Параметричні коефіцієнти кореляції – це статистичні міри, які використовуються для визначення ступеня зв'язку між змінними у випадку, коли дані відповідають певним параметричним умовам. Ці коефіцієнти кореляції базуються на припущенні про певний розподіл даних та інших статистичних вимогах. Деякі з найбільш відомих параметричних коефіцієнтів кореляції: Пірсона, Спірмена, Кендала.

Непараметричні показники зв'язку – це статистичні методи для визначення ступеня зв'язку між двома змінними, які не вимагають певних параметричних умов щодо розподілу даних. Ці методи особливо корисні, коли дані відповідають менш суворим умовам або коли немає підстав припускати, що зв'язок є лінійним.

Кореляційне відношення Пірсона (r) є параметричним коефіцієнтом кореляції, який вимірює ступінь лінійного статистичного зв'язку між двома неперервними змінними. Цей коефіцієнт кореляції допомагає визначити, наскільки змінні корелюють одна з одною і у якому напрямку (позитивно чи негативно).

Коефіцієнт кореляції Пірсона дійсний в діапазоні від -1 до +1:

- якщо r = +1, це означає повну позитивну кореляцію, тобто змінні зростають разом ідеально;
- якщо r = -1, це означає повну негативну кореляцію, тобто одна змінна зростає, коли інша спадає і навпаки;
- якщо r = 0, це означає відсутність кореляції між змінними.

Позитивне значення r вказує на те, що змінні зазвичай зростають разом, тоді як негативне значення r вказує на зворотну залежність, де одна змінна зазвичай зменшується, коли інша зростає.

Аналіз рядів – це статистичний процес вивчення та розуміння властивостей послідовних спостережень, які зазвичай подаються у вигляді часових рядів або послідовностей подій. Цей аналіз допомагає виявити тенденції, закономірності, аномалії та інші важливі характеристики даних. Спочатку проводиться збір та підготовка даних, потім візуалізація даних, статистичний аналіз, моделювання рядів, передбачення та стратегії, й наостанок оцінка і контроль.

Математичне програмування допомагає вирішувати задачі оптимізації та приймати оптимальні рішення в складних системах, де існують обмеження на ресурси чи обмеження на змінні. Методи математичного програмування, такі як лінійне програмування, цільове програмування та динамічне програмування, є ефективними і популярними засобами для розв'язання різних оптимізаційних задач. Вони дозволяють максимізувати прибуток, мінімізувати витрати, оптимізувати розподіл ресурсів та багато інших аспектів.

Отже, кожен з розглянутих методів має особливості й обмеження. Тому вибір конкретного методу повинен базуватися на характеристиках системи, доступних даних та меті дослідження. Евристичні методи корисні при відсутності точних даних, а формалізовані методи допомагають розглядати структуру системи. Статистичні методи дозволяють виявляти взаємозв'язки, а математичне програмування – знаходити оптимальні рішення. Залежно від задачі, може вимагатися комбінація різних методів для досягнення результатів в моделюванні та аналізі складних систем.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ СКЛАДНИХ СЛАБОСТРУКТУРОВАНИХ СИСТЕМ ЗА ДЖ. КЛІРОМ

В даному розділі розглядається широке поле системних проблем, серед яких визначальне значення мають інформаційні, реляційні та структурні проблеми. Із загальних позицій виділяються та досліджуються класи систем із певними типами відношень між елементами системи виділеної на досліджуваному об’єкті.

2.1 Концептуальна схема дослідження складних слабоструктурованих систем за Дж. Кліром

Системологія, розроблена Дж. Кліром [3], відзначається особливим підходом до класифікації моделей, й приділяє значну увагу урахуванню нечіткості та неоднозначностей в умовах обмежених даних. Цей підхід є особливо цінним у контексті складних систем, де формальні та статистичні методи можуть виявитися недостатніми.

Системологія Дж. Кліра в поєднанні з інструментами математичного моделювання створює ефективний аналітичний інструмент для аналізу та оптимізації складних систем. Особливо важливим стає врахування не лише внутрішніх взаємодій елементів системи, але й зовнішніх впливів та динаміки середовища, що розширює можливості моделювання для прогнозування та управління системами у різних умовах.

Дж. Клір поділяв можливість моделювання за 2 принципами: структуровані складні системи та метасистеми послідовної дії. Метасистемам в системології Дж. Кліра приділяється особлива увага [3].

Вчений запропонував визначати систему як ієрархічну структуру, де кожен рівень включає підсистеми, які в свою чергу можуть мати власні ієрархії.

Далі детально про рівні ієрархічної структури. Нижчий рівень відображає базовий рівень системного розуміння, тоді як кожен наступний рівень додає нові шари знань і концепцій [7].

Рівень 0 «Початкова система»:

– вибір способу взаємодії з системою залежить від мети дослідження, умов, та наявних знань;
– система може бути направленою (з визначеними вхідними та вихідними змінними) або нейтральною (без такого розділення).

Рівень 1 «Система даних»:

– після додавання даних, система розглядається як система даних;
– включає реальні стани основних змінних при певних параметрах.

Рівень 2 «Породжувальна система»:

– система, яка включає знання про інваріантні параметри відносин між – розглядуваними змінними;
– генерує точні або наближені дані при визначених умовах.

Рівень 3 «Підсистеми загальної системи – структурована система»:

– системи, визначені як породжувальні системи (або іноді системи нижчого рівня), є підсистемами загальної системи;
– підсистеми можуть з'єднуватися або взаємодіяти між собою.

Рівень 4 «Метасистеми»:

– системи, які складаються з набору систем, визначених на рівні 1, 2 або 3, та певних інваріантних параметрів метахарактеристик;
– вони визначають зміни в системах нижчого рівня та потребують, щоб системи нижчого рівня мали спільну вихідну систему.

Рівень 5 «Мета-метасистеми»:

- метахарактеристика може змінювати множину параметрів відповідно інваріантним параметрам характеристик більш високого рівня або мета-метахарактеристикам;
- системи на цьому рівні називаються мета-метасистемами або метасистемами другого порядку.

Такий підхід дозволяє розглядати систему як сукупність взаємопов'язаних елементів, що співпрацюють для досягнення спільної мети. Застосування системології Дж. Кліра передбачає створення математичних моделей для аналізу та оптимізації функціонування системи.

В контексті математичного моделювання слабоструктурованих складних систем системологія Дж. Кліра надає основи для розробки детальних ієрархічних моделей. Ці моделі дозволяють краще розуміти структуру системи та визначати ключові параметри та залежності між ними. Такий підхід відкриває можливості для аналізу та оптимізації функціонування складних систем різної фізичної природи.

В наступних пунктах даного розділу розглянемо більш детально методологію дослідження складних слабоструктурованих систем.

2.2 Формування початкової системи

Побудова початкової системи передбачає виконання ряду етапів. На першому етапі необхідно визначити об'єкт, який стане предметом дослідження. Це може бути як матеріальне, так і абстрактне явище. Якщо є конкретна мета дослідження, необхідно визначити основну проблему, яку необхідно вирішити.

Далі, необхідно розділити об'єкт на категорії відповідно до його природи. Оскільки об'єкт володіє нескінченною кількістю властивостей, наступним кроком є визначення ключових характеристик, які стануть предметом

дослідження. Це дозволить зосередитися на найважливіших аспектах об'єкта і зробити дослідження більш спрямованим. Для кожної обраної характеристики слід встановити процедури вимірювання або спостереження. Таким чином, коли з'являються конкретні характеристики і відповідні процедури вимірювання, можна сказати, що система визначається цим обраним набором властивостей і пов'язаними з ними змінними.

Змінна є способом подання властивості через вимірювання чи спостереження. Кожна змінна має унікальну мітку і є пов'язаною з множиною станів (значення змінної). Подібно до цього, параметр представляє уявлення бази і також має унікальну мітку, пов'язану з параметричною множиною.

Базою зветься оперативне уявлення властивості через конкретну процедуру вимірювання чи спостереження. Якщо розглядати властивості та бази, то передбачається, що різні спостереження однієї змінної різняться за значеннями параметрів. Важливо, щоб кожне значення параметра ідентифікувало одне і тільки одне спостереження змінних.

Крім конкретних змінних та параметрів розглядаються загальні змінні та параметри. Ці абстрактні величини не визначаються через будь-які властивості чи бази. Їх множини станів та параметричні множини, а також відносини на цих множинах, представляються стандартним чином. Загальна змінна отримує інтерпретацію, коли множина її станів відображається ізоморфно в елементи множини станів конкретної змінної. Те ж саме стосується і загальних, і конкретних параметрів та їх параметричних множин.

Канал спостереження є методом, який використовується для проведення вимірювань або спостережень. Він може включати різні техніки, методи вимірювання або засоби спостереження, які використовуються для отримання інформації про властивості об'єкта чи системи.

Початкова система складається за допомогою взаємодії між 3 примітивними системами, котрі включають систему об’єкта, представляючу систему і загальну представляюючу систему.

Систему об’єкта можна узагальнити як:

$$O = (\{(a_i, A_i) \mid i \in N_n\}, \{(b_j, B_j) \mid j \in N_m\}), \quad (2.1)$$

де $N_n = \{1, 2, \dots, n\}$ і $N_m = \{1, 2, \dots, m\}$ – множини значень цілих додатніх чисел від 1 до індексу;

a_i і A_i – властивість та множина її проявів;

b_j – база і множина її елементів.

Представляюча система і загальна представляюча система виглядають як:

$$\dot{I} = (\{(\dot{v}_i, \dot{V}_i) \mid i \in N_n\}, \{(\dot{w}_j, \dot{W}_j) \mid j \in N_m\}), \quad (2.2)$$

$$I = (\{(v_i, V_i) \mid i \in N_n\}, \{(w_j, W_j) \mid j \in N_m\}), \quad (2.3)$$

де сенс елементів той же, що і у системи об’єкта.

З цього випливає, що початкова система є основою для проведення емпіричних вимірювань.

