МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проєкт

здобувача освіти денної форми навчання

РП.07.05.000.ДП

КАРА ЄВГЕНА МИКОЛАЙОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Інженерія програмного забезпечення»**

Група: **4РП-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів

Проєктний матеріал складається з пояснювальної записки на **77** сторінках та графічного (презентаційного) матеріалу на **10** аркушах (слайдах).

Дипломник _____ (Кара Є. М.)

Керівник _____ (Іванова Л. В.)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист «25» 06 2024 р. Протокол ЕК № 2

Оцінка ЕК 4 (добре) / 85 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І. В.*

"18" 01 *2024 року*

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Кара Євгену Миколайовичу*

1. Тема проєкту *Розробка програмної моделі "Хижак-Жертва"*

для вивчення стохастичних процесів

Затверджена наказом по коледжу від "02" *листопада* *2023 р.*, наказ № *244-А2-ОД*

2 Термін здачі закінченого проєкту 10.06.2024

3. Вихідні дані до проєкту

1. Визначити математичну модель

2. Створити графічну візуалізацію моделі засобами WindowsForms

3. Розробка програмної підтримки засобами мови програмування C#

4. Використовувати принципи ООП

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

1. Аналіз предметної області. 2. Технології та засоби розробки (проєктування).

3. Проєктування графічного інтерфейсу. 4. Проєктування архітектури десктопн застосунку.

5. Розробка десктоп застосунку. 6. Тестування створеного десктоп застосунку.

7. Економічний розрахунок. 8. Аспекти охорона праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація Power Point – 10 слайдів

(Предметна область, мета, практична цінність; Базовий алгоритм; Архітектура застосунку;

Модулі застосунку; Процес розробки; Особливості застосунку; Демонстрація симуляції;

Тестування швидкодії)

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Іванова Л. В.*		
Економічний розділ	*Іванченков В. С.*		
Розділ охорони праці	*Чорновол Н. І.*		
Нормоконтроль	*Петрашова В. І.*		
Старший консультант	*Кривченко Ю. В.*		

7. Дата видачі завдання _____

Керівник *Іванова Л. В.* _____ (підпис)

Завдання прийняв до виконання *Кара Є. М.* _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту (роботи)	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Формування вступу*	*29.04.24*	виконано
2	*Аналіз предметної області*	*10.05.24*	виконано
3	*Підбір технічної літератури*	*19.05.24*	виконано
4	*Вибір технологій та засобів розробки (проєктування)*	*20.05.24*	виконано
5	*Проєктування дизайну застосунку*	*22.05.24*	виконано
6	*Проєктування архітектури застосунку*	*24.05.24*	виконано
7	*Розробка застосунку*	*27.05.24*	виконано
8	*Тестування створеного застосунку*	*29.05.24*	виконано
9	*Оформлення пояснювальної записки*	*31.05.24*	виконано
10	*Оформлення графічної (презентаційної) частини*	*01.06.24*	виконано
11	*Економічний розрахунок*	*02.06.24*	виконано
12	*Опис охорони праці та техніки безпеки*	*09.06.24*	виконано
13	*Аналіз результатів проєктування*	*13.06.24*	виконано
14	*Підготовка доповіді для захисту*	*16.06.24*	виконано

Дипломник _____ (підпис)

Керівник _____ (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 05 000. 00 ДП	Дипломний проєкт		
А4			РП 07. 05 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 05 000. 00 ДП ПЗ		
Розроб.		Кара Є.М.		10.06	Розробка програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів	Літ.	Аркуш 4 / Аркушів 77
Перевірив		Іванова Л.В.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4РП-07	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

Розуміння стохастичних процесів та їхнього впливу на різноманітні сфери знань є ключовим елементом сучасного наукового та технологічного прогресу. Стохастичні процеси, що характеризуються випадковими змінами у часі, знаходять широке застосування у фінансах, економіці, біології, фізиці, соціології та багатьох інших галузях.

У фінансовому секторі, стохастичні моделі використовуються для прогнозування цін на акції, опціонів та інших фінансових інструментів, а також для управління ризиками та портфелями. У біологічних дослідженнях, стохастичні процеси допомагають моделювати еволюцію популяцій, розподіл генетичних властивостей та поширення хвороб. У фізиці, вони використовуються для розуміння теплового руху частинок та інших стохастичних явищ.

Стек технологій, заснований на мові програмування C# та платформі Windows Forms, обрано не випадково. Використання C# забезпечує не лише потужність та ефективність у реалізації алгоритмів, але й широкі можливості у взаємодії з іншими інструментами та бібліотеками. Платформа Windows Forms, у свою чергу, надає зручний інтерфейс для користувача та можливості швидкої розробки інтерактивних програм.

Мета розробки програмної моделі "Хижак-Жертва" полягає у створенні інструменту, який дозволить дослідникам та розробникам аналізувати та експериментувати зі стохастичними процесами. Ця модель буде використовуватися для вивчення взаємодії між "хижаком" та "жертвою" у різних умовах та параметрах, допомагаючи розкрити важливі закономірності та тенденції.

Розробка програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів відкриває можливості для більш глибокого розуміння цих процесів у різних областях. Ця модель може бути застосована у наукових дослідженнях, освітніх цілях та практичних застосуваннях, сприяючи розвитку нових методів аналізу та моделювання складних систем з випадковими змінами.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз предметної області

1.1.1 Огляд існуючих рішень

Розглянемо існуючи рішення, знайдені на перших сторінках пошукачів, наприклад, такі як “Visualize It”, “The University of Queensland” та “Jon Darkow”.

Першим існуючим рішенням може бути сайт “Jon Darkow”.

Сайт “Jon Darkow” – це онлайн-ресурс, що містить колекцію обчислювальних моделей для біології та екології. Автор, ймовірно, має ім'я Jon Darkow, який розробив ці моделі з метою використання їх у навчанні та дослідженнях. Сайт пропонує користувачам можливість гіпотезувати, тестувати та коригувати свої ментальні моделі, використовуючи обчислювальні експерименти.

Основні переваги сайту включають:

Навчання наукових практик: Сайт надає можливість використовувати обчислювальні моделі для вивчення наукових практик, які визначені в курсах AP science та Next Generation Science Standards. Це допомагає студентам краще розуміти процес наукових досліджень та розвиває їхні аналітичні навички.

Взаємодія зі складними системами: Завдяки обчислювальним моделям студенти можуть експериментувати зі складними системами біології та екології, досліджуючи їх динаміку та взаємозв'язки.

Зрозуміння складних математичних відносин: Обчислювальні моделі дозволяють студентам маніпулювати кількостями системи та спостерігати, як нова кількість впливає на систему, навіть не розуміючи всі математичні деталі.

Недоліки сайту можуть включати:

Обмежений спектр вивчених систем: Хоча сайт містить широкий спектр обчислювальних моделей для біології та екології, можливо, деякі конкретні системи або питання можуть бути недостатньо покриті.

Потреба у підтримці: Стрімка робота та підтримка сайту можуть потребувати постійного фінансування, яке забезпечується через пожертви патронів.

Загалом, сайт “Jon Darkow” може бути корисним ресурсом для навчання біології та екології за допомогою обчислювальних моделей, що допомагають студентам краще розуміти складні системи та наукові концепції.

Цей сайт також надає симуляцію моделі “Хижак-жертва”.

На рисунку 1.1 зображено сайт “Jon Darkow”.

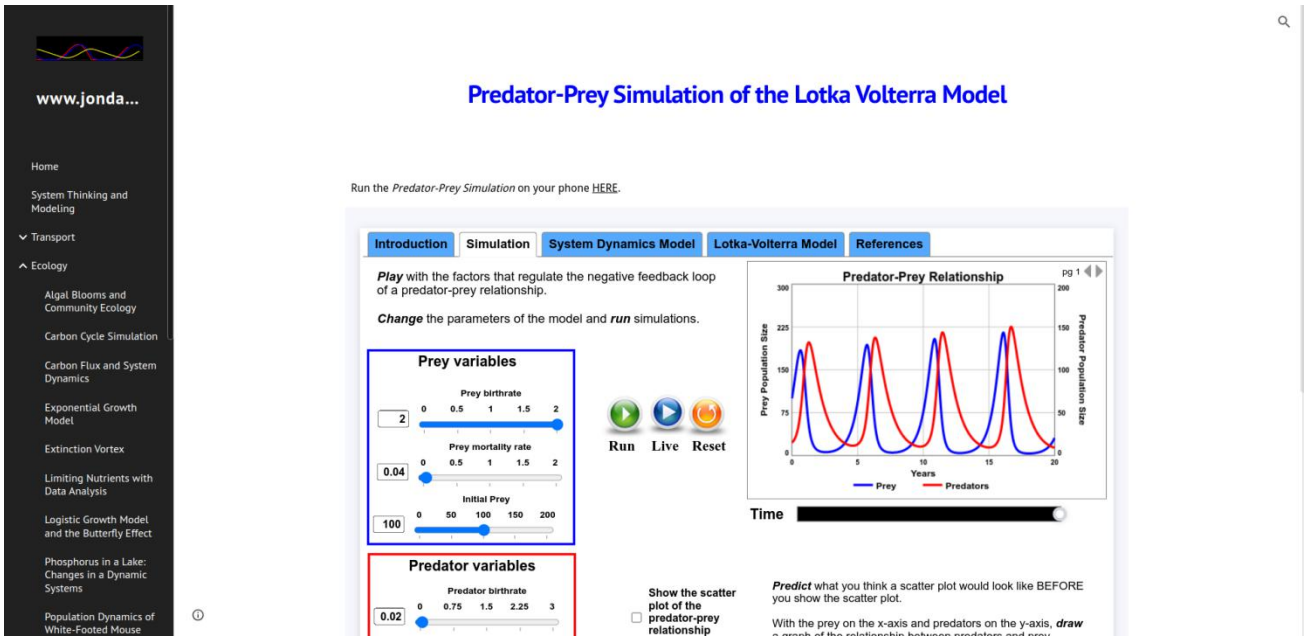


Рисунок 1.1. Сайт “Jon Darkow”

Наприклад, другим існуючим рішенням може бути сайт “Visualize It”.

Сайт “Visualize It” є онлайн-ресурсом, що містить колекцію інтерактивних візуалізацій та симуляцій з різних наукових областей. Автором цього сайту є Chandan Relekar, який розробив ці інтерактивні візуалізації з метою надати користувачам можливість вивчати складні наукові концепції шляхом візуального представлення.

Основні переваги сайту включають:

Інтерактивність: Кожна візуалізація або симуляція на сайті є інтерактивною, що дозволяє користувачам експериментувати з параметрами та спостерігати реальні або теоретичні ефекти у реальному часі.

Мультидисциплінарність: Сайт охоплює широкий спектр наукових областей, включаючи фізику, математику, комп'ютерні науки та складні системи. Це дозволяє користувачам вивчати різноманітні наукові концепції з одного ресурсу.

Навчання та освіта: Сайт може бути використаний для навчання та самостійного вивчення наукових концепцій, а також для демонстрації в класному середовищі.

Недоліки сайту можуть включати:

Обмеженість вмісту: Хоча сайт містить велику кількість візуалізацій, деякі області науки можуть бути менше представленими, ніж інші.

Необхідність актуалізації: Оскільки наука постійно розвивається, сайт може потребувати постійного оновлення та додавання нового вмісту для відображення останніх наукових відкриттів і технологій.

Загалом, сайт "Visualize It" є корисним ресурсом для навчання та вивчення наукових концепцій за допомогою інтерактивних візуалізацій та симуляцій.

Цей сайт також надає симуляцію моделі “Хижак-жертва”.

На рисунку 1.2 зображено сайт “Visualize It”.

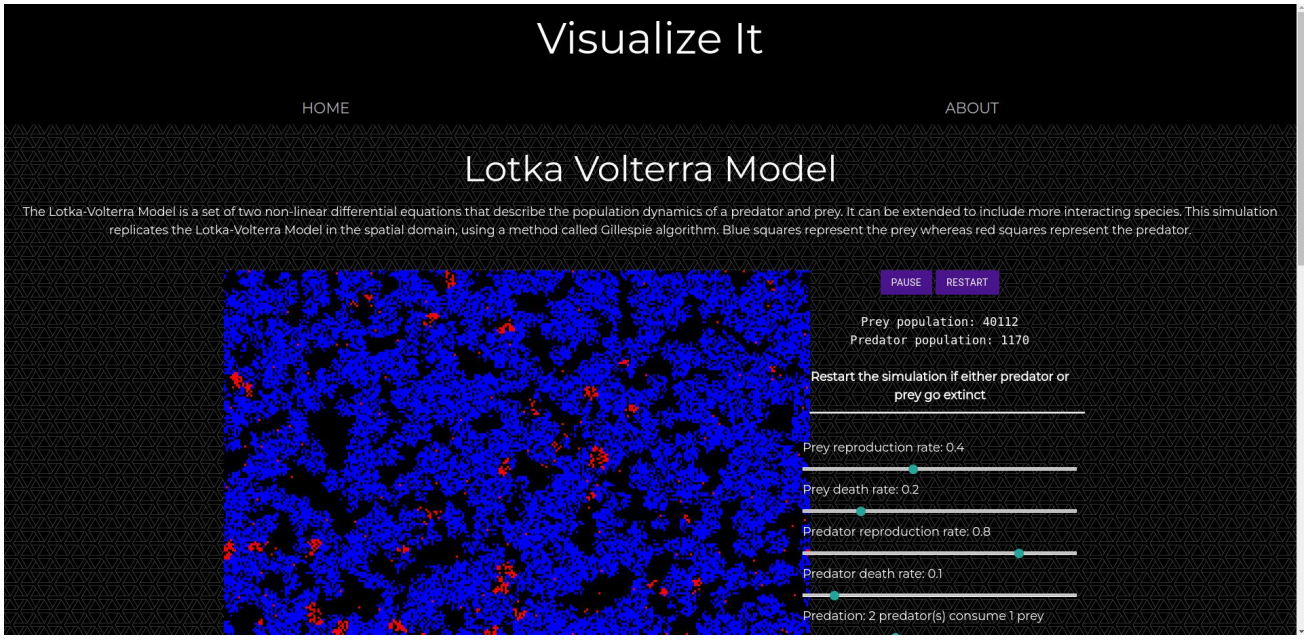


Рисунок 1.2. Сайт “Visualize It”

Сайт Школи Математики та Фізики Університету Квінсленда (The University of Queensland) є важливим онлайн-ресурсом, який надає доступ до інформації про навчальні програми, дослідження та новини в галузі математики та фізики. Сайт присвячений математиці та фізиці, двом ключовим науковим дисциплінам. Він забезпечує студентів, викладачів та дослідників інформацією про академічні можливості, дослідницькі проєкти, новини та інші аспекти цих наукових галузей.