2.3 Формування системи даних

Далі, доцільно дослідити лише загальну представляючу систему даних I. Нехай

$$W = W_1 \times W_2 \times \ldots \times W_m, \quad (2.4)$$
$$V = V_1 \times V_2 \times \ldots \times V_n.$$

Тоді чіткі дані можна описати як дійсний стан змінних при нескінченній параметричній множині:

$$d: W \to V. \quad (2.5)$$

Поєднання системи I з функцією d можна описати як систему даних наступним чином:

$$D = (I, d). \quad (2.6)$$

Замінивши I на S(початкову системи даних), отримаємо систему данних з семантикою:

$$D^S = (S, d). \quad (2.7)$$

Схожим чином можна отримати спрямовані системи даних без семантики і з семантикою:

$$\widehat{D^S} = \left(\widehat{I}, d\right), \quad (2.8)$$
$$\widehat{D^S} = (\widehat{I}, d).$$

2.4 Формування породжувальної системи

Для подальшого дослідження необхідно визначити такий термін як маска, що визначається через змінні, параметричну множину і правила зсуву:

$$r_j: W \to W. \tag{2.9}$$

Для упорядкованної параметричної множини правило зсуву виглядає як:

$$r_j(w) = w + \alpha, \tag{2.10}$$

де α – ціла константа, що може примати значення -1, 0, 1 відповідно.

За необхідності маску можна розбити на підмаски вигляду:

$$M_G = (M, M_g, M_{\bar{g}}), \tag{2.11}$$

де g –породжувані змінні ($g \in G$), $\bar{g}$ – породжуючі змінні.

Породжуюча функція поведінки:

$$f_{GB}(\bar{g}, g) = \begin{cases} 1, \text{якщо } g \text{ і якщо } \bar{g}, \\ 0, \text{якщо } !\, g \text{ і якщо } \bar{g}, \end{cases} \tag{2.12}$$

Породжувальна система з поведінкою – це система, яка породжує об'єкти чи властивості, керуючи цим процесом за допомогою концепції маски. Враховуючи контекст чи умови, вона може генерувати різні види об'єктів або властивостей:

$$F_{GB} = (I, M_G, f_{GB}). \tag{2.13}$$

2.5 Формування структурованої системи

Далі, дослідження здійснюється через формування структурованих систем. Структуровані системи уособлюють спосіб, яким вирішуються завдання взаємодії між частинами та цілим. Підсистеми загальної системи, визначені як множини початкових систем, систем даних, або ж породжувальних систем, взаємодіють, використовуючи загальні змінні. Окрім цього, жоден елемент не повинен бути підсистемою іншого елемента тієї ж системи. Порушення цієї умови неприпустимо.

То ж, структуровані системи визначаються як виразна інтеграція підсистем, де кожна частина виконує унікальну роль в досягненні загальної мети. Взаємодія на рівні загальних змінних надає системам гнучкість і можливість адаптації, підкреслюючи їх ефективність в розв'язанні складних задач.

2.6 Формування мета-системи

Метасистемний підхід є унікальним у своєму роді, так як кардинально відрізняється від традиційного. Його особливість заключається в тому, що інтегрування систем здійснюється за параметричними множинами, не зважаючи на те, чи мають системи ідентичні множини змінних. Більше, перед метасистемами стоїть низка завдань, таких як:

- визначення першим кроком діапазонів найбільш ефективної роботи систем;
- оцінка та поліпшення готовності систем до майбутнього використання;
- визначення та забезпечення взаємодії між всіма компонентами;

– розробка стратегії для одночасного перемикання окремих систем, або навіть цілих груп;

– оптимальне перерозподілення обмежених загальносистемних ресурсів.

Елементи метасистеми не залежать один від одного, їх сполучення під забороною. У будь-який момент часу функціонувати дозволено не всім елементи. Слід зазначати, що також існують багаторівневі метасистеми. До їх класифікації входять лише ті метасистеми, котрі побудовані з двох і більше метасистем рівнем одна над одною.

Як висновок, починаючи з початкової системи і системи даних, розвиток понять відбувається через породжувальні і структуровані системи до метасистем, а згодом й до багаторівневих метасистем. Кожен епістемологічний рівень має свої унікальні характеристики та функції і будується один над одним, утворюючи «піраміду», що мінімізує можливість виникнення похибок в процесі тестування вже на завершальному етапі розробки.

3 АЛГОРИТМІЗАЦІЯ ТА АВТОМАТИЗАЦІЯ ПРОЦЕСУ ДОСЛІДЖЕННЯ СЛАБОСТРУКТУРОВАНИХ СКЛАДНИХ СИСТЕМ. РЕАЛІЗАЦІЯ МАТЕМАТИЧНОЇ МОДЕЛІ

Даний розділ присвячено розгляду реалізації метасистеми послідовної дії за методологією Кліра. Описано етапи розробки програмного продукту, наведено інструкцію для кінцевого користувача з використання програми, а також проведено обчислювальний експеремент за розробленим програмним продуктом на конкретному прикладі.

3.1 Обґрунтування вибору та опис головних інструментів розробки

Підбір доречних інструментів відіграє важливу роль у долі проєкту. Розробка математичної моделі відбуватиметься з використанням мови програмування Python та інтегрованого середовища розробки Visual Studio Code для простоти оптимізації.

3.1.1 Обґрунтування вибору мови програмування

Вибір мови програмування грає важливу роль у процесі виконання наукових досліджень подібного роду. В даному пункті наводиться обґрунтування вибору використовуваної при розробці прикладного програмного забезпечення одного з головних інструментів програмної розробки – мови програмування. В роботі в якості мови програмування обрано мову Python, яка являє собою динамічну інтерпретовану об’єктно-орієнтовану скриптову мову

програмування із строгою динамічною типізацією, має простий синтаксис та дозволяє швидко й ефективно розробляти прикладне програмне забезпечення в різних галузях знань, в тому числі у теорії систем та математичному моделюванні складних систем.

Далі перелічено причини, з котрих мову програмування Python було прийнято за фундамент застосунку.

По-перше, Python – це високорівнева та динамічна мова програмування, що дозволяє зручно реалізовувати складні алгоритми та забезпечує швидку розробку програм. Синтаксис Python є простим і лаконічним, і це полегшує читання та обслуговування коду [7].

По-друге, ключовою перевагою Python є його широка підтримка в галузі наукових обчислень та машинного навчання. Наявність потужних бібліотек, таких як NumPy, Pandas, та багатьох інших, робить Python ідеальним інструментом для аналізу даних та створення моделей.

За роки існування, ця мова програмування оволоділа великою спільнотою розробників, і це робить її ідеальним кандидатом для наукових досліджень. Наявність багатьох онлайн-ресурсів, форумів та документації значно полегшує вирішення задач та виникаючих питань під час розробки.

Крім того, Python є крос-платформеним інструментом. Завдяки цій привілегії можна легко переносити розроблені моделі між різними операційними системами.

Таким чином, обираючи Python для розробки імітаційної моделі, можна розраховувати на універсальність готового продукту.

3.1.2 Обґрунтування вибору програмного середовища

В даному пункті наводиться обґрунтування вибору використовуваного в роботі одного з головних інструментів програмної розробки – програмного середовища спеціального призначення, необхідного для реалізації принципів об'єктно-орієнтованого підходу у розробці програмного забезпечення. В якості такого доречного програмного середовища в роботі обрано середовище Visual Studio Code, яке є потужним IDE, розробленим компанією Microsoft. Це програмне середовище надає розробникам зручні інструменти для написання, тестування та відлагодження коду. Однією з переваг VS Code є його відкритість та розширюваність [8].

VS Code підтримує інтеграцію з багатьма мовами програмування, включаючи Python, яка була обрана для розробки імітаційної моделі. Інтеграція з Git також робить його потужним інструментом для керування версіями та спільної роботи у команді.

Один із факторів вибору VS Code – це його зручний інтерфейс та можливості налаштувань. Він надає велику свободу вибору для налаштування робочого середовища розробника згідно із потребами та уподобаннями.

Для поставленої мети можливість використання VS Code для побудови і відлагодження складних алгоритмів є найбільш вагомим аргументом.

У коді використовується кілька бібліотек, які надають додаткові можливості для обробки даних та математичних операцій. Бібліотека numpy забезпечує підтримку роботи з масивами та матрицями, а також великою кількістю математичних функцій. В свою чергу, бібліотека statistics надає функції для обчислення статистичних характеристик даних, таких як середнє значення. Matplotlib – для створення графіків та виведення результатів аналізу даних. Pandas використана для збереження результатів аналізу даних у структурах даних.

3.2 Основні етапи створення автоматизованого програмного продукту

Розробка моделі оцінки стану серцево-судинної системи з використанням мови програмування Python та інтегрованого середовища розробки Visual Studio Code складається з наступних етапів:

а) ініціалізація функцій з min/max значеннями пульсу, артеріального тиску, ЕКГ, насичення киснем, рівня холестерину та цукру в крові для обчислення нормальних меж медичних параметрів відповідно до 2 параметрів – віку та гендеру;

б) створення класу SystemS1, де здійснюватиметься ініціалізація унікальних ідентифікаторів пацієнта:

- № пацієнта;
- Ф.І.О.
- вік;
- гендер;

в) створення класу SystemS2 та обробка початкових результатів аналізів, що містять таку інформацію:

- початковий пульс;
- початковий артеріальний тиск;
- початкова ЕКГ;
- початкове насичення киснем;
- початковий рівень холестерину;
- початковий рівень цукру в крові;

г) демонстрація результатів пацієнту;

д) створення класу SystemS3, який взаємодіє з SystemS2 та містить інформацію про результати повторних аналізів пацієнта:

- повторний пульс;

- повторний артеріальний тиск;
- повторна ЕКГ;
- повторне насичення киснем;
- повторний рівень холестерину;
- повторний рівень цукру в крові;

е) ініціалізація функції для генерування тестових даних пацієнта;

ж) порівняння результатів середнього суми між SystemS3 і SystemS2 зфункціями min/max та визначення ступеня відхилень від норми(min_value чи max_value);

з) ініціалізація функції для візуалізації даних пацієнта і окремо для автоматичного збереження до файлу формату xlsx;

и) введення для користувача як можливості обрати готові дані, так і вводити свої, зокрема результати початкових та повторних аналіз, даних пацієнта; можливість проведення повторного дослідження для іншого пацієнта до бажання користувача завершити користування програмою.

3.3 Архітектура програми

У розробленій моделі оцінювання стану серцево-судинної системи пацієнта використовується об'єктно-орієнтований підхід, який базується на використанні об'єктів та їх взаємодії для реалізації програм.

Модель реалізовано як метасистему послідовної дії для аналізу медичних показників обраного пацієнта. Програму побудовано у вигляді послідовної логічної структури (системи аналогії), де кожен крок (функція або клас) відповідає конкретному етапу оцінювання.

Оскільки метасистему побудовано на основі методології Кліра, то слід зазначити, що основні властивості досліджуваної системи є заданими семантично.

Семантику закладено і в порівнянні результатів аналізів пацієнта між різними системами (S1, S2, S3) та визначенні відхилення від норми. Це дозволяє отримувати неінтерпритовану інформацію про стан здоров'я пацієнта та необхідність повторного обстеження. Метасистема враховує динаміку стану пацієнта. Зв'язок між системами (S1, S2, S3) показує, як дані про пацієнта прогресують та як змінюється їх стан з часом. Програма дозволяє отримати графічне представлення результатів аналізів у вигляді графіку, що дозволяє легше сприймати та аналізувати отриману інформацію.

Введення компонент (класів): системи аналізу, клас пацієнта, допоміжні функції тощо дозволяє побудувати логічну структуру програмного продукту. Програма організована за логічними кроками, що полегшує розуміння та розробку. Кожен крок відповідає за конкретний етап роботи системи аналізу здоров'я пацієнта. Програма може отримувати інформацію від користувача (з консолі) або використовувати тестові дані, що забезпечує більшу гнучкість. Результати аналізів виводяться на екран, графічно відображаються та зберігаються до окремого файлу.

Перейдемо до структури й послідовності виконання коду. Першим кроком ініціалізовано функції для визначення мінімальних та максимальних значень різних параметрів залежно від віку та гендеру пацієнта. Дані для функцій про оптимальні показники брались з медичного англомовного сайту [9]. У наступному кроці визначено клас Patient, який представляє пацієнта. Також створено клас SystemS1, який містить список пацієнтів і функції для додавання пацієнта та порівняння результатів. Також додано функцію для генерації тестових даних, що базується на основі оптимальних показників з деяким відхиленням між початковими та повторними даними. Третім кроком визначено

клас SystemS2, який використовує результати з SystemS1 і дозволяє додавати результати аналізів для пацієнта. Далі визначено клас SystemS3, який використовує результати з SystemS2 і дозволяє додавати результати повторних аналізів для пацієнта. Після цього визначено клас SystemS4, який порівнює результати між SystemS3 та SystemS2 для пацієнта. Потім визначено допоміжні функції для введення числових значень та даних пацієнта. Додано функцію plot_results, яка будує графік, що підсумовує дані та відображає стан пацієнта. Передостанім кроком визначено функцію choose_input_method, яка дозволяє користувачу вибрати спосіб введення даних. І наостанок визначено функцію main, яка є основною частиною програми. Вона викликає попередньо визначені функції для виконання програми і, до того ж, активує збереження данних про поточного пацієнта до файлу.

3.4 Інструкція для кінцевого споживача щодо використання програми

Інструкція для кінцевого споживача щодо використання програми представляється наступним чином:

а) запуск файлу програми (скрипту) на виконання, використовуючи командний рядок або зручне програмне середовище; автоматично виведено запит про вибір способу введення даних: "консоль" або "тест". Необхідно вибрати "консоль", якщо хочеться ввести дані самостійно, або "тест", якщо хочеться використати тестові дані (див рис. 3.1);

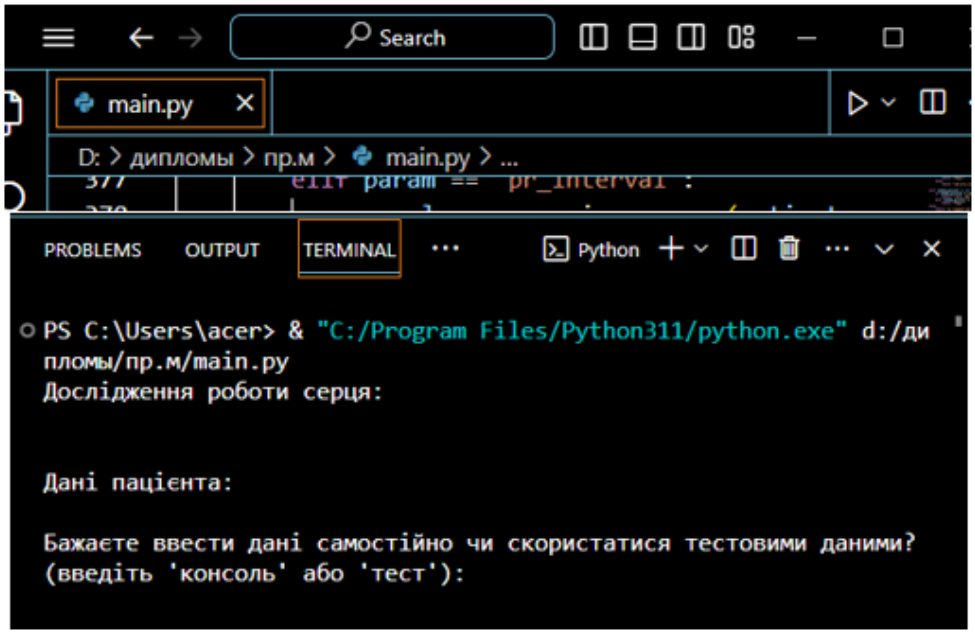


Рисунок 3.1 – Демонстрація запуску

б) якщо вибір пав на "тест", дані будуть згенеровані автоматично для тестування програми (див рис. 3.2);

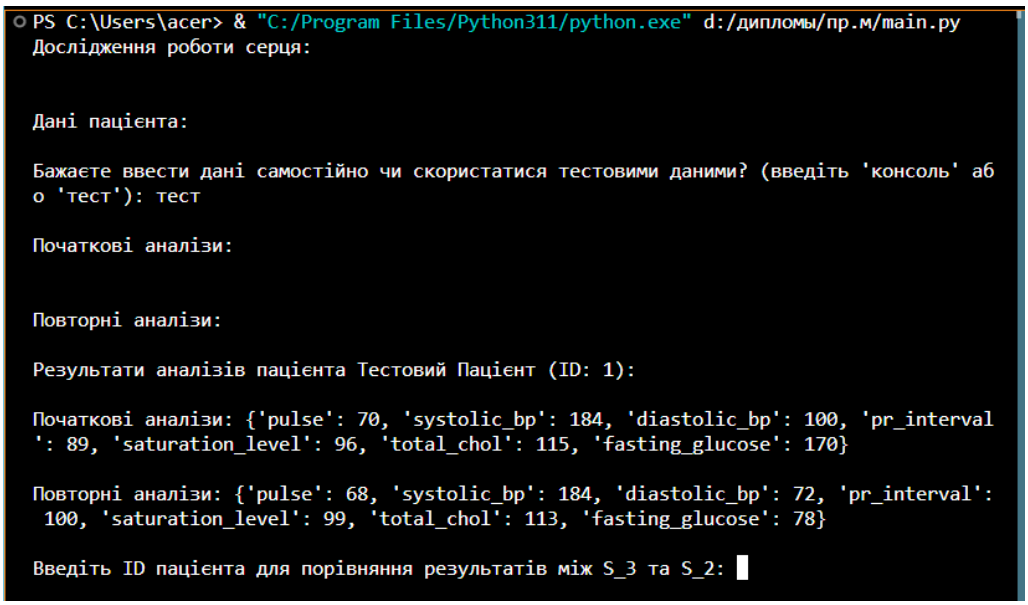


Рисунок 3.2 – Введення «тест» до командного рядка

в) користувачу необхідно ввести ID заданого пацієнта, після чого проводиться автоматичний розрахунок, а під кінець з’являється рядок вибору продовжити використання програми чи завершити (див рис. 3.3, рис.3.4);

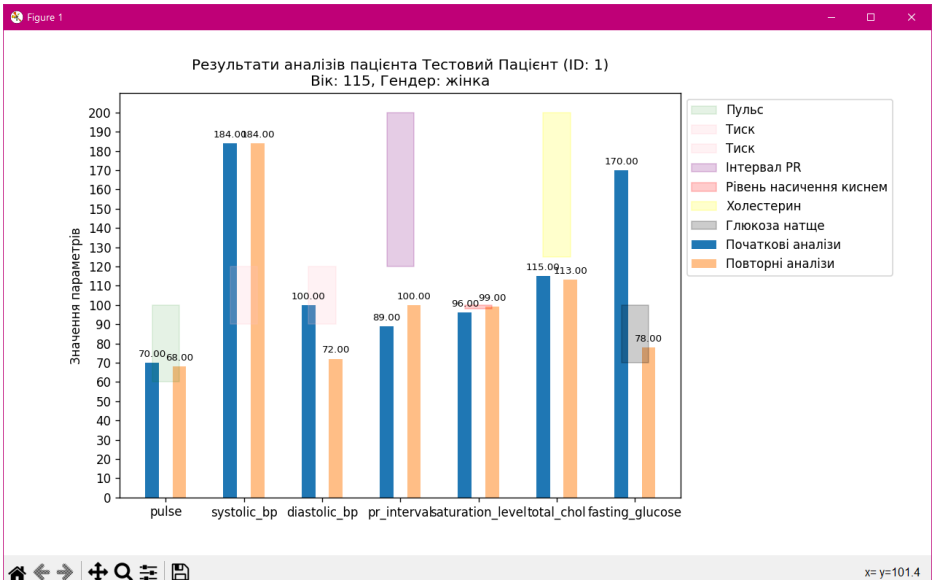


Рисунок 3.3 – Візуалізація даних «тест»

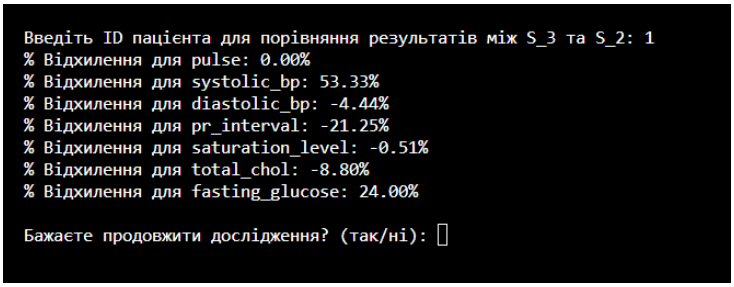


Рисунок 3.4 – Подальший розрахунок «тест»

г) якщо вибір пав на "консоль", необхідно:

1) ввести ID пацієнта, ім'я, вік та гендер (чоловік/жінка) згідно із запитаннями (див рис 3.5);