Основні переваги сайту включають:

Інформативність: Сайт надає докладну інформацію про навчальні програми, дослідження та події відбуваються в рамках Школи Математики та Фізики.

Розмаїття тематик: Він охоплює широкий спектр наукових тем, що стосуються математики та фізики, від основних теорій до сучасних досліджень.

Доступність ресурсів: Студенти та дослідники можуть скористатися різними ресурсами, що надаються на сайті, для підвищення рівня знань та розвитку в науці.

Недоліки сайту можуть включати:

Обмеженість оновлень: Через динамічний характер наукових досліджень, деякі інформаційні ресурси можуть бути застарілими або не містити останніх досягнень.

Недостатність взаємодії: Відсутність можливості активної взаємодії з відвідувачами сайту може зменшити ефективність комунікації та обміну інформацією.

Загалом, сайт Школи Математики та Фізики Університету Квінсленда є корисним ресурсом для студентів, викладачів та дослідників, проте йому може бути корисно оновлюватися та покращувати взаємодію з користувачами для підвищення ефективності навчання та досліджень.

Цей сайт також надає симуляцію моделі “Хижак-жертва”.

На рисунку 1.3 зображено сайт Школи Математики та Фізики Університету Квінсленда.

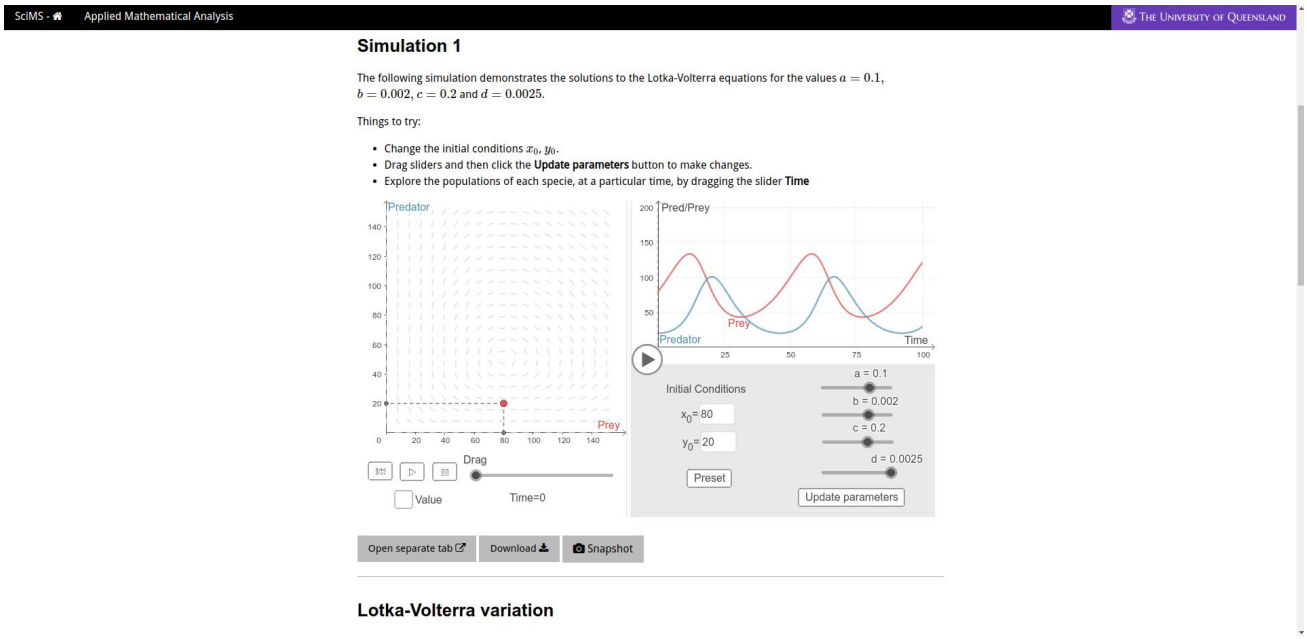


Рисунок 1.1. Сайт Школи Математики та Фізики Університету Квінсленда

Розглянемо порівняльну таблицю аналогів “Visualize It”, “The University of Queensland” та “Jon Darkow”.

У таблиці 1.1. надана порівняльна таблиця сайтів “Visualize It”, “The University of Queensland” та “Jon Darkow”.

Таблиця 1.1. Порівняльна таблиця сайтів

Особливості	**Visualize It**	**The University of Queensland**	**Jon Darkow**
Тип сайту	Онлайн-ресурс із візуалізаціями та симуляціями	Академічний університетський веб-сайт	Онлайн-ресурс із обчислювальними моделями
Предметні області	Фізика, математика, комп'ютерні науки, інші	Математика, фізика, статистика	Біологія, екологія, фізіологія, інші
Інтерактивність	Так	Ні	Так
Мультидисциплінарність	Так	Ні	Так
Навчання та освіта	Так	Ні	Так
Оновлення та додавання вмісту	Немає інформації	Так	Немає інформації

Порівнюючи сайти “Visualize It”, “The University of Queensland” та “Jon Darkow”, можна зробити такі висновки:

“Visualize It” охоплює різноманітні наукові області, такі як фізика, математика та комп'ютерні науки.

Сайт “The University of Queensland” спеціалізується на математиці та фізиці, що робить його корисним ресурсом для студентів та вчених.

“Jon Darkow” зосереджується на біологічних, екологічних та фізіологічних областях, пропонуючи комп'ютерні моделі для вивчення цих процесів.

Інтерактивність є ключовою особливістю “Visualize It” та “Jon Darkow”, що дозволяє користувачам експериментувати та спостерігати наукові ефекти у реальному часі.

“The University of Queensland” не має такого рівня інтерактивності, але надає глибокі матеріали для студентів та дослідників.

“Visualize It” та “Jon Darkow” також відзначаються мультидисциплінарністю, охоплюючи широкий спектр наукових областей у своїх візуалізаціях та моделях.

“The University of Queensland” забезпечує студентів та дослідників різноманітними можливостями для навчання та дослідження в області математики та фізики.

У порівнянні з “Visualize It” та “Jon Darkow”, “The University of Queensland” має більш формальний підхід до навчання та дослідження.

Важливим аспектом “Jon Darkow” є акцент на використанні обчислювальних моделей для вивчення біологічних та екологічних систем.

Кожен з цих сайтів пропонує свій підхід до навчання та дослідження наукових концепцій, залежно від потреб та інтересів користувача.

1.1.2 Технології та засоби розробки

Розглянемо статистику використання мова програмування, знайдені на перших сторінках пошукачів, наприклад, засобами сайту “InvoZone”.

“InvoZone” – це компанія, що надає послуги розробки програмного забезпечення для клієнтів по всьому світу. Вона спеціалізується на створенні веб-застосунків, мобільних застосунків, програмного забезпечення для електронної комерції та інших IT-рішень. Компанія працює з різними технологіями та платформами, щоб задовольнити потреби своїх клієнтів [4].

На рисунку 1.4 зображено статистику використання мова програмування на сайті “InvoZone”.

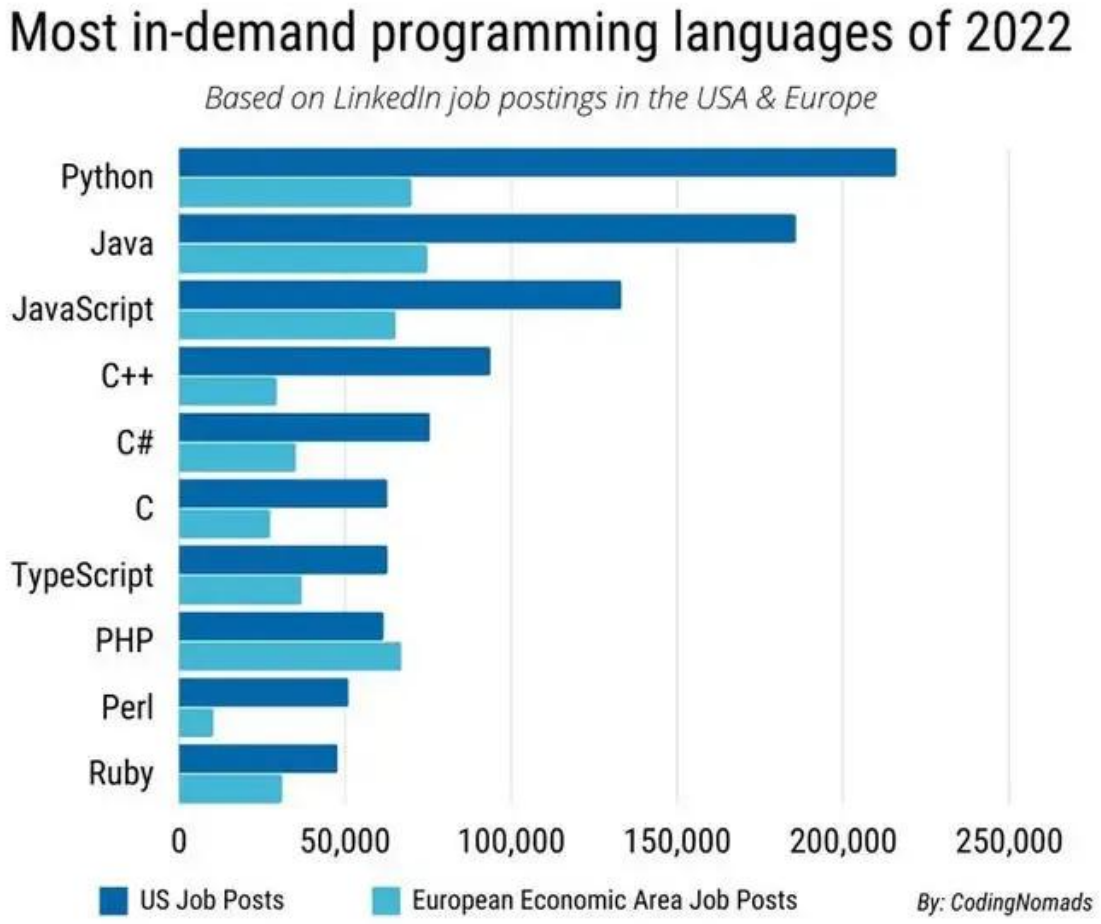


Рисунок 1.4. Статистика використання мов програмування на сайті “InvoZone”

C# (C-sharp) являє собою мову програмування, розроблена компанією Microsoft. Вона популярна в розробці різноманітних застосунків, зокрема веб-застосунків, десктопних програм, мобільних застосунків та ігор. Розглянемо основні переваги та недоліки цієї мови програмування.

Недоліки мови можуть включати:

Простота і зрозумілість: C# має простий і зрозумілий синтаксис, що полегшує навчання для новачків і швидку розробку для досвідчених програмістів.

Підтримка від Microsoft: Мова активно підтримується та розвивається компанією Microsoft, що забезпечує регулярні оновлення та вдосконалення.

Потужний фреймворк .NET: C# інтегрується з .NET, що надає багатий набір бібліотек і інструментів для різноманітних завдань, таких як обробка даних, розробка GUI, робота з базами даних тощо.

Кросплатформеність: Завдяки .NET Core і .NET 5+ C# дозволяє створювати застосунки, які можуть працювати на різних платформах, включаючи Windows, macOS та Linux.

Підтримка різних парадигм програмування: C# підтримує об'єктно-орієнтоване програмування (ООП), функціональне програмування та компоненти імперативного програмування, що дозволяє розробникам використовувати різні підходи для вирішення завдань.

Безпека типів: C# має статичну типізацію, що дозволяє виявляти багато помилок на етапі компіляції, знижуючи кількість багів у виконуваному коді.

Підтримка асинхронного програмування: C# добре підходить для розробки високопродуктивних та масштабованих застосунків завдяки підтримці асинхронного програмування (async/await).

Недоліки мови можуть включати:

Залежність від Microsoft: Хоча це може бути перевагою, але сильна залежність від екосистеми Microsoft також може бути недоліком для деяких розробників, особливо тих, хто працює у середовищах з відкритим кодом.

Відносно висока вимогливість до ресурсів: Застосунки на C# можуть вимагати більше ресурсів у порівнянні з застосунками на деяких інших мовах програмування, таких як C++ або Rust, що може бути критичним для систем з обмеженими ресурсами.

Обмежена підтримка на деяких платформах: Хоча C# є кросплатформеним, деякі бібліотеки або функції можуть бути доступні тільки на Windows або можуть працювати краще на цій платформі.

Затримка у старті: C# застосунки, особливо ті, що працюють на .NET, можуть мати повільний старт через необхідність завантаження та компіляції JIT (Just-In-Time).

Менша спільнота в порівнянні з деякими іншими мовами: Хоча спільнота C# велика і активна, вона може бути меншою порівняно з іншими мовами, такими як JavaScript або Python [5, 6].

Розглянемо порівняльну таблицю мов програмування C# та Java.

У таблиці 1.2. надана порівняльна таблиця мов програмування C# та Java.

Таблиця 1.2. Порівняльна таблиця мов програмування

Особливість	**C#**	**Java**
Тип мови	Об'єктно-орієнтована, має події	Об'єктно-орієнтована
Платформа	Платформа .NET	Java Virtual Machine (JVM)
Синтаксис	Схожий на C++ і Java	Спрощений синтаксис, подібний до C++
Робота з пам'яттю	Менш натхненна, використовує автоматичне керування пам'яттю (garbage collection)	Використовує автоматичне керування пам'яттю (garbage collection)
Обробка винятків	Використовує блоки try-catch-finally для обробки винятків	Використовує блоки try-catch-finally для обробки винятків
Потоки	Використовує класи з простору імен System.Threading	Використовує класи з пакету java.lang.Thread
Наслідування	Підтримує одноразове наслідування класів і багаторазове наслідування інтерфейсів (multiple interface inheritance)	Підтримує тільки одноразове наслідування класів, але може мати багато інтерфейсів
Підтримка мови	Від Microsoft	Від Oracle

C# та Java є популярними об'єктно-орієнтованими мовами програмування, кожна з яких має свої особливості та переваги. C# розроблена Microsoft і працює на платформі .NET, тоді як Java, розроблена Oracle, виконується на Java Virtual

Machine (JVM). Синтаксис обох мов схожий на C++ і є доволі зрозумілим для тих, хто знайомий із цією мовою.

Обидві мови використовують автоматичне керування пам'яттю (garbage collection), що зменшує ризик утворення помилок, пов'язаних з управлінням пам'яттю. Обробка винятків у C# та Java реалізована через блоки try-catch-finally, що забезпечує надійну роботу з виключними ситуаціями.

Під час роботи з потоками, C# використовує класи з простору імен System.Threading, тоді як Java використовує класи з пакету java.lang.Thread. В аспекті наслідування, C# підтримує одноразове наслідування класів та багаторазове наслідування інтерфейсів, тоді як Java підтримує тільки одноразове наслідування класів, але також дозволяє наслідування декількох інтерфейсів.

Обидві мови мають свої специфічні переваги: C# інтегрована з екосистемою Microsoft, що робить її чудовим вибором для розробки Windows-застосунків, тоді як Java є більш кросплатформною завдяки JVM. Вибір між цими мовами залежить від конкретних потреб проєкту та цільової платформи. Враховуючи ці особливості, розробники можуть обирати мову, яка найкраще відповідає їхнім вимогам та середовищу розробки.

Обрана мова програмування має технологію для створення застосунків для десктопної платформи.

Windows Forms являє собою графічну бібліотеку для створення користувацьких інтерфейсів (UI) в застосунках на платформі Windows. Вона є частиною .NET Framework і забезпечує розробникам прості способи створення потужних настільних застосунків.

Переваги технології можуть включати:

Простота використання: Windows Forms забезпечує розробникам простий і зрозумілий інтерфейс для створення графічних застосунків. Завдяки використанню середовища розробки, такого як Visual Studio, розробники можуть створювати UI за допомогою перетягування елементів на форму.

Швидкість розробки: Завдяки інтуїтивно зрозумілому інтерфейсу та великій кількості вбудованих контролів, розробка застосунків на Windows Forms відбувається швидко та ефективно.

Вбудована підтримка .NET: Windows Forms повністю інтегрована з .NET Framework, що дозволяє використовувати всі можливості цієї платформи, включаючи управління пам’яттю, обробку виключень та роботу з базами даних.

Велика кількість бібліотек та компонентів: Існує багато бібліотек та сторонніх компонентів, які можуть бути використані для розширення можливостей Windows Forms застосунків.

Підтримка складних контролів: Windows Forms забезпечує підтримку складних та настроюваних контролів, таких як таблиці, графіки, календарі та інші елементи UI.

Недоліки технології можуть включати:

Застарілість: Windows Forms була створена для старих версій Windows і не використовує переваги нових технологій, таких як Windows Presentation Foundation (WPF) або Universal Windows Platform (UWP).

Обмежена кросплатформеність: Windows Forms призначена виключно для Windows. Це означає, що застосунки, розроблені з її використанням, не будуть працювати на інших платформах, таких як macOS або Linux.

Меньші можливості для сучасних UI: У порівнянні з WPF або іншими сучасними фреймворками, Windows Forms має обмежені можливості для створення складних і сучасних користувацьких інтерфейсів.

Відсутність підтримки нового апаратного забезпечення: Windows Forms не надає вбудованої підтримки для нових технологій, таких як сенсорні екрани або високоякісні дисплеї (наприклад, 4K монітори).

Розглянемо порівняльну таблицю технологій Windows Forms та Swing.

У таблиці 1.3. надана порівняльна таблиця технологій Windows Forms та Swing.

Таблиця 1.3. Порівняльна таблиця технологій

Особливість	**Windows Forms (C#)**	**Swing (Java)**
Платформа	Windows	Будь-яка, де доступний JVM
Мова програмування	C#	Java
Стиль програмування	Імперативний	Імперативний
Шаблони дизайну	Має дизайнер форм для швидкого створення і редагування інтерфейсу	Зазвичай використовується код для створення інтерфейсу
Відомість	Дуже популярний серед розробників .NET	Дуже популярний серед Java-розробників
Гнучкість	Менше гнучкості, але зазвичай працює добре на Windows	Дозволяє більшу гнучкість, але може виникнути проблеми з виглядом на різних платформах
Інтеграція з мовою	Інтегрується добре з C# та .NET	Інтегрується добре з Java
Компоненти	Набір компонентів .NET	Широкий вибір компонентів для будь-яких потреб

Продовження таблиці 1.3.

Документація та підтримка	Добра документація, швидка підтримка від Microsoft	Добра документація, активна спільнота розробників
Ресурсоємність	Може бути більш ресурсоємкою, особливо на старших версіях Windows	Зазвичай має більш низьку вимогливість до ресурсів

Windows Forms (C#) і Swing (Java) є популярними фреймворками для створення графічних інтерфейсів користувача, кожен з яких має свої переваги та недоліки. Windows Forms, працюючи на платформі Windows, інтегрується з мовою програмування C# і екосистемою .NET, що робить його зручним для розробників, які працюють в середовищі Microsoft. Swing, з іншого боку, працює на будь-якій платформі з доступним JVM, що робить його кросплатформним рішенням для розробників на Java.

Windows Forms дозволяє швидко створювати і редагувати інтерфейс за допомогою вбудованого дизайнера форм, тоді як Swing зазвичай вимагає створення інтерфейсу через код, що може займати більше часу. Обидва фреймворки мають хорошу документацію і активну підтримку: Windows Forms підтримується Microsoft, а Swing має активну спільноту Java-розробників.

Що стосується гнучкості, Swing пропонує більшу гнучкість, але може стикатися з проблемами сумісності на різних платформах. Windows Forms менш гнучкий, але зазвичай добре працює на Windows. Відносно ресурсоємності, Windows Forms може бути більш ресурсоємним, особливо на старіших версіях Windows, тоді як Swing зазвичай має меншу вимогливість до ресурсів.

Обираючи між Windows Forms і Swing, розробники повинні враховувати цільову платформу, потреби проєкту та власні навички програмування. Обидва фреймворки можуть ефективно використовуватися для створення потужних і функціональних графічних інтерфейсів користувача.

Таким чином, маємо наступний стек технологій.

На рисунку 1.5 зображено стек технологій.

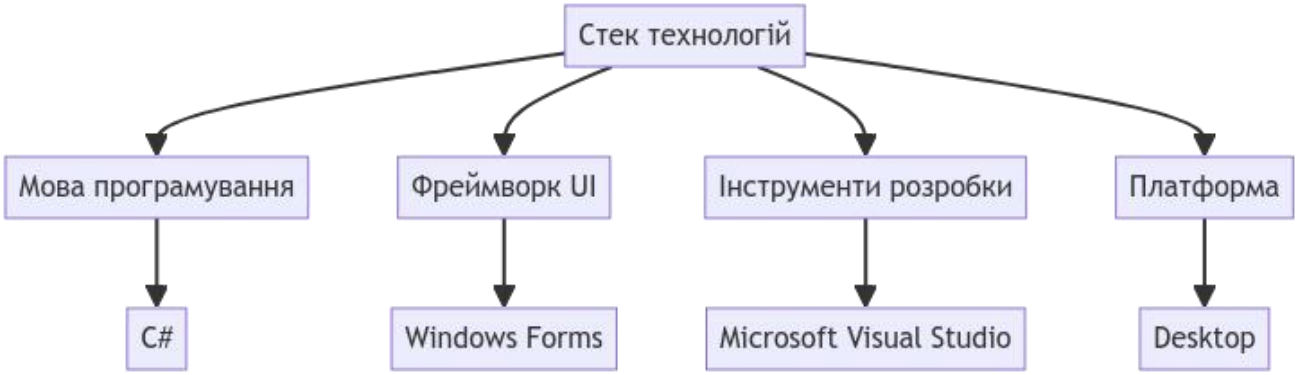


Рисунок 1.5. Стек технологій

1.2 Проєктування десктоп-застосунку

1.2.1 Технічне завдання на розробку

Технічне завдання на розробку включає в себе декілька аспектів.

Аспект: Вивід даних у таблицю.

Програмне забезпечення повинно мати функціонал для відображення даних у табличному форматі. Таблиця повинна підтримувати основні операції: додавання, редагування, видалення та сортування даних. Передбачається динамічне оновлення таблиці при зміні даних, а також можливість налаштування колонок (додавання/видалення, зміна ширини, перейменування). Також має бути підтримка копіювання даних з таблиці в буфер обміну, а також пошук та фільтрація даних у таблиці.

Аспект: Вивід даних у графік.

Програмне забезпечення повинно мати функціонал для відображення даних у графічному форматі. Потрібно підтримувати різні типи графіків (лінійні, стовпчикові, кругові, тощо) з можливістю налаштування параметрів графіка (колір, товщина ліній, підписи осей, легенда). Графік повинен динамічно оновлюватися при зміні даних. Повинна бути можливість збільшення/зменшення масштабу графіка, а також підтримка експорту графіка у зображення.

Аспект: Вивід даних у графічну симуляцію.

Програмне забезпечення повинно мати функціонал для відображення даних у вигляді графічних симуляцій. Це передбачає можливість взаємодії користувача з елементами симуляції для зміни параметрів і спостереження за результатами в реальному часі. Симуляції повинні підтримувати різні моделі та параметри, що налаштовуються, забезпечуючи візуальне представлення складних процесів або систем.

Аспект: Адаптивний графічний інтерфейс.

Програмне забезпечення повинно мати адаптивний графічний інтерфейс, який автоматично підлаштовується під розмір і роздільну здатність екрану користувача. Інтерфейс повинен бути зручним і інтуїтивно зрозумілим, забезпечуючи легкий доступ до всіх основних функцій програми. Адаптивність інтерфейсу повинна гарантувати коректне відображення та роботу на різних пристроях, включаючи комп’ютери, планшети та смартфони.

Аспект: Збереження даних у файл.

Програмне забезпечення повинно мати функціонал для збереження даних у файл. Підтримка різних форматів файлів (наприклад, CSV, JSON, XML) для збереження даних. Функція збереження повинна бути доступною з інтерфейсу програми та забезпечувати збереження всіх поточних даних, включаючи налаштування користувача та стан інтерфейсу.

Аспект: Завантаження даних з файлу.

Програмне забезпечення повинно мати функціонал для завантаження даних з файлу. Підтримка імпорту даних з різних форматів файлів (наприклад, CSV, JSON, XML). Функція завантаження повинна забезпечувати коректне відображення та обробку імпортованих даних у програмі. Повинна бути можливість попереднього перегляду та вибору даних перед завантаженням.

1.2.2 Проєктування графічного інтерфейсу

Розглянемо графічний інтерфейс програмного застосунку.

Windows Forms використовує стиль графічного інтерфейсу, що відомий як "native". Це означає, що елементи керування і структура інтерфейсу повністю відповідають стандартам, зазвичай встановленим для операційних систем Windows. Вони використовують традиційні вигляди кнопок, текстових полів, списків і т. д., які є характерними для інтерфейсу Windows [7].

Графічний інтерфейс імітації моделі хижак-жертва може включати наступні компоненти:

Менюбар: Розташований зазвичай у верхній частині вікна програми, це меню надає доступ до різноманітних опцій, таких як відкриття нових файлів, збереження даних, налаштування програми тощо.

Поля вводу вхідних даних: Ці поля дозволяють користувачеві ввести параметри моделі, такі як початкову кількість хижаків та жертв, швидкості розмноження, коефіцієнти взаємодії тощо.

Вихідні дані у таблиці: Таблиця або сітка, яка відображає числові дані, що виводяться моделлю у кожен момент часу. Ця таблиця може містити інформацію про кількість хижаків та жертв, їх розподіл на певних територіях тощо.

Динаміка змін графіком: Графік або діаграма, яка показує зміну показників моделі в залежності від часу. Наприклад, це може бути графік залежності кількості хижаків і жертв від часу, що допомагає користувачеві візуально оцінити динаміку системи.

Імітація реального часу засобами малювання: Ця функція дозволяє візуалізувати рух хижаків та жертв на екрані в реальному часі. Вона може використовувати графічні об'єкти, такі як кола чи іконки, щоб показати положення та рух об'єктів моделі на екрані.

Ці компоненти допомагають користувачам аналізувати та взаємодіяти з імітованою моделлю хижака-жертви шляхом введення параметрів, спостереження за змінами та вивчення реакції системи на різні умови.

Таким чином маємо наступні модулі графічного інтерфейсу.

На рисунку 1.6 зображено модулі графічного інтерфейсу.

Рисунок 1.6. Модулі графічного інтерфейсу

На рисунку 1.7 зображено процес створення графічного інтерфейсу.

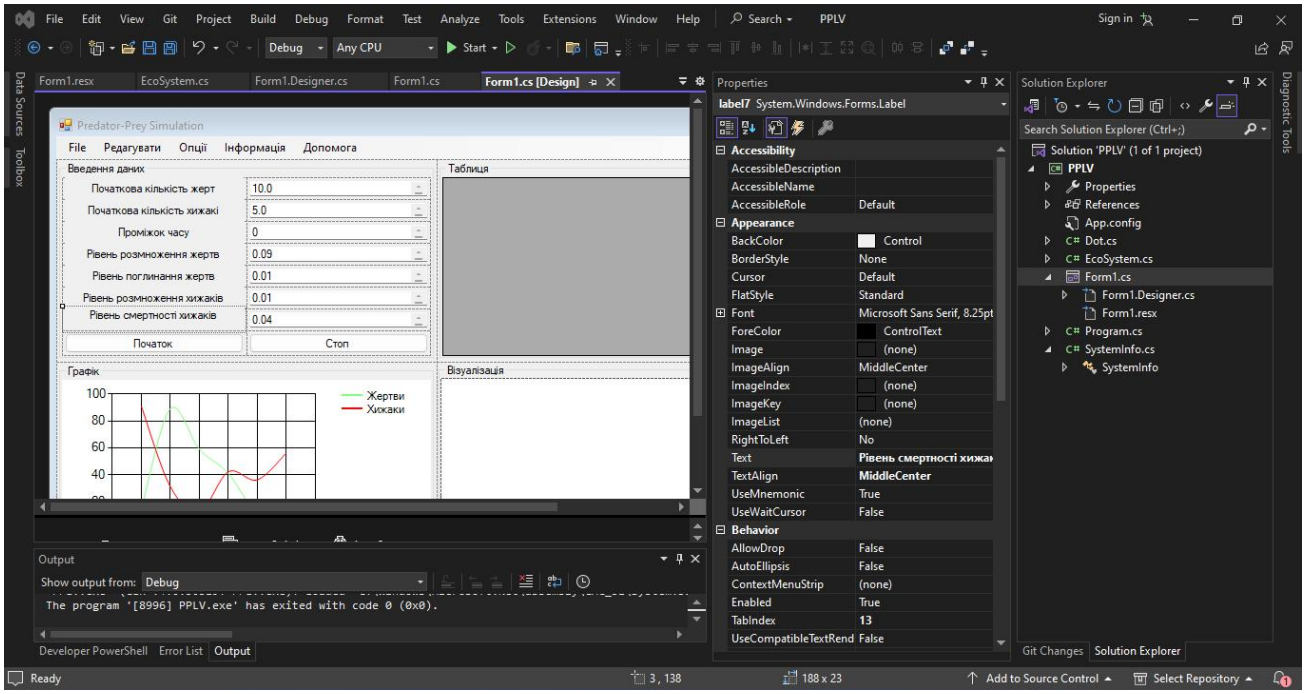


Рисунок 1.7. Процес створення графічного інтерфейсу

1.2.3 Проєктування алгоритму

Ця математична модель буде використовувати модель Лотки-Вольтерра.