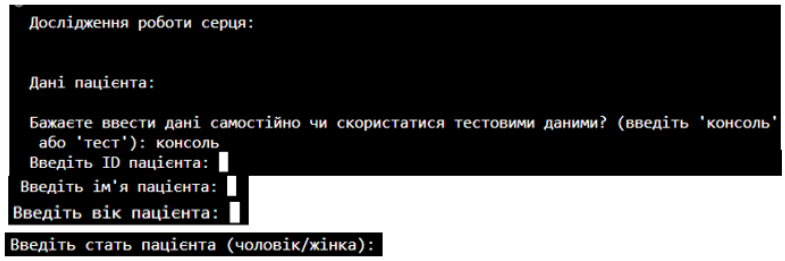


Рисунок 3.5 – Рядки для вводу особистих даних пацієнта

2) ввести значення пульсу, систолічного тиску, діастолічного тиску, інтервалу PR, рівня насичення киснем, загального холестерину, рівня холестерину HDL, рівня холестерину LDL та рівня глюкози натщесерце за вказівками програми (див рис 3.6);

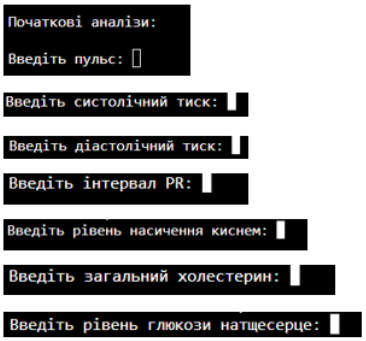


Рисунок 3.6 – Рядки для вводу результатів початкових аналізів пацієнта

3) ввести повторні значення тих самих параметрів для подальшого аналізу (див рис 3.7);

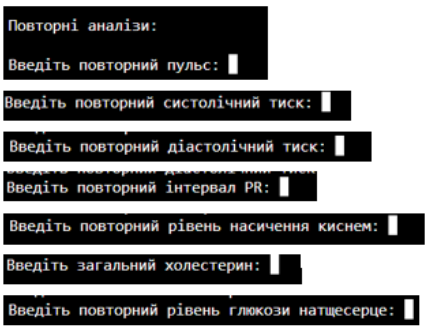


Рисунок 3.7 – Рядки для вводу результатів повторних аналізів пацієнта

4) програма проведе аналіз результатів аналізів та визначить, чи є відхилення від норми, а згодом графічно відобразить результати аналізів у вигляді графіку(можно зберегти як зображення), де порівнюються початкові та повторні аналізи (див. рис. 3.8, див. рис. 3.9);

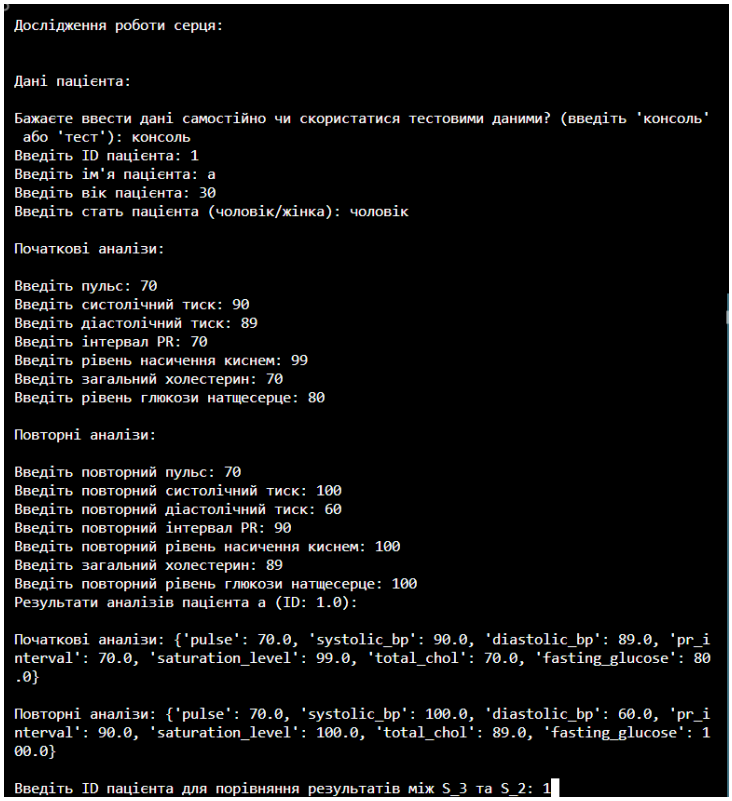


Рисунок 3.8 – Відображення прикладу введених даних «консоль»

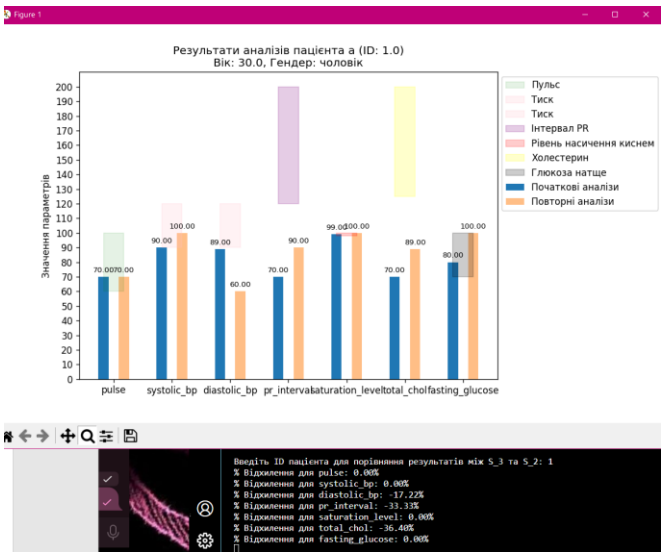


Рисунок 3.9 – Візуалізація прикладу введених даних «консоль»

д) для завершення роботи програми необхідно закрити вікно з графіком та ввести «ні» (див. рис. 3.10);

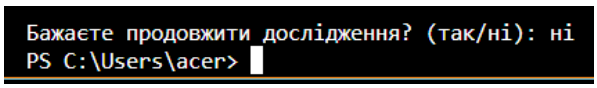


Рисунок 3.10 – Завершення роботи

е) збереження для обох випадків автоматично виконалось до файлу patient_data.xlsx за шляхом D:\ patient_data.xlsx (див. рис. 3.11).

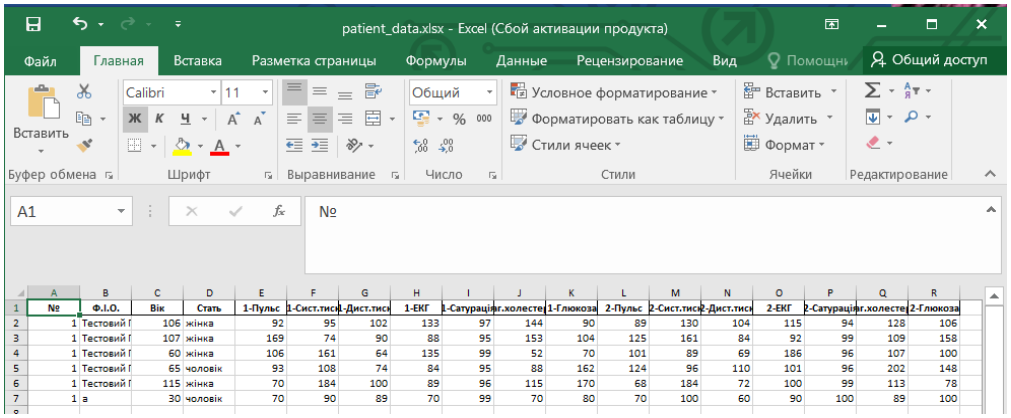


Рисунок 3.11 – Збереження даних

3.5 Апробація створеної моделі

Апробація математичної моделі визначає її ефективність за реальних умов оцінювання стану серцево-судинної системи пацієнта. Ретельна перевірка кожного етапу та внесення коректівів у випадку виявлення невідповідностей гарантує достовірність та точність моделі при використанні на практиці.

Перед проведенням апробації було підготовано приклад даних для вводу з консолі, який відображає різноманіття сценаріїв стану пацієнта. Він включає введення значень для віку, гендеру та показників аналізів.

Приклад.

а) основні дані пацієнта:

- ID пацієнта: 1;
- ім'я пацієнта: Іван;
- вік пацієнта: 4;
- гендер пацієнта: чоловік;

б) початкові аналізи:

- пульс: 90;
- систолічний тиск: 105;
- діастолічний тиск: 70;
- інтервал PR: 90;
- рівень насичення киснем: 98;
- загальний холестерин: 110;
- рівень глюкози натщесерце: 90;

в) повторні аналізи:

- повторний пульс: 105;
- повторний систолічний тиск: 93;
- повторний діастолічний тиск: 68;
- повторний інтервал PR: 110;

– повторний рівень насичення киснем: 99;

– повторний загальний холестерин: 160;

– повторний рівень глюкози натщесерце: 92;

До цього прикладу проілюстровано як математична модель впоралась зі всіма можливими труднощами.

Якщо всі дані введено коректно, користувач отримає повноцінний аналіз даних та їх візуалізацію (див рис. 3.12, рис. 3.13, рис. 3.14, рис. 3.15, рис. 3.16).

Після запуску програми, привітальних повідомлень і вибору користувачем методу «консоль», йде запит на введення особистих даних пацієнта, що й буде успішно виконано(див рис. 3.12).

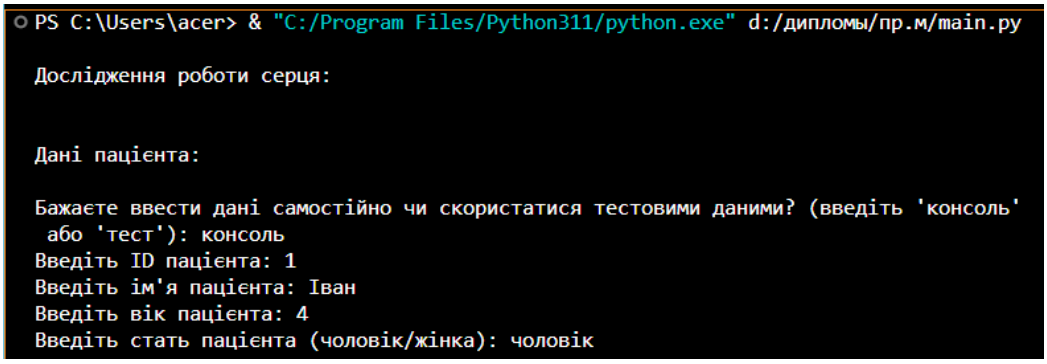


Рисунок 3.12 – Вдала спроба введення особистих даних пацієнта

Далі йде запит на введення результатів початкових аналізів пацієнта, що й буде успішно виконано(див рис. 3.13).