Розглянемо модель Лотки-Вольтерра.

Модель Лотки-Вольтерра - це математична модель, яка описує взаємодію між двома видами в екосистемі: один вид є хижаком, а інший - здобиччю. Ця модель використовує систему диференціальних рівнянь, щоб описати зміни у популяціях цих двох видів з часом.

Основні концепції моделі Лотки-Вольтерра включають:

Зростання популяції здобиччю: Популяція здобиччю зазвичай зростає експоненційно без наявності хижаків.

Зменшення популяції здобиччю через хижаків: Чим більше хижаків у системі, тим більше здобичі вони з'їдають, що зменшує популяцію здобичі.

Зростання популяції хижаків через здобич: Хижаки зазвичай залежать від здобичі для їжі, тому коли популяція здобичі зростає, популяція хижаків також може зростати.

Природна смертність хижаків: Хижаки можуть також помирати від природної смертності.

Ця модель може бути використана для дослідження різних аспектів взаємодії між хижаками та їх здобиччю у природних екосистемах. Наприклад, вона може допомогти передбачити, які зміни в популяціях одного виду можуть вплинути на популяції інших видів та які фактори можуть впливати на стабільність екосистеми.

Таким чином, маємо наступний алгоритм.

На рисунку 1.8 зображено схему алгоритму.

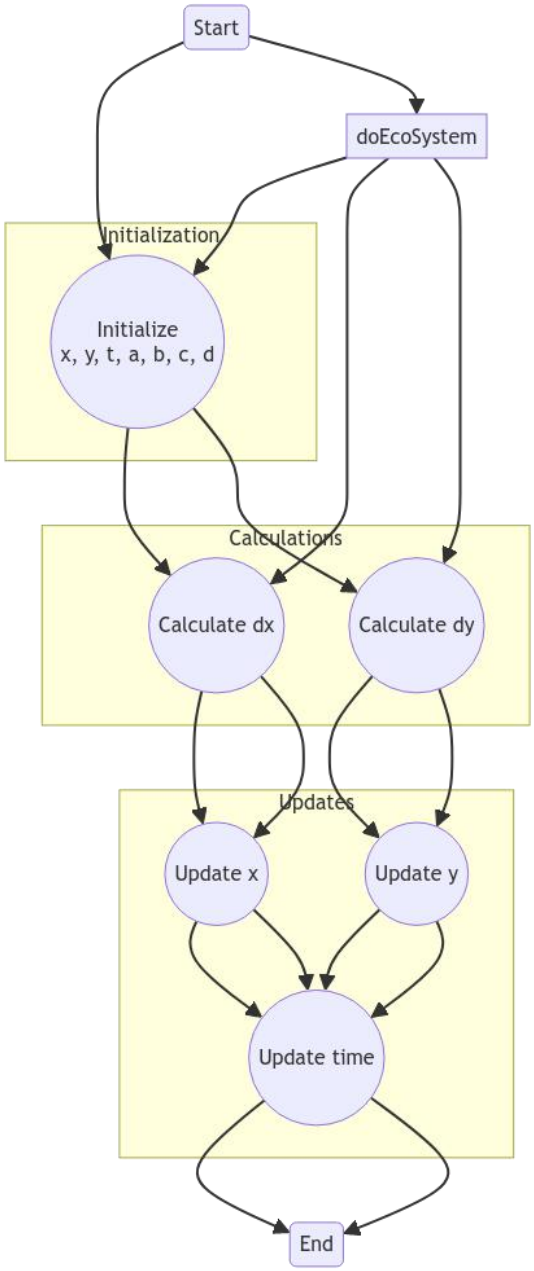


Рисунок 1.8. Схема алгоритму

Алгоритм, реалізований у схемі, відображає динаміку популяцій хижаків та здобичі у моделі Лотки-Вольтерра.

Основні кроки алгоритму такі:

Ініціалізація змінних: Спочатку визначаються початкові значення популяцій здобичі та хижаків (x та y відповідно), часу (t) та параметрів моделі (a, b, c, d).

Обчислення змін: Здійснюється обчислення змін популяцій здобичі та хижаків за одиницю часу. Це робиться згідно зі співвідношеннями, що визначають динаміку моделі Лотки-Вольтерра:

Популяція здобичі зростає за рахунок приросту (a * x) та зменшення через взаємодію з хижаками (-b * x * y).

Популяція хижаків зростає через харчування здобиччю (c * x * y), а зменшується через природну смертність (d * y).

Оновлення значень: Оновлюються значення популяцій здобичі та хижаків (x та y відповідно) за допомогою обчислених змін, а також часу (t) за фіксований проміжок часу.

Повторення кроків: Ці кроки повторюються зазначену кількість разів або до досягнення певної умови завершення моделювання [8].

Отже, цей алгоритм відображає динаміку взаємодії між популяціями хижаків та здобичі у природній екосистемі, де кожна популяція впливає на зміни в іншій популяції через харчування та конкуренцію за ресурси.

1.2.4 Проєктування архітектури десктоп-застосунку

Розглянемо архітектуру застосунку.

З описаних файлів можна зробити припущення, що це проєкт на мові програмування C# (.csproj файл), який містить деякі класи (Dot.cs, EcoSystem.cs, Form1.cs, SystemInfo.cs) та код для конфігурації застосунку (App.config).

Опис можливої архітектури застосунку:

Program.cs: Це вхідна точка застосунку. Вона ініціалізує виконання програми і містить головний метод Main(), який починає виконання.

Form1.cs, Form1.Designer.cs, Form1.resx: Ці файли відповідають за головну форму застосунку. Form1.cs містить логіку взаємодії з формою, Form1.Designer.cs містить автоматично згенерований код, створений візуальним редактором форм Visual Studio, а Form1.resx містить ресурси, пов'язані з формою.

Dot.cs і EcoSystem.cs: Ці файли, містять класи та методи, які відповідають за деякі функції в застосунку. Наприклад, Dot.cs може містити класи та методи,

що стосуються маніпулювання точками, а EcoSystem.cs може містити класи, пов'язані з екосистемою.

SystemInfo.cs: Цей файл може містити класи та методи, які отримують інформацію про систему, на якій працює застосунок.

App.config: Цей файл містить конфігураційні налаштування для застосунку, такі як налаштування бази даних, налаштування з'єднання тощо.

PPLV.csproj: Це файл проєкту Visual Studio. Він містить конфігураційні дані проєкту, такі як посилання на файли коду, залежності та налаштування збірки.

Таким чином, маємо наступну архітектуру.

На рисунку 1.9 зображено схему архітектури застосунку.

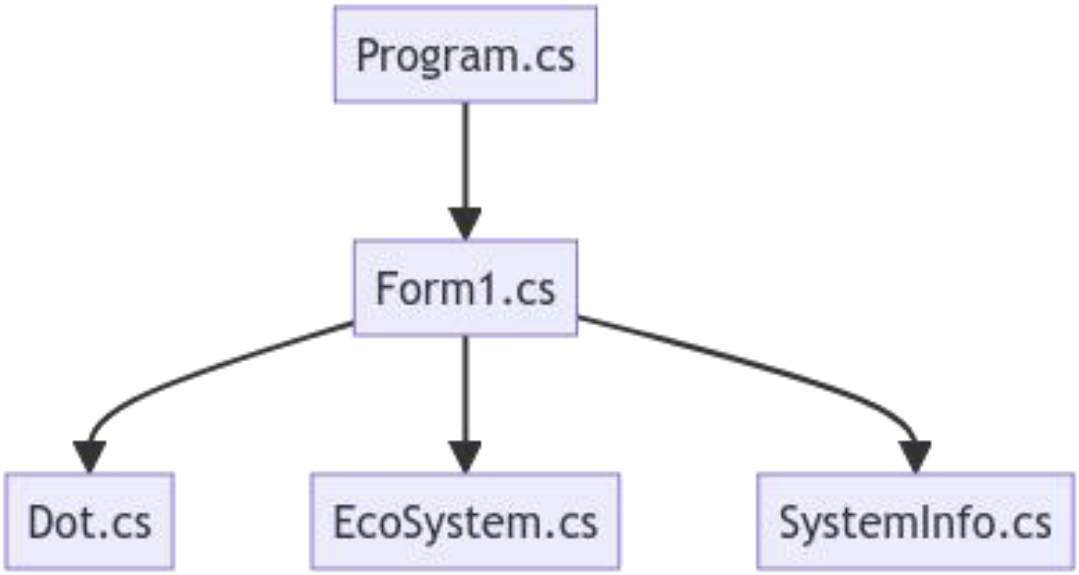


Рисунок 1.9. Схема архітектури застосунку

На рисунку 1.10 зображено файлову архітектуру застосунку.

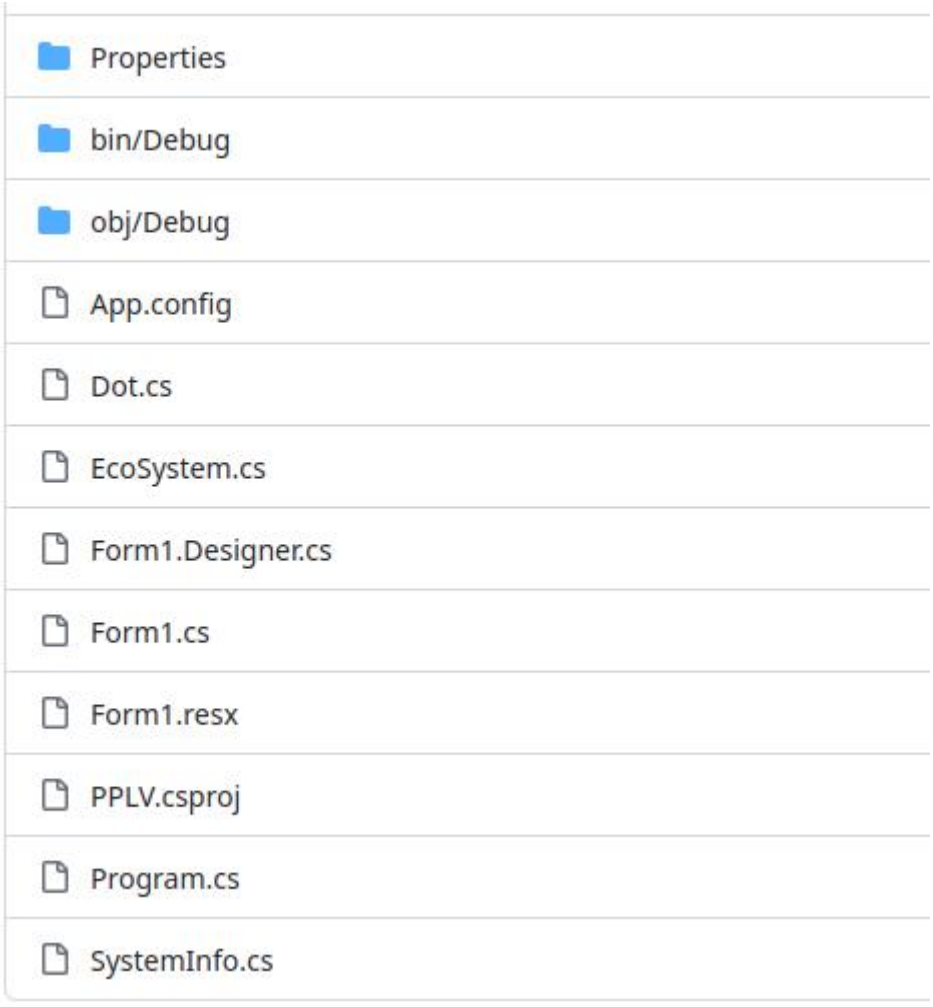


Рисунок 1.10. Файлова архітектура застосунку

1.3 Реалізація десктоп-застосунку

1.3.1 Клас “Program.cs”

Розглянемо клас “Program.cs”.

Цей код написаний на C# і є точкою входу для Windows Forms застосунку під назвою "PPLV".

Нижче наведено код класу “Program.cs”.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PPLV
{
    static class Program
    {
        /// <summary>
```

```
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Ось коротке пояснення того, що він робить:

Код визначає статичний клас Program, який є контейнером для основного методу застосунку.

Метод Main є головною точкою входу для застосунку, яку викликає операційна система при запуску.

Атрибут [STAThread] вказує, що модель багатопоточності застосунку буде одно-апартаментною (Single-Threaded Apartment), що є необхідним для більшості Windows Forms застосунків.

Викликається метод Application.EnableVisualStyles, який увімкне візуальні стилі для застосунку, роблячи його вигляд сучаснішим.

Викликається метод Application.SetCompatibleTextRenderingDefault(false), який встановлює значення за замовчуванням для рендерінгу тексту в контролах.

Метод Application.Run(new Form1()) запускає головний цикл обробки повідомлень для застосунку та показує форму Form1.

Форма Form1 є основним вікном застосунку, яке користувач побачить при запуску.

Усі інші функції та форми застосунку будуть викликані або створені з цього головного вікна.

Цей код забезпечує ініціалізацію і запуск застосунку з необхідними налаштуваннями.

Програма не виконує жодних дій до виклику методу Application.Run, після чого вона залишається активною до закриття головної форми [9].

1.3.2 Клас “EcoSystem.cs”

Розглянемо клас “EcoSystem.cs”. Цей код визначає клас EcoSystem, який моделює екосистему з популяціями хижаків і жертв.

Нижче наведено код класу “EcoSystem.cs”.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PPLV
{
    class EcoSystem
    {
        public double x; // prey
        public double y; // predators
        public double t; // time
        public double a; // growth rate of prey
        public double b; // death of prey
        public double c; // growth of predators
        public double d; // death of predators

        public EcoSystem(double x, double y, double t, double a, double b, double
c, double d)
        {
            this.x = x;
            this.y = y;
            this.t = t;
            this.a = a;
            this.b = b;
            this.c = c;
            this.d = d;
        }

        public void doEcoSystem()
        {
            double dx = (a * x) - (b * x * y);
            double dy = (c * x * y) - (d * y);
            int dt = 1;

            x = x + dx;
            y = y + dy;
            t = t + dt;
        }

        public double getPrey()
        {
            return x;
        }

        public double getPredators()
        {
            return y;
        }

        public double getTime()
```

```
            {
                return t;
            }

        }
    }
```

Ось детальний опис його функціональності:

class EcoSystem - оголошення класу EcoSystem.

В межах класу визначено шість полів типу double: x, y, t, a, b, c, d.

x - популяція жертв.

y - популяція хижаків.

t - час.

a - коефіцієнт росту популяції жертв.

b - коефіцієнт смертності жертв через хижаків.

c - коефіцієнт росту популяції хижаків за рахунок поїдання жертв.

d - коефіцієнт смертності хижаків.

Конструктор EcoSystem(double x, double y, double t, double a, double b, double c, double d) ініціалізує всі поля класу переданими значеннями.

Метод doEcoSystem моделює один крок еволюції популяцій за диференціальними рівняннями Лотки-Вольтерри.

double dx = (a * x) - (b * x * y); - обчислення приросту популяції жертв.

double dy = (c * x * y) - (d * y); - обчислення приросту популяції хижаків.

int dt = 1; - крок часу.

x = x + dx; - оновлення популяції жертв.

y = y + dy; - оновлення популяції хижаків.

t = t + dt; - оновлення часу.

Метод getPrey повертає поточну популяцію жертв.

Метод getPredators повертає поточну популяцію хижаків.

Метод getTime повертає поточний час.

Таким чином, doEcoSystem симулює зміни в популяціях хижаків і жертв на кожному кроці часу, відповідно до простих моделей росту і взаємодії.

Використання рівнянь Лотки-Вольтерри дозволяє моделювати коливання популяцій, які виникають через взаємодію хижаків і жертв.

Код демонструє базову симуляцію екосистеми, що дозволяє аналізувати динаміку популяцій з плином часу.

Клас EcoSystem може бути використаний для подальшого розширення або включення в більші моделі екологічних систем.

Кожен крок моделювання представляє зміну популяцій за одиницю часу, що дозволяє побачити, як змінюються популяції в короткостроковій перспективі.

Використання методу getPrey дозволяє отримати поточну кількість жертв для аналізу або візуалізації.

Використання методу getPredators дозволяє отримати поточну кількість хижаків для аналізу або візуалізації.

Метод getTime корисний для відстеження прогресу симуляції та синхронізації з іншими процесами.

Початкові значення популяцій і коефіцієнтів можуть бути налаштовані через конструктор, дозволяючи створювати різні сценарії для моделювання.

Клас EcoSystem може бути розширений для включення більш складних взаємодій або додаткових видів.

Простота моделі робить її зручною для навчальних цілей та первинного дослідження динаміки популяцій.

Завдяки чіткій структурі, код легко модифікувати для експериментів з різними параметрами.

В загальному, клас EcoSystem забезпечує основу для вивчення екологічних систем і їх поведінки через час.

1.3.3 Клас “Dot.cs”

Розглянемо клас “Dot.cs”.

Цей код визначає клас Dot, який використовується для малювання та переміщення точок на графічній поверхні.

Нижче наведено код класу “Dot.cs”.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PPLV
{
    public class Dot
    {
        public int X { get; set; }
        public int Y { get; set; }
        public int Widthsize { get; set; }
        public int Heightsize { get; set; }

        public Graphics g;

        Random random = new Random();

        public Dot(Graphics graphics)
        {
            g = graphics;
        }
        public void Plot(Color color)
        {
            g.FillEllipse(new   SolidBrush(Color.White),   X,   Y,   Widthsize,
Heightsize);
            X += random.Next(-5, 5);
            Y += random.Next(-5, 5);
            g.FillEllipse(new SolidBrush(color), X, Y, Widthsize, Heightsize);
        }

        public void Clear()
        {
            g.FillEllipse(new   SolidBrush(Color.White),   X,   Y,   Widthsize,
Heightsize);
        }
    }
}
```

Ось детальний опис його функціональності:

class Dot - оголошення класу Dot.

В межах класу визначено п'ять властивостей типу int: X, Y, Widthsize, Heightsize.

X - координата точки по осі X.

Y - координата точки по осі Y.

Widthsize - ширина еліпса, що представляє точку.

Heightsize - висота еліпса, що представляє точку.

Поле Graphics g типу Graphics використовується для малювання на графічній поверхні.

Поле Random random типу Random ініціалізується для генерації випадкових чисел.

Конструктор Dot(Graphics graphics) приймає об'єкт Graphics і присвоює його полю g.

Метод Plot(Color color) малює еліпс на поточній графічній поверхні.

Спочатку еліпс малюється білим кольором для очищення попереднього положення точки.

Координати X та Y змінюються на випадкове значення в діапазоні від -5 до 5.

Потім еліпс малюється в новій позиції вказаним кольором.

Метод Clear() очищує поточну позицію точки, малюючи еліпс білим кольором.

Властивості X, Y, Widthsize, Heightsize мають автоматичні геттери і сеттери, що дозволяє легко змінювати їх значення.

Клас Dot дозволяє малювати точку на графічній поверхні та змінювати її положення за допомогою випадкових чисел.

Конструктор класу приймає графічний об'єкт, що дозволяє малювати на будь-якій поверхні, яка підтримує графіку.

Поле Random забезпечує випадковість переміщення точки, роблячи її траєкторію непередбачуваною.

Метод Plot відповідає за візуалізацію точки, оновлюючи її положення кожного разу при виклику.

Метод Clear корисний для видалення точки з поточного положення перед її переміщенням.

Клас Dot може бути використаний для створення простих анімацій або симуляцій руху точок на екрані.

Зміна положення точки за допомогою випадкових чисел створює ефект "броунівського руху".

Клас можна розширити, додавши більше функціональності, наприклад, зміну розміру або форми точки.

Використання кольору в методі Plot дозволяє легко змінювати вигляд точки.

Код є простим і легко модифікується для різних графічних застосунків.

Завдяки використанню об'єкта Graphics, клас може взаємодіяти з різними графічними середовищами в .NET.

Загалом, клас Dot забезпечує базову функціональність для малювання і переміщення точок на графічній поверхні з використанням випадковості.

1.3.4 Клас “Form1.cs”

Розглянемо клас “Form1.cs”.

Цей код визначає клас Form1, який є формою Windows Forms застосунку, що симулює взаємодію хижаків і жертв у екосистемі.

Нижче наведено код класу “Form1.cs”.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PPLV
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            disableControls(this.Controls);
            osToolStripMenuItem.Text = SystemInfo.getOS();
            lANIPToolStripMenuItem.Text = SystemInfo.getLANIP();
            wANIPToolStripMenuItem.Text = SystemInfo.getWANIP();

            dataGridView1.ColumnCount = 3;
            dataGridView1.RowCount = 1000;
            dataGridView1.Columns[0].Name = "Час";
            dataGridView1.Columns[1].Name = "Жертви";
            dataGridView1.Columns[2].Name = "Хижаки";
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
    try
    {
        OpenFileDialog ofd = new OpenFileDialog();
        if (ofd.ShowDialog() == DialogResult.Cancel)
        {
            return;
        }

        // Read the file lines
        string[] lines = System.IO.File.ReadAllLines(ofd.FileName);
        if (lines.Length < 6)
        {
            MessageBox.Show(this, "Файл не містить достатньо даних!",
                "Message",
                MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
            return;
        }

        // Assign the values to the textboxes
        textBox1.Text = lines[0];
        textBox2.Text = lines[1];
        textBox3.Text = lines[2];
        textBox4.Text = lines[3];
        textBox5.Text = lines[4];
        textBox6.Text = lines[5];
        textBox7.Text = lines[6];

        MessageBox.Show(this, "Файл відкрит!",
            "Message",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
            "Message",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
}

private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
        if (sfd.ShowDialog() == DialogResult.Cancel)
        {
            return;
        }

        // Gather data from textboxes
        string[] lines = new string[6];
        lines[0] = textBox1.Text;
        lines[1] = textBox2.Text;
        lines[2] = textBox3.Text;
        lines[3] = textBox4.Text;
        lines[4] = textBox5.Text;
```

```
            lines[5] = textBox6.Text;
            lines[5] = textBox7.Text;

            // Write the data to the file
            System.IO.File.WriteAllLines(sfd.FileName, lines);

            MessageBox.Show(this, "Файл збережено!",
                "Message",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
                "Message",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
    }

    public void clearControls(Control.ControlCollection ctrlCollection)
    {
        foreach (Control ctrl in ctrlCollection)
        {
            if (ctrl is TextBoxBase)
            {
                ctrl.Text = String.Empty;
            }
            else
            {
                clearControls(ctrl.Controls);
            }
        }
    }

    public void enableControls(Control.ControlCollection ctrlCollection)
    {
        foreach (Control ctrl in ctrlCollection)
        {
            if (ctrl is TextBoxBase)
            {
                ctrl.Enabled = true;
            }
            else if (ctrl is ButtonBase)
            {
                ctrl.Enabled = true;
            }
            else
            {
                enableControls(ctrl.Controls);
            }
        }
    }

    public void disableControls(Control.ControlCollection ctrlCollection)
    {
        foreach (Control ctrl in ctrlCollection)
        {
            if (ctrl is TextBoxBase)
            {
                ctrl.Enabled = false;
```

```
            }
            else if (ctrl is ButtonBase)
            {
                ctrl.Enabled = false;
            }
            else
            {
                disableControls(ctrl.Controls);
            }
        }
    }

    private void unlockToolStripMenuItem_Click(object sender, EventArgs e)
    {
        enableControls(this.Controls);
    }

    private void clearToolStripMenuItem_Click(object sender, EventArgs e)
    {
        clearControls(this.Controls);
    }

    private void restartToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Restart();
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Environment.Exit(0);
    }

    private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
    {
        MessageBox.Show(this, "Predator-Prey Simulation\r\n"
                    + "Kara Evgeny\r\n"
                    + "2024",
                    "Message",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        dateToolStripMenuItem.Text = SystemInfo.getDate();
        timeToolStripMenuItem.Text = SystemInfo.getTime();
        stopwatchToolStripMenuItem.Text = SystemInfo.getStopwatch();
    }

    public static int cnt;

    EcoSystem ecoSystem;

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            dataGridView1.DataSource = null;
            dataGridView1.Rows.Clear();
            dataGridView1.RowCount = 1000;
            dataGridView1.Refresh();
```

```
                chart1.Series[0].Points.Clear();
                chart1.Series[1].Points.Clear();

                cnt = 0;

                ecoSystem = new EcoSystem(Convert.ToDouble(textBox1.Text),
                    Convert.ToDouble(textBox2.Text),
                    Convert.ToDouble(textBox3.Text),
                    Convert.ToDouble(textBox4.Text),
                    Convert.ToDouble(textBox5.Text),
                    Convert.ToDouble(textBox6.Text),
                    Convert.ToDouble(textBox7.Text));

                timer2.Enabled = true;

            }
            catch (Exception ex)
            {
                MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
                    "Message",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            timer2.Enabled = false;
            dotPrey.Clear();
            dotPredators.Clear();
            tempPrey = 0;
            tempPredators = 0;
            Graphics g = panel1.CreateGraphics();
            g.Clear(Color.White);
        }

        Random random = new Random();

        public static List<Dot> dotPrey = new List<Dot>();

        public static List<Dot> dotPredators = new List<Dot>();

        public static int tempPrey = 0;

        public static int tempPredators = 0;

        private void timer2_Tick(object sender, EventArgs e)
        {
            try
            {
                ecoSystem.doEcoSystem();

                Graphics g = panel1.CreateGraphics();

                if ((int)Math.Round(ecoSystem.getPrey()) > tempPrey)
                {
                    for (int i = tempPrey; i < (int)Math.Round(ecoSystem.getPrey()); i++)
                    {
                        dotPrey.Add(new Dot(g)
```

```
                        {
                            X = random.Next(0, panel1.Width),
                            Y = random.Next(0, panel1.Height),
                            Widthsize = random.Next(8, 10),
                            Heightsize = random.Next(8, 10)
                        });
                    }
                }
                else if (tempPrey > (int)Math.Round(ecoSystem.getPrey()))
                {
                    for (int i = 0; i < Math.Abs((int)Math.Round(ecoSystem.getPrey()) - tempPrey); i++)
                    {
                        if (i >= 0)
                        {
                            dotPrey.ElementAt(i).Clear();
                            dotPrey.RemoveAt(i);
                        }
                        else
                        {
                            break;
                        }
                    }
                }

                if ((int)Math.Round(ecoSystem.getPredators()) > tempPredators)
                {
                    for (int i = tempPredators; i < (int)Math.Round(ecoSystem.getPredators()); i++)
                    {
                        dotPredators.Add(new Dot(g)
                        {
                            X = random.Next(0, panel1.Width),
                            Y = random.Next(0, panel1.Height),
                            Widthsize = random.Next(8, 10),
                            Heightsize = random.Next(8, 10)
                        });
                    }
                }
                else if (tempPredators > (int)Math.Round(ecoSystem.getPredators()))
                {
                    for (int i = 0; i < Math.Abs((int)Math.Round(ecoSystem.getPredators()) - tempPredators); i++)
                    {
                        if (i >= 0)
                        {
                            dotPredators.ElementAt(i).Clear();
                            dotPredators.RemoveAt(i);
                        }
                        else
                        {
                            break;
                        }
                    }
                }

                dotPrey.ForEach(dot => { dot.Plot(Color.Green); });
                dotPredators.ForEach(dot => { dot.Plot(Color.Red); });

                tempPrey = (int)Math.Round(ecoSystem.getPrey());
                tempPredators = (int)Math.Round(ecoSystem.getPredators());
```