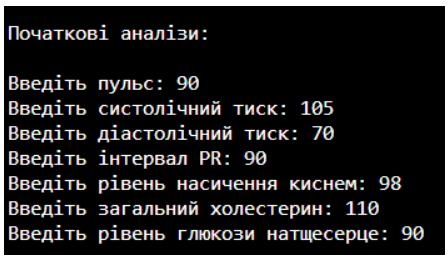


Рисунок 3.13 – Початкові результати аналізів пацієнта

Далі йде запит на введення результатів повторних аналізів пацієнта, що й буде успішно виконано(див рис. 3.14).

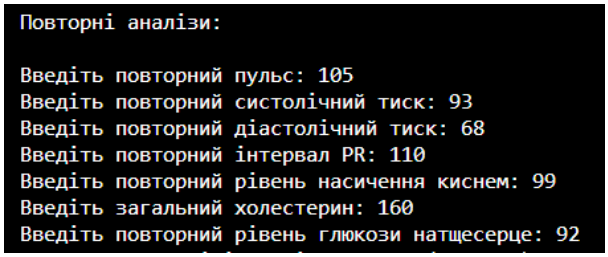


Рисунок 3.14 – Повторні результати аналізів пацієнта

Після успішного введення всіх даних з прикладу, наступним кроком програма виводить стислу довідку. Далі, після підтвердження користувачем № пацієнта, виконується розрахунок відхилень між початковими та повторними значеннями результатів аналізів й порівняннями середнього значення кожного показника з оптимальними показниками, що згодом перетворюється в відсотки(%) і демонструється користувачу (див рис. 3.15).

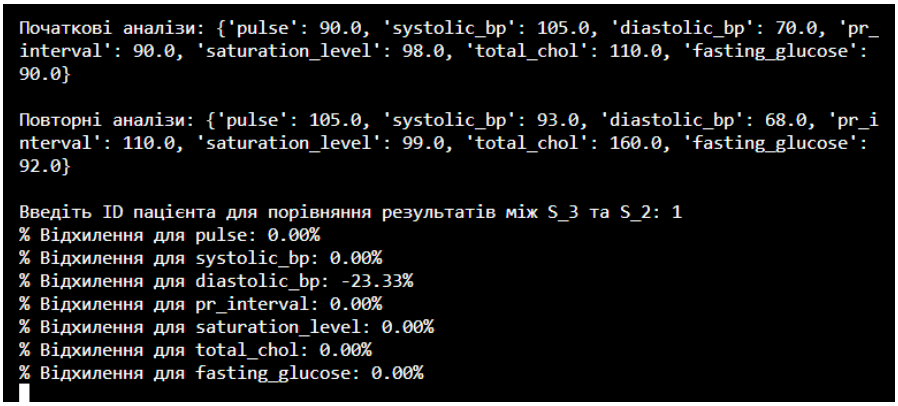


Рисунок 3.15 – Виведення стислої довідки про введені значення та розрахунок відхилень від оптимальних показників

Потім відбувається візуалізація даних на графіку, де відображаються дані та стовпчики з показниками всіх зроблених аналізів пацієнта, а також показано діапазони оптимальних значень для кожного показника, що поясненено в довідці кольорів справа (див рис. 3.16).

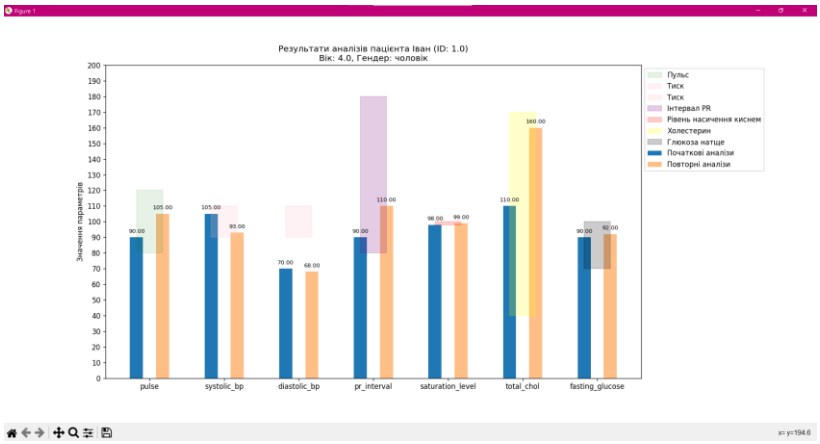


Рисунок 3.16 – Візуалізація даних «консоль»

Після цього виникає вибір: продовжити чи завершити роботу програми. При продовженні, отримуємо другу спробу та обираємо генерацію даних «тест» (див. рис. 3.17).

```
Бажаєте продовжити дослідження? (так/ні): так
Дослідження роботи серця:


Дані пацієнта:

Бажаєте ввести дані самостійно чи скористатися тестовими даними? (введіть 'консоль'
 або 'тест'): тест

Початкові аналізи:


Повторні аналізи:

Результати аналізів пацієнта Тестовий Пацієнт (ID: 1):

Початкові аналізи: {'pulse': 157, 'systolic_bp': 81, 'diastolic_bp': 56, 'pr_interv
al': 131, 'saturation_level': 97, 'total_chol': 212, 'fasting_glucose': 189}

Повторні аналізи: {'pulse': 168, 'systolic_bp': 73, 'diastolic_bp': 93, 'pr_interva
l': 120, 'saturation_level': 97, 'total_chol': 92, 'fasting_glucose': 72}

Введіть ID пацієнта для порівняння результатів між S_3 та S_2: 1
% Відхилення для pulse: 62.50%
% Відхилення для systolic_bp: -14.44%
% Відхилення для diastolic_bp: -17.22%
% Відхилення для pr_interval: 0.00%
% Відхилення для saturation_level: -1.02%
% Відхилення для total_chol: 0.00%
% Відхилення для fasting_glucose: 30.50%
```

Рисунок 3.17 – Тестування генерації даних

Потім відбувається візуалізація даних на графіку, де відображаються дані та стовпчики з показниками всіх зроблених аналізів пацієнта, а також показано діапазони оптимальних значень для кожного показника, що поясненено в довідці кольорів справа (див рис. 3.18).

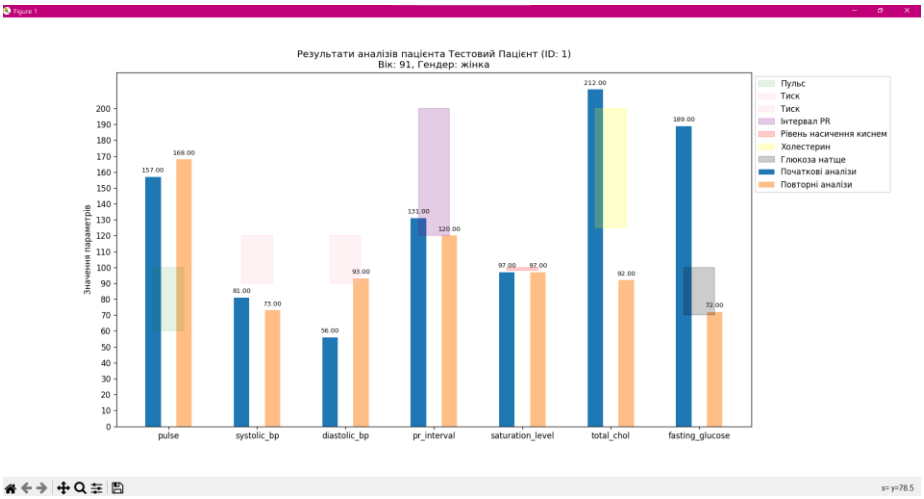


Рисунок 3.18 – Візуалізація даних «тест»

Наприкінці завершаємо життєвий цикл програми(див рис. 3.19).

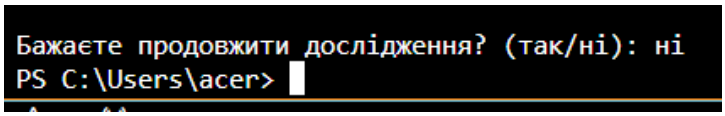
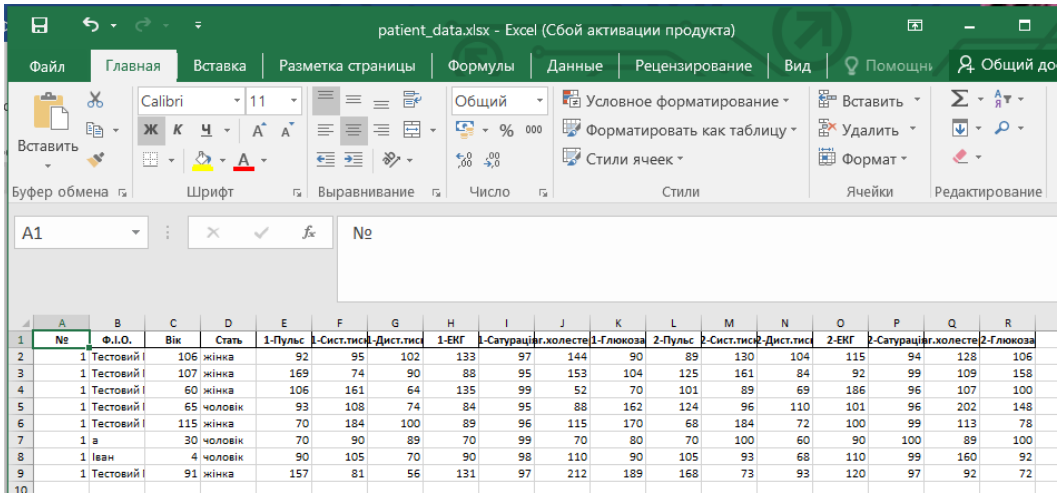


Рисунок 3.19 – Зупинка взаємодії

Перевіримо чи збереглись дані до визначеного файлу (див рис. 3.20).