```
                dataGridView1[0, cnt].Value = ecoSystem.getTime().ToString("F1");
                dataGridView1[1, cnt].Value = ecoSystem.getPrey().ToString("F3");
                dataGridView1[2,                    cnt].Value                    =
ecoSystem.getPredators().ToString("F3");

                chart1.Series[0].Points.AddXY(ecoSystem.getTime(),
ecoSystem.getPrey());
                chart1.Series[1].Points.AddXY(ecoSystem.getTime(),
ecoSystem.getPredators());

                cnt++;
            }
            catch (Exception ex)
            {
                timer2.Enabled = false;
                MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
                    "Message",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
            }
        }
    }
}
```

Ось детальний опис його функціональності:

public partial class Form1 : Form - оголошення класу Form1, який успадковує клас Form.

Конструктор Form1() викликає метод InitializeComponent(), щоб ініціалізувати компоненти форми.

Метод Form1_Load виконується при завантаженні форми і налаштовує початкові параметри.

Викликає disableControls, щоб вимкнути всі елементи управління на формі.

Оновлює елементи меню інформацією про операційну систему, IP-адреси LAN і WAN за допомогою методів класу SystemInfo.

Налаштовує таблицю dataGridView1 для відображення даних з трьома колонками: "Time", "Preys", "Predators".

Метод openToolStripMenuItem_Click відкриває діалогове вікно для вибору файлу та відображає повідомлення про успішне відкриття файлу або помилку.

Метод saveToolStripMenuItem_Click відкриває діалогове вікно для збереження файлу та відображає повідомлення про успішне збереження файлу або помилку.

Метод clearControls очищує всі текстові поля на формі.

Метод enableControls вмикає всі елементи управління на формі.

Метод disableControls вимикає всі елементи управління на формі.

Метод unlockToolStripMenuItem_Click викликає enableControls, щоб увімкнути всі елементи управління на формі.

Метод clearToolStripMenuItem_Click викликає clearControls, щоб очистити всі текстові поля на формі.

Метод restartToolStripMenuItem_Click перезапускає застосунок.

Метод exitToolStripMenuItem_Click закриває застосунок.

Метод aboutToolStripMenuItem_Click відображає інформаційне повідомлення про застосунок.

Метод timer1_Tick оновлює час, дату та значення секундоміра в меню.

Метод button1_Click ініціалізує нову екосистему з введеними параметрами, очищує попередні дані та запускає таймер.

Метод button2_Click зупиняє таймер, очищує точки хижаків та жертв, і очищує панель для малювання.

Метод timer2_Tick виконується при кожному тіку таймера, оновлює стан екосистеми, малює точки хижаків і жертв на панелі, оновлює таблицю та графік, а також обробляє можливі помилки.

1.4 Мануальне тестування десктоп застосунку

1.4.1 Тестування симуляції

Розглянемо запуск застосунку.

На рисунку 1.11 зображено запуск застосунку.

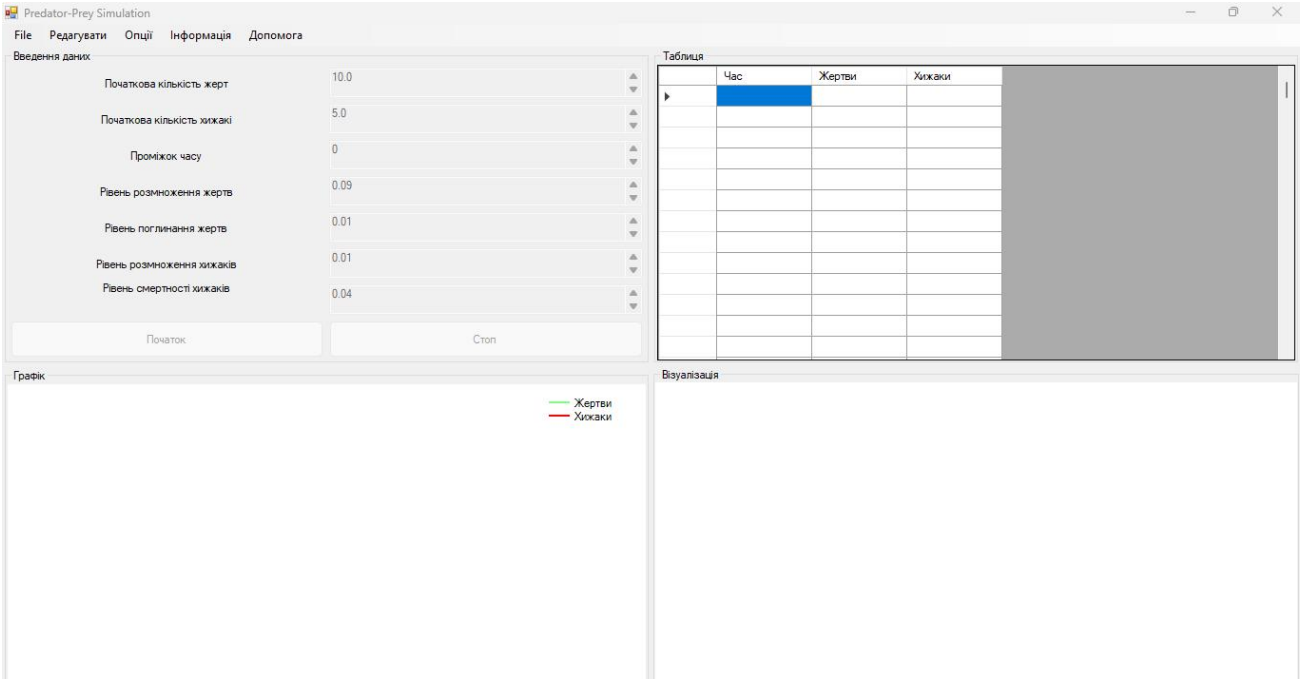


Рисунок 1.11. Запуск застосунку

Розглянемо запуск застосунку.

На рисунку 1.12 зображено запуск симуляції.

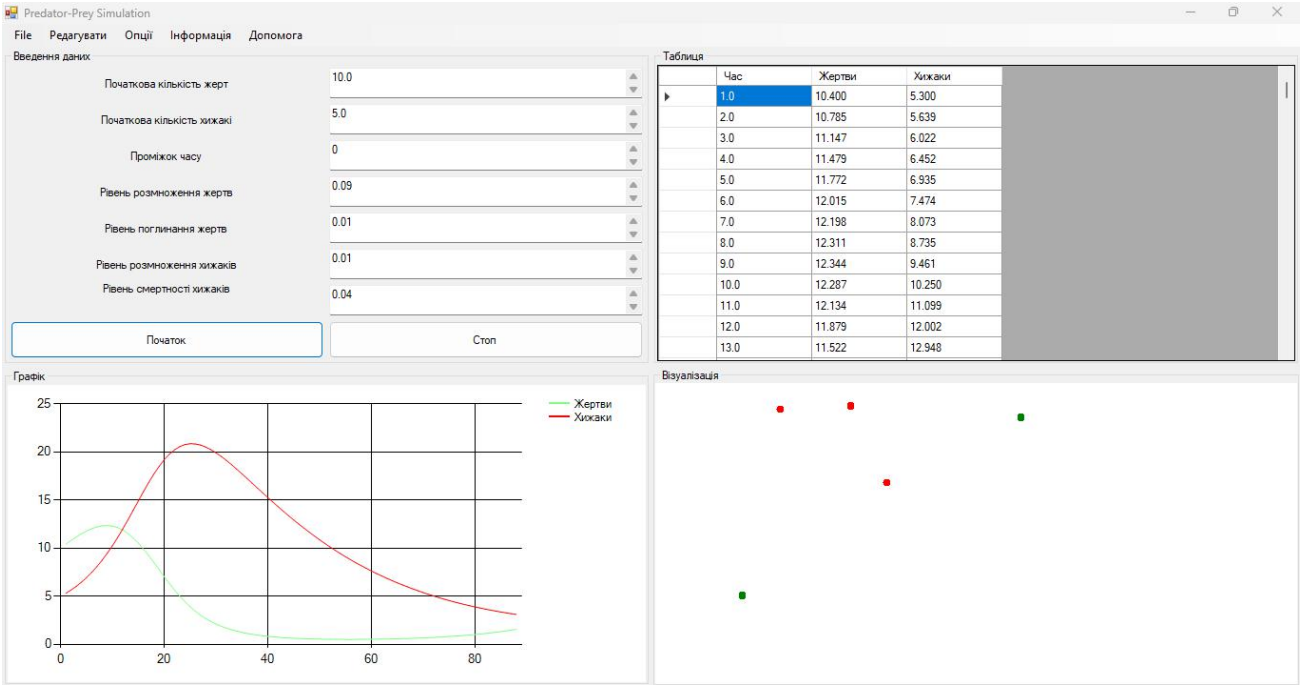


Рисунок 1.12. Запуск симуляції

1.4.2 Тестування адаптивності

Розглянемо поведінку застосунку у маленькому розмірі вікна.

На рисунку 1.13 зображено поведінку застосунку у маленькому вікні.

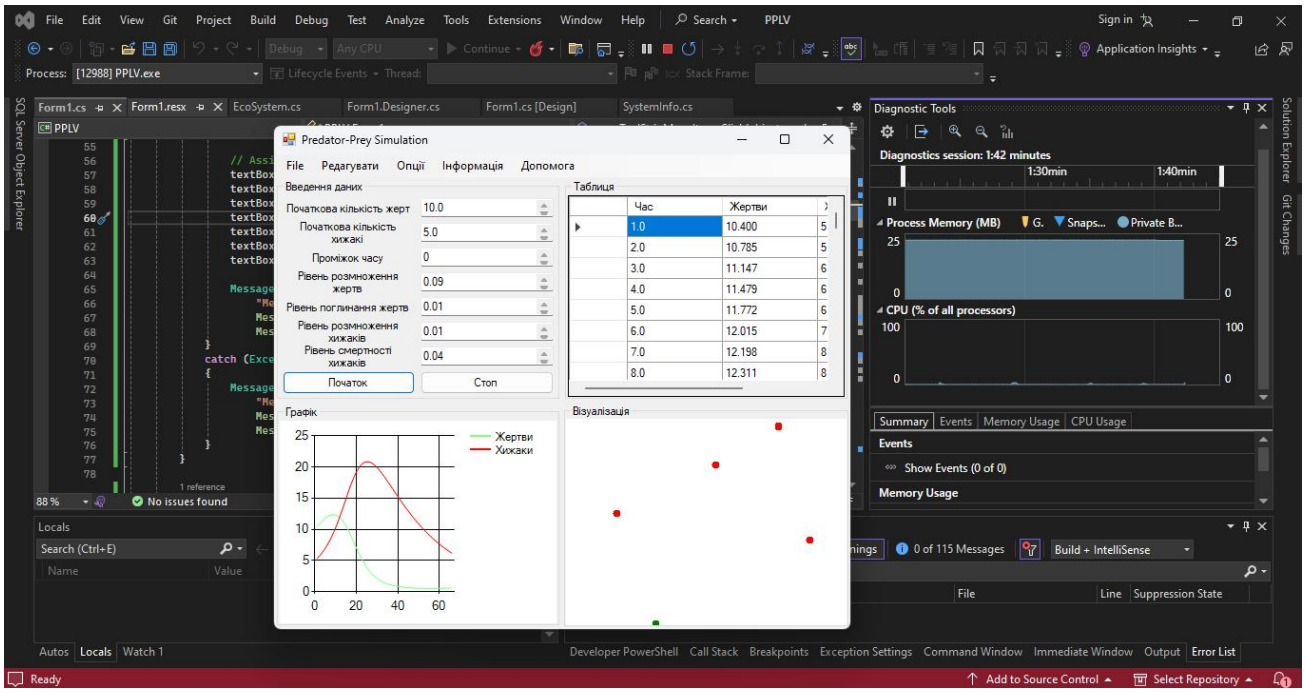


Рисунок 1.13. Застосунок у маленькому вікні

Розглянемо поведінку застосунку у великому вікні.

На рисунку 1.14 зображено поведінку застосунку у великому вікні.

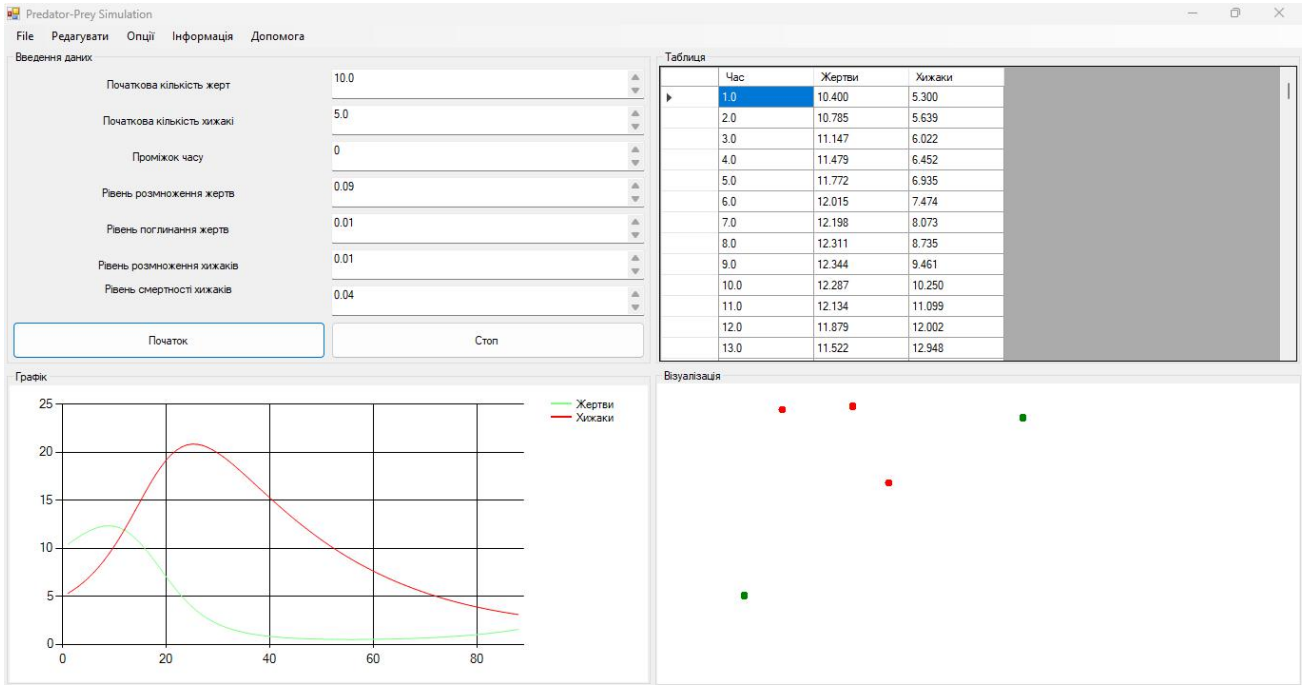


Рисунок 1.14. Застосунок у великому вікні

1.4.3 Тестування швидкодії

Розглянемо поведінку застосунку у великому вікні.

Застосунок під час симуляції витрачає лише 25 мб. оперативної пам’яті та 1% процесору.

На рисунку 1.15 зображено тестування швидкодії застосунку під час симуляції.

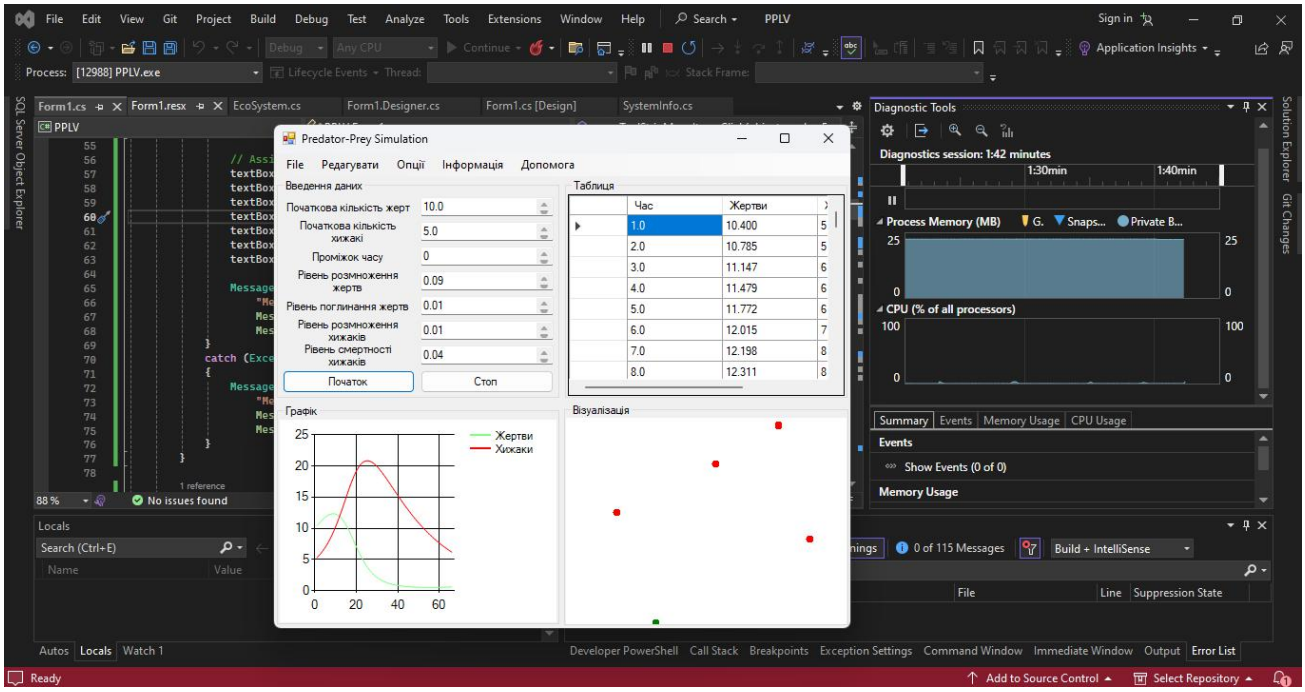


Рисунок 1.15. Тестування швидкодії застосунку під час симуляції

1.4.4 Тестування файлів

Розглянемо відкриття та зберігання файлів вхідної інформації у файл.

На рисунку 1.16 зображено відкриття та зберігання файлу вхідної інформації.

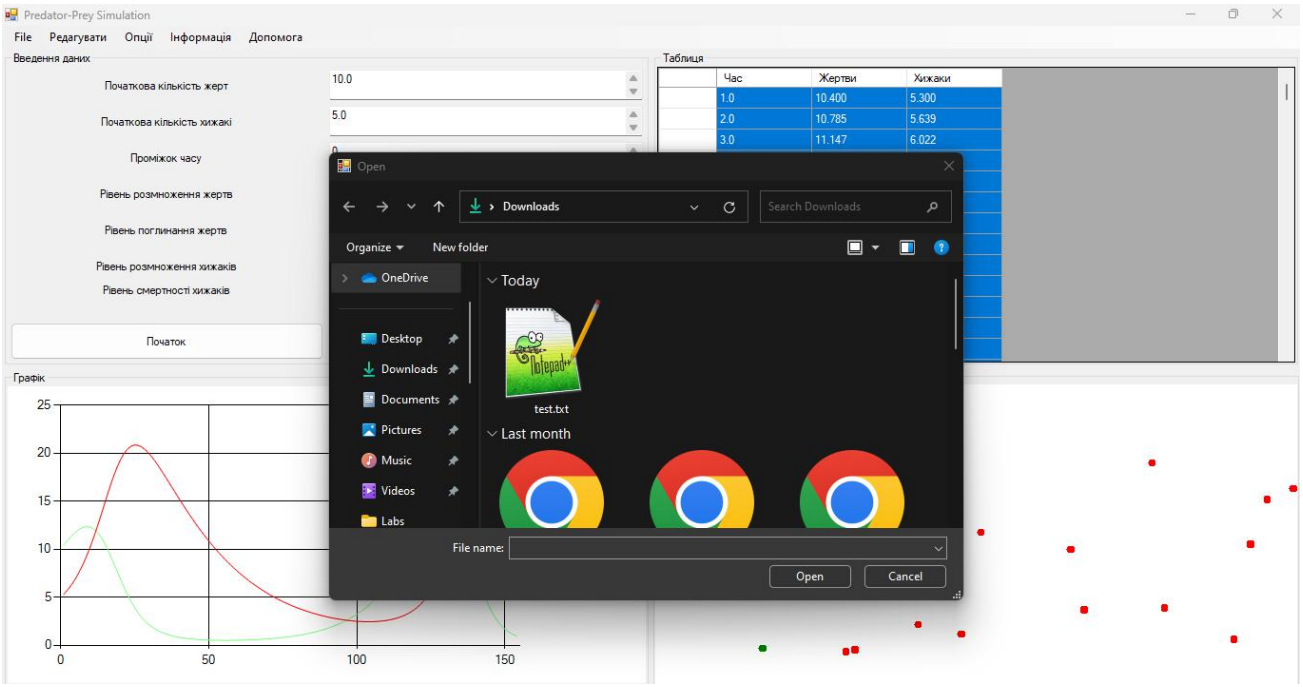


Рисунок 1.16. Відкриття та зберігання файлу вхідної інформації.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проєкті був створений програмний застосунок на мові програмування C# з використанням платформи Windows Forms. Цей застосунок дозволяє візуалізувати та аналізувати взаємодію між "хижаком" та "жертвою" за допомогою стохастичних процесів.

Створення програмної моделі "Хижак-Жертва" для аналізу стохастичних процесів має великий потенціал у різних галузях науки та технологій. Ця модель дозволяє вивчати взаємодію між "хижаком" та "жертвою" і досліджувати їхні впливи від фінансів до біології та фізики.

Використання стохастичних моделей у фінансовому секторі допомагає у прогнозуванні цін на акції, опціони та інші фінансові інструменти, а також в управлінні ризиками та портфелями. У біологічних дослідженнях стохастичні процеси використовуються для моделювання еволюції популяцій, розподілу генетичних властивостей та поширення хвороб.

Поза науковими дослідженнями, модель "Хижак-Жертва" може бути корисною для освітніх цілей та практичних застосувань, сприяючи розвитку нових методів аналізу та моделювання складних систем з випадковими змінами.

Ефективність програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

Проведемо розрахунки визначення трудомісткості розробки даного програмного продукту.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по

відповіднихкаталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

У табл. 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП введення інформації	1060 – 5750

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл. 2.2.

Таблиця 2.2. Обсяг команд

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується

поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар=229х0,8=183,2 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \leftrightarrow L_1 \leftrightarrow K_H \quad (2.1)$$

$$T_{ТП} = T^a p \leftrightarrow L_2 \leftrightarrow K_H \quad (2.2)$$

$$T_{РП} = T^a p \leftrightarrow L_3 \leftrightarrow K_H \leftrightarrow K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеног параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Ттз=Та*L1*Кн=183,2 *0,12*0,7= 15,39 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проєкту

$$Ттп=Та*L2*Кн =183.2 *0,11*0,7 = 17,42 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проєкту

$$Трп=Та*L3*Кн*Кт=183,2 *0,61*0,7*0,7=54,76 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап:технічне завдання Nтз=2 (стор), розробка ТП Nтп=45(стор), розробка робочого проєкту Nрп=8 (стор), пояснювальна записка відповідно Nпз=15 (стор).

Розрахунок зведений у табл. 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин.**		
1.ТЗ	$T_{Pтз}$=15,39	$T_{кк}=0,7*N_{тз}=$ 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=$ 0,15*2=0,30
2.Розробка ТП	$T_{Pтп}$=14,12	$T_{кк}=0,7*N_{тп}=$ 0,7*45 = 31,5	$T_{нк}=0,15*N_{тп}=$ 0,15*45=6,75
3.Розробка РП	$T_{Pрп}$=54,76	$T_{кк}=0,7*N_{рп}=$ 0,7*8=5,6	$T_{нк}=0,15*N_{рп}=$ 0,15*8=1,2
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}=$ 1,5*15 =22,5	$T_{кк}=0,7*N_{тз}=$ 0,7*15=10,5	$T_{нк}=0,15*N_{пз}=$ 0,15*15 =2,25
Усього, в т.ч.:	164,02		
- на розробку	$T_{р}$=106,77		
- контроль		$T_{кк}$=49,00	
- нормоконтроль			$T_{нк}$=8,25

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у табл. 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 квітня 2024 року - 8000 гривень; мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	106,77	48,00	5124,96
2.Контроль керівника	49,00	80,00	3920,00
3.Нормоконтроль	8,25	80,00	660,00
Усього	-	-	Зо= 9704,96

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в табл. 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	76	4.00	304,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0,1 \leftrightarrow Вм1$ = 0,1х304=30,40
Усього				$В_м = В_{мі} + В_{тр\_з=}$ 334.40

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в табл. 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	334.40	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	9704,96	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	970,49	$Зд = 0,1 \leftrightarrow Зо$ =9704,96х0,1
4.Відрахування до єдиного фонду соціального внеску	2606,20	$Вє.с.в. = 0,22 \leftrightarrow (Зо + Зд)$ =0,22х (9704,96+970,49)
5. Накладні витрати	3881,98	$Внак. = 0,4 \leftrightarrow Зо$ =0,4 х 9704,96
6. Повна собівартість	17498,03	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 334.40+9704,96+970,49+2606,20+ 3881,98

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_{пов}*Р)/100=(17498,03*10)/100=1749,80грн. \quad (2.4)$$

Де р – плановий рівень рентабельності (10-20%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о=С_п+П=17498,03+1749,80=19247,83 \text{ грн.} \quad (2.5)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р= Ц_о+ПДВ =19247,83 +19247,83 *0,2=23097,40 \text{ грн.} \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

Зростання сфер діяльності людини, в яких використовуються інформаційні технолог є однією із характерних особливостей сучасного розвитку суспільства. Використання персональних комп'ютерів загострило проблеми збереження власного та суспільного здоров'я, вимагає вдосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів.

В розділі охорони праці дипломного проєкту розглядається питання безпеки праці програміста.

3.2 Аналіз умов праці й забезпечення безпеки при виконання основних видів робіт на об’єкті дипломного проєктування

Оператори і програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика тощо.

3.3 Гігієнічні вимоги до виробничого середовища

Для безпечної та високопродуктивної праці на робочому місці програміста повинні бути створені умови у відповідності з вимогами нормативних та законодавчих актів.

3.3.1 Вимоги до приміщення експлуатації ПК

Розміщення робочих місць з ПК у підвальних приміщеннях, на цокольних поверхах заборонено. Санітарна норма площі на одне робоче місце становить не менше ніж 6,0 м2, а об’єм – не менше ніж об'єм 20,0 м3.

Об'ємно-планувальні рішення будівель та приміщень для роботи з ПК мають відповідати вимогам ДСанПіН 3.3.2.007–98.

Приміщення для роботи з ПК повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2006. Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче, ніж 1,5%.

Виробничі приміщення повинні обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо, з урахуванням вимог до площі приміщень. Також приміщення з ПК мають бути оснащені аптечками першої медичної допомоги.

3.3.2 Гігієнічні вимоги до параметрів повітря приміщень із ПК

У виробничих приміщеннях на робочих місця із ПК мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря відповідно до ДСН 3.3.6.042-99 «Державні санітарні норми мікроклімату виробничих приміщень».

У таблиці 3.1 наведено норми мікроклімату для приміщень.

Таблиця 3.1. Норми мікроклімату для приміщень

Пора року	**Категорія робіт**	**Температура повітря, °С, не більше**	**Відносна вологість повітря %**	**Швидкість руху повітря, м/с**
Холодна	Легка-1а	22-24	40-60	0,1
	Легка-1б	21-23	40-60	0,1
Тепла	Легка-1а	23-25	40-60	0,1
	Легка-1б	22-24	40-60	0,1

Рівні позитивних і негативних іонів у повітрі приміщень з ВДТ мають відповідати санітарно-гігієнічним нормам № 2152-80.

У таблиці 3.2 наведено санітарно-гігієнічні нормам № 2152-80.

Таблиця 3.2. Санітарно-гігієнічні нормам № 2152-80

Рівні	Число іонів в 1 $см^3$ повітря	Число іонів в 1 $см^3$ повітря
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

3.3.3 Виробниче освітлення

Штучне освітлення в приміщеннях із робочими місцями, обладнаними ПК має здійснюватись системою загального рівномірного освітлення. У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення, додатково встановлюються світильники місцевого освітлення)

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300–500 лк, Якщо це неможливо забезпечити системою загального освітлення, допускається використання місцевого освітлення. При цьому світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати бликів на поверхні екрана, а освітленість екрана має не перевищувати 300 лк. Як джерела світла для штучного освітлення застосовуються переважно люмінесцентні лампи типу ЛБ.

3.3.4 Організація робочих місць із ПК

Обладнання й організація робочого місця програміста мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам, з урахуванням особливостей трудової діяльності.

Конструкція робочого місця користувача ПК має забезпечити підтримання оптимальної робочої пози..

При розміщенні робочих столів з ПК слід дотримуватись таких відстаней: між бічними поверхнями БДТ – 1,2 м; від тильної поверхні одного ПК до екрана іншого – 2,5 м.

Екран ПК має розташовуватися на оптимальній відстані тещ очей користувача, що становить 600…700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури в межах 5… 15°.

3.3.5 Вимоги до режимів праці та відпочинку при роботі з ПК

Для збереження здоров'я працюючих, запобігання професійним захворюванням і підтримки працездатності при організації праці, пов'язаної з використанням ПК передбачаються внутрішньо змінні регламентовані перерви для відпочинку.

Впродовж робочої зміни мають передбачатися:

- перерви для відпочинку і вживання їжі (обідні перерви);
- перерви для відпочинку й особистих потреб (згідно з трудовими нормами);
- додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, запобігання втомі.

Працюючі з ПК підлягають обов'язковим медичним оглядам: попереднім – при влаштуванні на роботу і періодичним – протягом трудової діяльності, відповідно до наказу МЗ України N45 від 31.03.94 р.

3.4 Пожежна безпека

Пожежі руйнують виробничі будівлі, знищують матеріали і готову продукцію, приводять в негідність обладнання, на тривалий час припиняють виробничий процес.

Під пожежною безпекою розуміють систему державних і суспільних заходів, спрямованих на охорону від вогню людей і матеріальних цінностей.

Протипожежний захист приміщення забезпечується застосуванням автоматичної установки пожежної сигналізації, наявністю засобів пожежогасіння, застосуванням основних будівельних конструкцій будинку з регламентованими межами вогнестійкості, організації своєчасної евакуації людей

Для ліквідації пожеж використовують первинні засоби пожежогасіння, які призначенні для гасіння пожеж у початковій стадії їх розвитку. Вони є у всіх виробничих приміщеннях, ділянках.

Це вогнегасники (вуглекислотні та порошкові), пожежний інвентар (покривала з негорючого полотна, ящики з піском, бочки з водою), пожежний інвентар.

Первинні засоби пожежогасіння призначені для ліквідації невеликих осередків пожеж, а також для гасіння пожеж на початковій стадії їхнього розвитку силами персоналу об’єктів до прибуття штатних підрозділів пожежної охорони.

Як правило, пожежний інвентар та інструменти розміщуються на спеціальних пожежних щитах.

На рисунку 3.1 зображено розміщення пожежного інвентарю.

Рисунок 3.1. Розміщення пожежного інвентарю

Щити встановлюють на території об’єкта з розрахунку один щит на площу 5000 м2. На видних місцях встановлюють відповідні знаки, що вказують на місце знаходження пожежного щита чи вогнегасника.

Як первинні засоби пожежогасіння використовують вогнегасники, які характеризуються високою вогнегасною спроможністю та значною швидкодією.

На рисунку 3.3 зображено засоби пожежогасіння.

Рисунок 3.1. Розміщення пожежного інвентарю

Приміщеннях з використанням ПК оснащують вуглекислотними вогнегасниками з урахуванням граничнодопустимої концентрації вогнегасної речовини

Необхідну кількість первинних засобів пожежогасіння визначають окремо для кожного поверху та приміщення, а також для етажерок відкритих установок. Якщо в одному приміщенні розміщені декілька різних за пожежною небезпекою виробництв, не відділених одне від одного протипожежними стінами, то всі ці приміщення забезпечують вогнегасниками, пожежним інвентарем та іншими видами засобів пожежогасіння за нормами найбільш небезпечного виробництва.

ВИСНОВКИ

Створення програмної моделі "Хижак-Жертва" для аналізу стохастичних процесів має великий потенціал у різних галузях науки та технологій. Ця модель дозволяє вивчати взаємодію між "хижаком" та "жертвою" і досліджувати їхні впливи від фінансів до біології та фізики.

Використання стохастичних моделей у фінансовому секторі допомагає у прогнозуванні цін на акції, опціони та інші фінансові інструменти, а також в управлінні ризиками та портфелями. У біологічних дослідженнях стохастичні процеси використовуються для моделювання еволюції популяцій, розподілу генетичних властивостей та поширення хвороб.

Поза науковими дослідженнями, модель "Хижак-Жертва" може бути корисною для освітних цілей та практичних застосувань, сприяючи розвитку нових методів аналізу та моделювання складних систем з випадковими змінами.

Для реалізації цієї моделі був створений програмний застосунок на мові програмування C# з використанням платформи Windows Forms. Цей застосунок дозволяє візуалізувати та аналізувати взаємодію між "хижаком" та "жертвою" за допомогою стохастичних процесів. У пояснювальній записці проєкту ретельно розглянуті всі аспекти розробки моделі "Хижак-Жертва", включаючи обґрунтування вибору технологій, опис використаних алгоритмів та їхню практичну реалізацію. Також надано аналіз можливих сценаріїв використання програмного продукту та оцінку його ефективності.

У пояснювальній записці проєкту ретельно розглянуті всі аспекти розробки моделі "Хижак-Жертва", включаючи обґрунтування вибору технологій, опис використаних алгоритмів та їхню практичну реалізацію. Також надано аналіз можливих сценаріїв використання програмного продукту та оцінку його ефективності.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Predator-Prey Simulation of the Lotka Volterra Model. [Веб-сайт]. URL: https://sites.google.com/site/biologydarkow/ecology/predator-prey-simulation-of-the-lotka-volterra-model/.
2. Lotka Volterra Model. [Веб-сайт]. URL: https://visualize-it.github.io/lotka_volterra/simulation.html/.
3. Lotka-Volterra model. [Веб-сайт]. URL: https://teaching.smp.uq.edu.au/scims/Appl_analysis/Lotka_Volterra.html/.
4. Lotka-Volterra model. [Веб-сайт]. URL: https://invozone.com/.
5. М. В. Добролюбова. Процедурна складова мови C# (навч. посібник). – «КПІ Ігоря Сікорського», 2021.
6. Huw Collingbourne. The Little Book Of C# Programming. – «Paperback», 2019 (англ. мовою).
7. Windows Forms Docs. [Веб-сайт]. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-8.0/.
8. James D. McCaffrey. [Веб-сайт]. URL: https://jamesmccaffrey.wordpress.com/2019/12/19/lotka-volterra-simulation-using-c/.
9. C# Docs. [Веб-сайт]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/.

ДОДАТОК А. Програмний код основної логіки ігрового застосунку