№	Ф.І.О.	Вік	Стать	1-Пульс	1-Сист.тиск	1-Дист.тиск	1-ЕКГ	1-Сатурація	г.холесте	1-Глюкоза	2-Пульс	2-Сист.тиск	2-Дист.тиск	2-ЕКГ	2-Сатурація	г.холесте	2-Глюкоза
1	Тестовий І	106	жінка	92	95	102	133	97	144	90	89	130	104	115	94	128	106
1	Тестовий І	107	жінка	169	74	90	88	95	153	104	125	161	84	92	99	109	158
1	Тестовий І	60	жінка	106	161	64	135	99	52	70	101	89	69	186	96	107	100
1	Тестовий І	65	чоловік	93	108	74	84	95	88	162	124	96	110	101	96	202	148
1	Тестовий І	115	жінка	70	184	100	89	96	115	170	68	184	72	100	99	113	78
1	а	30	чоловік	70	90	89	70	99	70	80	70	100	60	90	100	89	100
1	Іван	4	чоловік	90	105	70	90	98	110	90	105	93	68	110	99	160	92
1	Тестовий І	91	жінка	157	81	56	131	97	212	189	168	73	93	120	97	92	72

Рисунок 3.20 – Збереження даних піддослідних пацієнтів

Якщо один із рядків введено некоректно, користувач отримає попередження та наступну спробу для введення коректних даних в відповідний рядок. Після цього повториться повноцінний аналіз даних та їх візуалізація. На наступному рисунку відображаються випадки, коли користувач вводить деякі зі значень даних хибними і їх обробку програмою. Помилки обробляються як в класі обробки особистих даних пацієнта, так і в класах початкових й повторних аналізів. Так як при користуванні програмою помилитись важно, всі «помилки» вмістились на одному рисунку (див рис. 3.21).

Отже, проаналізувавши обидва методи введення даних та проілюстровані результати, перевірено та доведено, що кожен компонент моделі відповідає поставленій задачі. Логіку метасистеми послідовної дії за методогією Кліра визначено коректно та не порушено.

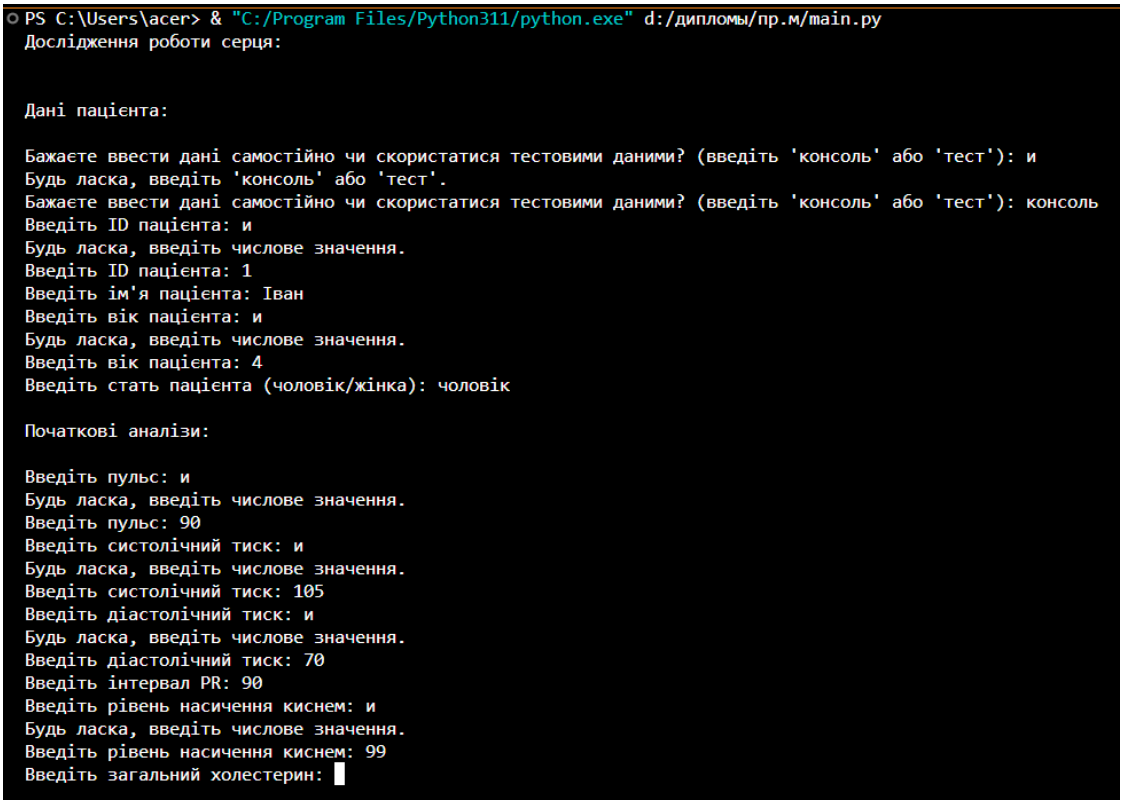


Рисунок 3.21 – Хибні дані та спроби виправлення

ВИСНОВКИ

В ході виконання кваліфікаційної роботи проведено аналіз еволюції методів математичного моделювання слабоструктурованих складних систем. Застосовуючи методологію Кліра, розроблено метасистему послідовної дії, що включає математичну модель для оцінювання стану серцево-судинної системи.

Результати отримані в ході дослідження підтверджують високий рівень ефективності створеної математичної моделі, яка виявляє потенціал для практичного використання в сфері медицини для прогнозування та оцінки стану пацієнтів. Важливо відзначити, що модель реалізовано за допомогою мови програмування Python, що свідчить про гнучкість моделі та легкість адаптації до різноманітних сценаріїв дослідження. Проте, слід зазначити, що результати можуть бути точними не на 100%. У майбутньому модель може бути вдосконалено шляхом розширення функціоналу та оптимізацією інтерфейсу.

У цілому, дипломна робота сприяє розвитку наукового підходу до вивчення слабоструктурованих складних систем, а отримані результати відкривають нові можливості для покращення методів діагностики та моніторингу стану пацієнтів медичних закладів, що має важливе значення для сучасної медичної практики.

ПЕРЕЛІК ПОСИЛАНЬ

1. Власов А. П., Лопатьєв А. О., Пітин М., П’янило Я. Системний підхід і математичне моделювання біологічних та природних об’єктів і процесів. 2016. URL : https://www.researchgate.net/profile/Oleg-Khudolii/publication/305914139_Sistemnij_pidhid_i_matematicne_modeluvanna_biologicnih_ta_prirodnih_ob%27ektiv_i_procesiv/links/57a5bb3608ae3f452931c786/Sistemnij-pidhid-i-matematicne-modeluvanna-biologicnih-ta-prirodnih-obektiv-i-procesiv.pdf (дата звернення: 15.09.2023).

2. Туленков М. В., Лобанова А. С., Яремчук С. С. Системний аналіз у соціології: підручник. Чернівці. Чернівецький національний університет імені Юрія Федьковича. 2023. URL : https://archer.chnu.edu.ua/jspui/bitstream/123456789/7010/1/Системний%20аналіз%20у%20соціології_05-05-23.pdf (дата звернення: 30.09.2023).

3. George J. Klir. Architecture of systems problem solving. State University of New York at Binghamton. Binghamton, New York. 1990.

4. Дивак М. П. Системний Аналіз: методичний посібник. Тернопіль. 2004. URL : http://dspace.wunu.edu.ua/bitstream/316497/627/1/Системний%20аналіз.pdf (дата звернення: 11.10.2023).

5. Метешкін К. А. Основи теорії систем: конспект лекцій. Харків. ХНУГХ. 2014. URL : https://eprints.kname.edu.ua/39257/1/2014%2023Л%20печ.%20Сборка%20конспект%20лекций%202014.pdf (дата звернення: 18.09.2023).

6. Калініченко В. В.Сучасні фізичні та математичні методи досліджень: конспект лекцій. Краматорськ. ДДМА. 2018. URL : http://www.dgma.donetsk.ua/docs/kafedry/kmsit/metod/Конспект%20лекцій-Сучасні%20фізичні%20та%20математичні%20методи%20досліджень-2018.pdf

(дата звернення: 15.09.2023).

7. Python. URL : https://docs.python.org/3/ (дата звернення: 30.09.2023).

8. Microsoft Visual Studio Code. URL : https://code.visualstudio.com (дата звернення: 30.09.2023).

9. MedlinePlus. URL : https://medlineplus.gov (дата звернення: 30.09.2023).

10. Pulse URL : https://medlineplus.gov/ency/article/003399.htm (дата звернення: 30.09.2023).

11. Глушков В. М. Вступ в АСК. Київ : Техніка, 2004. 167 с.

12. Лесечко М. Д. Основи системного підходу: теорія, методологія, практика : Навч. посіб. Львів : ЛРІДУ УАДУ, 2002. 300с.

13. Дудник І. М. Вступ до загальної теорії систем. Київ : Кондор, 2009. 205с.

14. Чорней Н. Б. Теорія систем і системний аналіз: Навч. посіб. для студ. вищ. навч. закл. Київ : МАУП, 2005. –56с.

15. Боєв В. Д. Імітаційне моделювання систем: навч, посібник для прикладного бакалаврату. 2017. 253 с.

16. Graeme Donald Snooks, "A general theory of complex living systems: Exploring the demand side of dynamics", Complexity, vol. 13, no. 6, July/August 2008.

17. John N. Warfield, "A proposal for Systems Science", Systems Research and Behavioral Science, 20, 2003, pp. 507–520.

18. Глушков В. М. Вступ в АСК. Київ : Техніка, 2004. 167 с.

19. Лесечко М. Д. Основи системного підходу: теорія, методологія, практика : Навч. посіб. Львів : ЛРІДУ УАДУ, 2002. 300с.

20. Дудник І. М. Вступ до загальної теорії систем. Київ : Кондор, 2009. 205с.

21. Чорней Н. Б. Теорія систем і системний аналіз : Навч. посіб. для студ. вищ. навч. закл. Київ : МАУП, 2005. 256с.

22. Стеценко І. В. Моделювання систем : навч. посіб. Черкаси : ЧДТУ, 2010. 399 с.

23. Томашевський В. М., Жданова О. Г., Жолдакова О. О. Вирішення практичних завдань методами комп’ютерного моделювання: Навч. посібник. Київ : Корнійчук, 2001. 267с.

24. Blood pressure measurement. URL : https://medlineplus.gov/ency/article/007490.htm (дата звернення: 30.09.2023).

25. Electrocardiogram. URL : https://medlineplus.gov/ency/article/003868.htm (дата звернення: 30.09.2023).

26. Pulse Oximetry. URL : https://medlineplus.gov/lab-tests/pulse-oximetry/ (дата звернення: 30.09.2023).

27. Cholesterol Levels: What You Need to Know. URL : https://medlineplus.gov/cholesterollevelswhatyouneedtoknow.html (дата звернення: 30.09.2023).

ДОДАТОК А

main.py – реалізація метасистеми

```
import numpy as np
import matplotlib.pyplot as plt
from statistics import mean
import sys
import pandas as pd

# -----------------КРОК 1 -----------------
# ІНІЦІАЛІЗАЦІЯ ФУНКЦІЙ ЗІ СТАНДАРТНИМИ MIN/MAX ЗНАЧЕННЯМИ ПАРАМЕТРІВ

# Стандартний пульс
def min_max_pulse(age):
    if age <= 1:
        min_val, max_val = (70, 190)
    elif age <= 2:
        min_val, max_val = (80, 130)
    elif age <= 4:
        min_val, max_val = (80, 120)
    elif age <= 6:
        min_val, max_val = (75, 115)
    elif age <= 9:
        min_val, max_val = (70, 110)
    else:
        min_val, max_val = (60, 100)
    return min_val, max_val

# Стандартний артеріальний тиск
def min_max_blood_pressure(age):
    if age <= 1:
        min_val, max_val = (75, 100)
    elif age <= 9:
        min_val, max_val = (90, 110)
    else:
```

```
        min_val, max_val = (90, 120)
    return min_val, max_val

# Стандартна ЕКГ(Електрокардіографія)
def min_max_ecg(age):
    if age <= 1:
        min_val, max_val = (80, 150)
    elif age <= 21:
        min_val, max_val = (80, 180)
    else:
        min_val, max_val = (120, 200)
    return min_val, max_val

# Стандартна сатурація
def min_max_oxygen_saturation(age):
    if age <= 150:
        min_val, max_val = (98, 100)
    return min_val, max_val

# Стандартний рівень холестерину в крові
def min_max_cholesterol(age):
    if age <= 19:
        min_val, max_val = (40, 170)
    else:
        min_val, max_val = (125, 200)
    return min_val, max_val

# Стандартний рівень цукру в крові
def min_max_blood_sugar(age):
    if age <= 150:
       min_val, max_val = (70, 100)
       return min_val, max_val

# -----------------КРОК 2 -----------------
#СТВОРЕННЯ СИСТЕМИ(класу) ДЛЯ ІНІЦІАЛІЗАЦІЇ ОСОБИСТИХ ДАНИХ ПАЦІЄНТА +
# + ФУНКЦІЯ ВИПАДКОВИХ ЗНАЧЕНЬ З ТЕСТОВИМИ ДАНИМИ

# Створення системи S_1 (особисті дані пацієнта)
```

```
class SystemS1:
    def __init__(self):
        self.patients = []

    def add_patient(self, patient):
        self.patients.append(patient)

    def compare_results(self, patient_id):
        patient_s1 = next((patient for patient in self.patients if patient.id ==
patient_id), None)

        if patient_s1:
            if patient_s1.has_violations:
                print(f"Пацієнт {patient_id} має порушення в системі S_1.")
            else:
                print(f"Пацієнт {patient_id} не має порушень в системі S_1.")
        else:
            print(f"Пацієнт з ID {patient_id} не знайдений.")

# Функція для генерації тестових даних пацієнта
def generate_test_patient():
    age = np.random.randint(0.1, 120)
    gender = np.random.choice(['чоловік', 'жінка'])

    initial_results = {
        'pulse': int(np.random.randint(60, 190)),
        'systolic_bp': int(np.random.randint(60, 190)),
        'diastolic_bp': int(np.random.randint(50, 120)),
        'pr_interval': int(np.random.uniform(80, 200)),
        'saturation_level': int(np.random.uniform(95, 100)),
        'total_chol': int(np.random.randint(50, 240)),
        'fasting_glucose': int(np.random.uniform(70, 200)),
    }

    repeated_results = {param: int(value + np.random.uniform(-1, 1)) for param, value in
initial_results.items()}

    return age, gender, initial_results, repeated_results
```

```
# -----------------КРОК 3 -----------------
#СТВОРЕННЯ СИСТЕМИ(класу) ДЛЯ ІНІЦІАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ПОЧАТКОВИХ АНАЛІЗІВ ПАЦІЄНТА

# Створення системи S_2 (результати початкових аналізів)
class SystemS2:
    def __init__(self, system_s1):
        self.system_s1 = system_s1
        self.patients = []

    def add_patient(self, patient):
        self.system_s1.add_patient(patient)
        self.patients.append(patient)

    def compare_results(self, patient_id):
        self.system_s1.compare_results(patient_id)

# -----------------КРОК 4 -----------------
#СТВОРЕННЯ СИСТЕМИ(класу) ДЛЯ ІНІЦІАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ПОВТОРНИХ АНАЛІЗІВ ПАЦІЄНТА

# Створення системи S_3
class SystemS3:
    def __init__(self, system_s2):
        self.system_s2 = system_s2
        self.patients = []

    def add_patient(self, patient):
        self.system_s2.add_patient(patient)
        self.patients.append(patient)

    def compare_results(self, patient_id):
        self.system_s2.compare_results(patient_id)

# -----------------КРОК 5 -----------------
#СТВОРЕННЯ СИСТЕМИ З ПОЄДНАННЯМ ПОПЕРЕДНІХ КЛАСІВ ДЛЯ УТВОРЕННЯ ЦІЛЬНОГО ОБРАЗУ ПАЦІЄНТА

# Створення СИСТЕМИ Patient
class Patient:
```

```
    def __init__(self, id, name, age, gender, initial_results=None,
repeated_results=None):
        self.id = id
        self.name = name
        self.age = age
        self.gender = gender
        self.initial_results = initial_results if initial_results is not None else {}
        self.repeated_results = repeated_results if repeated_results is not None else {}
        self.has_violations = False

    def add_initial_results(self, results):
        self.initial_results = results

    def add_repeated_results(self, results):
        self.repeated_results = results

    def display_results(self):
        print(f"Результати аналізів пацієнта {self.name} (ID: {self.id}):\n")
        print("Початкові аналізи:", self.initial_results, "\n")
        print("Повторні аналізи:", self.repeated_results, "\n")

# -----------------КРОК 6 -----------------
#ДОПОМІЖНІ ФУНКЦІЇ ДЛЯ КОРЕКТНОЇ РОБОТИ МЕТАСИСТЕМИ

# Функція для введення числового значення з перевіркою на коректність
def input_numeric_value(prompt, min_value=float('-inf'), max_value=float('inf')):
    while True:
        try:
            value = float(input(prompt))
            if min_value <= value <= max_value:
                return value
            else:
                confirm = input(f"Ви впевнені, що ввели правильне значення ({min_value} -
{max_value})? (так/ні): ")
                if confirm.lower() == 'так':
                    return value
        except ValueError:
            print("Будь ласка, введіть числове значення.")
```

```
# Функція для введення даних пацієнта
def input_patient():
    patient_id = input_numeric_value("Введіть ID пацієнта: ", 0)
    patient_name = input("Введіть ім'я пацієнта: ")
    patient_age = input_numeric_value("Введіть вік пацієнта: ")
    patient_gender = input("Введіть стать пацієнта (чоловік/жінка): ")

    return Patient(id=patient_id, name=patient_name, age=patient_age,
gender=patient_gender)

# Функція для введення початкових аналізів
def input_initial_results(patient):
    patient_data = {
        'pulse': input_numeric_value("Введіть пульс: "),
        'systolic_bp': input_numeric_value("Введіть систолічний тиск: "),
        'diastolic_bp': input_numeric_value("Введіть діастолічний тиск: "),
        'pr_interval': input_numeric_value("Введіть інтервал PR: "),
        'saturation_level': input_numeric_value("Введіть рівень насичення киснем: ", 0,
100),
        'total_chol': input_numeric_value("Введіть загальний холестерин: "),
        'fasting_glucose': input_numeric_value("Введіть рівень глюкози натщесерце: "),
    }

    patient.add_initial_results(patient_data)

# Функція для введення повторних аналізів
def input_repeated_results(patient):
    repeated_results = {
        'pulse': input_numeric_value("Введіть повторний пульс: "),
        'systolic_bp': input_numeric_value("Введіть повторний систолічний тиск: "),
        'diastolic_bp': input_numeric_value("Введіть повторний діастолічний тиск: "),
        'pr_interval': input_numeric_value("Введіть повторний інтервал PR: "),
        'saturation_level': input_numeric_value("Введіть повторний рівень насичення
киснем: ", 0, 100),
        'total_chol': input_numeric_value("Введіть загальний холестерин: "),
        'fasting_glucose': input_numeric_value("Введіть повторний рівень глюкози
натщесерце: "),
```

```
    }

    patient.add_repeated_results(repeated_results)

# -----------------КРОК 6 -----------------
#ПОРІВНЯННЯ В НОВІЙ СИСТЕМІ(класі) РЕЗУЛЬТАТІВ ПОЧАТКОВИХ ТА ПОВТОРНИХ
# АНАЛІЗІВ ПАЦІЄНТА, ПОШУК % ВІДХИЛЕНЬ ВІД MIN/MAX

#Створення системи S_4
class SystemS4:
    def __init__(self, system_s3, system_s2):
        self.system_s3 = system_s3
        self.system_s2 = system_s2

    def compare_results(self, patient_id):
        patient_s3 = next((patient for patient in self.system_s3.patients if patient.id ==
patient_id), None)
        patient_s2 = next((patient for patient in self.system_s2.patients if patient.id ==
patient_id), None)

        if patient_s3 and patient_s2:
            # Перший крок: розрахунок середнього значення для кожного параметра
            average_results = {}
            parameters = [
                'pulse', 'systolic_bp', 'diastolic_bp', 'pr_interval', 'saturation_level',
                'total_chol', 'fasting_glucose'
            ]

            for param in parameters:
                average_results[param] = (patient_s3.initial_results[param] +
patient_s3.repeated_results[param]) / 2

            # Другий крок: порівняння результатів з функціями min_max_...
            for param in parameters:
                if param == 'pulse':
                   min_val, max_val = min_max_pulse(patient_s3.age)
                elif param in ['systolic_bp', 'diastolic_bp']:
                    min_val, max_val = min_max_blood_pressure(patient_s3.age)
```

```
                elif param == 'pr_interval':
                   min_val, max_val = min_max_ecg(patient_s3.age)
                elif param == 'saturation_level':
                   min_val, max_val = min_max_oxygen_saturation(patient_s3.age)
                elif param == 'total_chol':
                   min_val, max_val = min_max_cholesterol(patient_s3.age)
                elif param == 'fasting_glucose':
                   min_val, max_val = min_max_blood_sugar(patient_s3.age)

                if min_val <= average_results[param] <= max_val:
                    deviation_percentage = 0
                elif average_results[param] > max_val:
                    deviation_percentage = ((average_results[param] - max_val) / max_val)
* 100
                elif average_results[param] < min_val:
                    deviation_percentage = ((average_results[param] - min_val) / min_val)
* 100

                print(f"% Відхилення для {param}: {deviation_percentage:.2f}%")

        else:
            print(f"Пацієнт з ID {patient_id} не знайдений в одній з систем (S_3 або
S_2).")

# -----------------КРОК 7 -----------------
#ПОБУДОВА ГРАФІКУ, ЩО ПІДСУМОВУЄ ДАНІ, ВІДОБРАЖАЮЧИ СТАН ПАЦІЄНТА

import matplotlib.pyplot as plt
import numpy as np

# Графік
def plot_results(patient):
    parameters = [
        'pulse', 'systolic_bp', 'diastolic_bp', 'pr_interval', 'saturation_level',
        'total_chol', 'fasting_glucose'
    ]

    categories = list(range(len(parameters)))
```

```
    initial_results = [patient.initial_results[param] for param in parameters]
    repeated_results = [patient.repeated_results[param] for param in parameters]

    normal_ranges = []

    for param in parameters:
        normal_range = None
        if param == 'pulse':
            normal_range = min_max_pulse(patient.age)
        elif param in ['systolic_bp', 'diastolic_bp']:
            normal_range = min_max_blood_pressure(patient.age)
        elif param == 'pr_interval':
            normal_range = min_max_ecg(patient.age)
        elif param == 'saturation_level':
            normal_range = min_max_oxygen_saturation(patient.age)
        elif param == 'total_chol':
            normal_range = min_max_cholesterol(patient.age)
        elif param == 'fasting_glucose':
            normal_range = min_max_blood_sugar(patient.age)

        normal_ranges.append(normal_range)

    fig, ax = plt.subplots(figsize=(10, 6))
    width = 0.35
    bar_width = width / 2

    # Стовпчики для початкових і повторних аналізів
    bars1 = ax.bar(np.array(categories) - bar_width, initial_results, width=bar_width,
label='Початкові аналізи')
    bars2 = ax.bar(np.array(categories) + bar_width, repeated_results, width=bar_width,
label='Повторні аналізи', alpha=0.5)

    for i, param in enumerate(parameters):
        normal_range = normal_ranges[i]
        if param == 'pulse':
            ax.fill_between([i - bar_width, i + bar_width], normal_range[0],
normal_range[1], color='green', alpha=0.1, label='Пульс')
        elif param in ['systolic_bp', 'diastolic_bp']:
```

```
            ax.fill_between([i - bar_width, i + bar_width], normal_range[0],
normal_range[1], color='pink', alpha=0.2, label='Тиск')
        elif param == 'pr_interval':
            ax.fill_between([i - bar_width, i + bar_width], normal_range[0],
normal_range[1], color='purple', alpha=0.2, label='Інтервал PR')
        elif param == 'saturation_level':
            ax.fill_between([i - bar_width, i + bar_width], normal_range[0],
normal_range[1], color='red', alpha=0.2, label='Рівень насичення киснем')
        elif param == 'total_chol':
            ax.fill_between([i - bar_width, i + bar_width], normal_range[0],
normal_range[1], color='yellow', alpha=0.2, label='Холестерин')
        elif param == 'fasting_glucose':
            ax.fill_between([i - bar_width, i + bar_width], normal_range[0],
normal_range[1], color='black', alpha=0.2, label='Глюкоза натще')

    ax.set_xticks(categories)
    ax.set_xticklabels(parameters)
    ax.set_yticks(range(0, 210, 10))  # діапазон
    ax.set_ylabel('Значення параметрів')
    ax.set_title(f'Результати аналізів пацієнта {patient.name} (ID: {patient.id})\nВік:
{patient.age}, Гендер: {patient.gender}')

    # Додавання числових значень над стовпцями
    for i, value in enumerate(initial_results):
        ax.text(i - bar_width, value + 2, f'{value:.2f}', ha='center', va='bottom',
color='black', fontsize=8)

    for i, value in enumerate(repeated_results):
        ax.text(i + bar_width, value + 2, f'{value:.2f}', ha='center', va='bottom',
color='black', fontsize=8)

    ax.legend(loc='upper left', bbox_to_anchor=(1, 1))

    plt.show()


# -----------------КРОК 8 -----------------
# ДОДАВАННЯ МОЖЛИВОСТІ ВИБРАТИ СПОСІБ ВВОДУ З КОНСОЛІ
```

```
# Функція для вибору введення даних: з консолі чи тестових даних
def choose_input_method():
    while True:
        sys.stdout.write("Бажаєте ввести дані самостійно чи скористатися тестовими даними?
(введіть 'консоль' або 'тест'): ")
        sys.stdout.flush()

        input_method = sys.stdin.readline().strip().lower()

        if input_method in ['консоль', 'тест']:
            return input_method
        else:
            sys.stdout.write("Неправильний ввід. Будь ласка, введіть 'консоль' або
'тест'.\n")
            sys.stdout.flush()

# -----------------КРОК 9 -----------------
def save_to_excel(patient_data, file_name):
    # Завантаження існуючого файлу, якщо він існує
    try:
        existing_df = pd.read_excel('d:/patient_data.xlsx')
    except FileNotFoundError:
        existing_df = pd.DataFrame()

    data = {
        '№': [patient_data.id],
        'Ф.І.О.': [patient_data.name],
        'Вік': [patient_data.age],
        'Стать': [patient_data.gender],
        '1-Пульс': [patient_data.initial_results['pulse']],
        '1-Сист.тиск': [patient_data.initial_results['systolic_bp']],
        '1-Дист.тиск': [patient_data.initial_results['diastolic_bp']],
        '1-ЕКГ': [patient_data.initial_results['pr_interval']],
        '1-Сатурація': [patient_data.initial_results['saturation_level']],
        '1-Заг.холестерин': [patient_data.initial_results['total_chol']],
        '1-Глюкоза': [patient_data.initial_results['fasting_glucose']],
        '2-Пульс': [patient_data.repeated_results['pulse']],
```

```
        '2-Сист.тиск': [patient_data.repeated_results['systolic_bp']],
        '2-Дист.тиск': [patient_data.repeated_results['diastolic_bp']],
        '2-ЕКГ': [patient_data.repeated_results['pr_interval']],
        '2-Сатурація': [patient_data.repeated_results['saturation_level']],
        '2-Заг.холестерин': [patient_data.repeated_results['total_chol']],
        '2-Глюкоза': [patient_data.repeated_results['fasting_glucose']]
    }

    df = pd.DataFrame(data)

    # Додати новий рядок до існуючого DataFrame
    updated_df = pd.concat([existing_df, df], ignore_index=True)

    # Зберегти оновлений DataFrame у файл
    updated_df.to_excel('d:/patient_data.xlsx', index=False)

# -----------------КРОК 10 -----------------
# ОСНОВНА ЧАСТИНА ПРОГРАМИ

def main():

    while True:
        print("Дослідження роботи серця:\n\n")

        # Створення системи S_1
        print("Дані пацієнта:\n")
      #  sys.stdout.flush()

        input_method = choose_input_method()

        if input_method == 'консоль':
            patient_s1 = input_patient()
            system_s1 = SystemS1()
            system_s1.add_patient(patient_s1)

        else:  # використання тестових даних
            age, gender, initial_results, repeated_results = generate_test_patient()
            patient_s1 = Patient(id=1, name='Тестовий Пацієнт', age=age, gender=gender,
```

```
                                initial_results=initial_results,
repeated_results=repeated_results)
            system_s1 = SystemS1()
            system_s1.add_patient(patient_s1)

        # Створення системи S_2 та введення даних пацієнта та результатів аналізів
        print("\nПочаткові аналізи:\n")
        sys.stdout.flush()

        system_s2 = None
        if input_method == 'консоль':
            input_initial_results(patient_s1)
            system_s2 = SystemS2(system_s1)  # Додайте цей рядок
        else:  # використання тестових даних
            age, gender, initial_results, repeated_results = generate_test_patient()
            patient_s1.add_initial_results(initial_results)

            system_s2 = SystemS2(system_s1)
            system_s2.add_patient(patient_s1)

        # Створення системи S_3 та введення даних пацієнта та результатів повторних
аналізів
        print("\nПовторні аналізи:\n")
        sys.stdout.flush()

        if input_method == 'консоль':
            input_repeated_results(patient_s1)
        else:  # використання тестових даних
            age, gender, initial_results, repeated_results = generate_test_patient()
            patient_s1.add_repeated_results(repeated_results)

        system_s3 = SystemS3(system_s2)
        system_s3.add_patient(patient_s1)
        patient_s1.display_results()

        # Створення системи S_4 та порівняння результатів
        system_s4 = SystemS4(system_s3, system_s2)
```

```
        patient_id = int(input("Введіть ID пацієнта для порівняння результатів між S_3 та
S_2: "))
        system_s4.compare_results(patient_id)

        # Виклик функції для відображення графіку результатів
        plot_results(patient_s1)

        # Зберегання даних поточного користувача у файл
        save_to_excel(patient_s1, 'patient_data.xlsx')

        # Запит у користувача
        sys.stdout.write("\nБажаєте продовжити дослідження? (так/ні): ")
        sys.stdout.flush()
        continue_research = sys.stdin.readline().strip().lower()

        if continue_research != 'так':
            break

if __name__ == "__main__":
    main()
```