```
// GameKernel.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PPLV
{
    class EcoSystem
    {
        public double x; // prey
        public double y; // predators
        public double t; // time
        public double a; // growth rate of prey
        public double b; // death of prey
        public double c; // growth of predators
        public double d; // death of predators

        public EcoSystem(double x, double y, double t, double a, double b, double c,
double d)
        {
            this.x = x;
            this.y = y;
            this.t = t;
            this.a = a;
            this.b = b;
            this.c = c;
            this.d = d;
        }

        public void doEcoSystem()
        {
            double dx = (a * x) - (b * x * y);
            double dy = (c * x * y) - (d * y);
            int dt = 1;

            x = x + dx;
            y = y + dy;
            t = t + dt;
        }

        public double getPrey()
        {
            return x;
        }

        public double getPredators()
        {
            return y;
        }

        public double getTime()
        {
```

```
            return t;
        }

    }
}

// Dot.cs

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PPLV
{
    public class Dot
    {
        public int X { get; set; }
        public int Y { get; set; }
        public int Widthsize { get; set; }
        public int Heightsize { get; set; }

        public Graphics g;

        Random random = new Random();

        public Dot(Graphics graphics)
        {
            g = graphics;
        }
        public void Plot(Color color)
        {
            g.FillEllipse(new SolidBrush(Color.White), X, Y, Widthsize, Heightsize);
            X += random.Next(-5, 5);
            Y += random.Next(-5, 5);
            g.FillEllipse(new SolidBrush(color), X, Y, Widthsize, Heightsize);
        }

        public void Clear()
        {
            g.FillEllipse(new SolidBrush(Color.White), X, Y, Widthsize, Heightsize);
        }
    }
}

// Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PPLV
{
```

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        disableControls(this.Controls);
        osToolStripMenuItem.Text = SystemInfo.getOS();
        lANIPToolStripMenuItem.Text = SystemInfo.getLANIP();
        wANIPToolStripMenuItem.Text = SystemInfo.getWANIP();

        dataGridView1.ColumnCount = 3;
        dataGridView1.RowCount = 1000;
        dataGridView1.Columns[0].Name = "Час";
        dataGridView1.Columns[1].Name = "Жертви";
        dataGridView1.Columns[2].Name = "Хижаки";
    }

    private void openToolStripMenuItem_Click(object sender, EventArgs e)
    {
        try
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.Cancel)
            {
                return;
            }

            // Read the file lines
            string[] lines = System.IO.File.ReadAllLines(ofd.FileName);
            if (lines.Length < 6)
            {
                MessageBox.Show(this, "Файл не містить достатньо даних!",
                    "Message",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Warning);
                return;
            }

            // Assign the values to the textboxes
            textBox1.Text = lines[0];
            textBox2.Text = lines[1];
            textBox3.Text = lines[2];
            textBox4.Text = lines[3];
            textBox5.Text = lines[4];
            textBox6.Text = lines[5];
            textBox7.Text = lines[6];

            MessageBox.Show(this, "Файл відкрит!",
                "Message",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
                "Message",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
```

```
    }
}

private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
        if (sfd.ShowDialog() == DialogResult.Cancel)
        {
            return;
        }

        // Gather data from textboxes
        string[] lines = new string[6];
        lines[0] = textBox1.Text;
        lines[1] = textBox2.Text;
        lines[2] = textBox3.Text;
        lines[3] = textBox4.Text;
        lines[4] = textBox5.Text;
        lines[5] = textBox6.Text;
        lines[5] = textBox7.Text;

        // Write the data to the file
        System.IO.File.WriteAllLines(sfd.FileName, lines);

        MessageBox.Show(this, "Файл збережено!",
            "Message",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
            "Message",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
}

public void clearControls(Control.ControlCollection ctrlCollection)
{
    foreach (Control ctrl in ctrlCollection)
    {
        if (ctrl is TextBoxBase)
        {
            ctrl.Text = String.Empty;
        }
        else
        {
            clearControls(ctrl.Controls);
        }
    }
}

public void enableControls(Control.ControlCollection ctrlCollection)
{
    foreach (Control ctrl in ctrlCollection)
    {
        if (ctrl is TextBoxBase)
        {
```

```
            ctrl.Enabled = true;
        }
        else if (ctrl is ButtonBase)
        {
            ctrl.Enabled = true;
        }
        else
        {
            enableControls(ctrl.Controls);
        }
    }
}

public void disableControls(Control.ControlCollection ctrlCollection)
{
    foreach (Control ctrl in ctrlCollection)
    {
        if (ctrl is TextBoxBase)
        {
            ctrl.Enabled = false;
        }
        else if (ctrl is ButtonBase)
        {
            ctrl.Enabled = false;
        }
        else
        {
            disableControls(ctrl.Controls);
        }
    }
}

private void unlockToolStripMenuItem_Click(object sender, EventArgs e)
{
    enableControls(this.Controls);
}

private void clearToolStripMenuItem_Click(object sender, EventArgs e)
{
    clearControls(this.Controls);
}

private void restartToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Restart();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Environment.Exit(0);
}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show(this, "Predator-Prey Simulation\r\n"
                + "Kara Evgeny\r\n"
                + "2024",
                "Message",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
}
```

```
private void timer1_Tick(object sender, EventArgs e)
{
    dateToolStripMenuItem.Text = SystemInfo.getDate();
    timeToolStripMenuItem.Text = SystemInfo.getTime();
    stopwatchToolStripMenuItem.Text = SystemInfo.getStopwatch();
}

public static int cnt;

EcoSystem ecoSystem;

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        dataGridView1.DataSource = null;
        dataGridView1.Rows.Clear();
        dataGridView1.RowCount = 1000;
        dataGridView1.Refresh();

        chart1.Series[0].Points.Clear();
        chart1.Series[1].Points.Clear();

        cnt = 0;

        ecoSystem = new EcoSystem(Convert.ToDouble(textBox1.Text),
            Convert.ToDouble(textBox2.Text),
            Convert.ToDouble(textBox3.Text),
            Convert.ToDouble(textBox4.Text),
            Convert.ToDouble(textBox5.Text),
            Convert.ToDouble(textBox6.Text),
            Convert.ToDouble(textBox7.Text));

        timer2.Enabled = true;

    }
    catch (Exception ex)
    {
        MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
            "Message",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
}

private void button2_Click(object sender, EventArgs e)
{
    timer2.Enabled = false;
    dotPrey.Clear();
    dotPredators.Clear();
    tempPrey = 0;
    tempPredators = 0;
    Graphics g = panel1.CreateGraphics();
    g.Clear(Color.White);
}

Random random = new Random();

public static List<Dot> dotPrey = new List<Dot>();

public static List<Dot> dotPredators = new List<Dot>();
```

```
        public static int tempPrey = 0;

        public static int tempPredators = 0;

        private void timer2_Tick(object sender, EventArgs e)
        {
            try
            {
                ecoSystem.doEcoSystem();

                Graphics g = panel1.CreateGraphics();

                if ((int)Math.Round(ecoSystem.getPrey()) > tempPrey)
                {
                    for (int i = tempPrey; i < (int)Math.Round(ecoSystem.getPrey()); i++)
                    {
                        dotPrey.Add(new Dot(g)
                        {
                            X = random.Next(0, panel1.Width),
                            Y = random.Next(0, panel1.Height),
                            Widthsize = random.Next(8, 10),
                            Heightsize = random.Next(8, 10)
                        });
                    }
                }
                else if (tempPrey > (int)Math.Round(ecoSystem.getPrey()))
                {
                    for (int i = 0; i < Math.Abs((int)Math.Round(ecoSystem.getPrey()) -
tempPrey); i++)
                    {
                        if (i >= 0)
                        {
                            dotPrey.ElementAt(i).Clear();
                            dotPrey.RemoveAt(i);
                        }
                        else
                        {
                            break;
                        }
                    }
                }

                if ((int)Math.Round(ecoSystem.getPredators()) > tempPredators)
                {
                    for (int i = tempPredators; i <
(int)Math.Round(ecoSystem.getPredators()); i++)
                    {
                        dotPredators.Add(new Dot(g)
                        {
                            X = random.Next(0, panel1.Width),
                            Y = random.Next(0, panel1.Height),
                            Widthsize = random.Next(8, 10),
                            Heightsize = random.Next(8, 10)
                        });
                    }
                }
                else if (tempPredators > (int)Math.Round(ecoSystem.getPredators()))
                {
                    for (int i = 0; i <
Math.Abs((int)Math.Round(ecoSystem.getPredators()) - tempPredators); i++)
                    {
                        if (i >= 0)
```

```
                        {
                            dotPredators.ElementAt(i).Clear();
                            dotPredators.RemoveAt(i);
                        }
                        else
                        {
                            break;
                        }
                    }
                }

                dotPrey.ForEach(dot => { dot.Plot(Color.Green); });
                dotPredators.ForEach(dot => { dot.Plot(Color.Red); });

                tempPrey = (int)Math.Round(ecoSystem.getPrey());
                tempPredators = (int)Math.Round(ecoSystem.getPredators());

                dataGridView1[0, cnt].Value = ecoSystem.getTime().ToString("F1");
                dataGridView1[1, cnt].Value = ecoSystem.getPrey().ToString("F3");
                dataGridView1[2, cnt].Value = ecoSystem.getPredators().ToString("F3");

                chart1.Series[0].Points.AddXY(ecoSystem.getTime(), ecoSystem.getPrey());
                chart1.Series[1].Points.AddXY(ecoSystem.getTime(),
ecoSystem.getPredators());

                cnt++;
            }
            catch (Exception ex)
            {
                timer2.Enabled = false;
                MessageBox.Show(this, "Щось сталось..." + "\r\n" + ex.ToString(),
                    "Message",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
            }
        }
    }
}
```

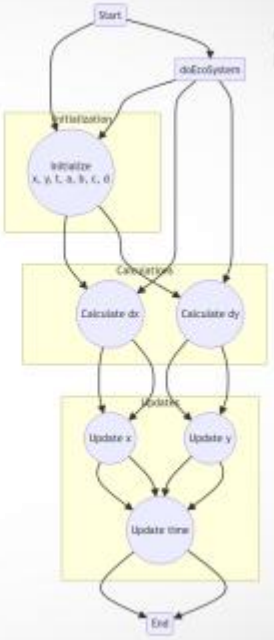
ДОДАТОК Б. Слайди мультимедійної презентації

Базовий алгоритм

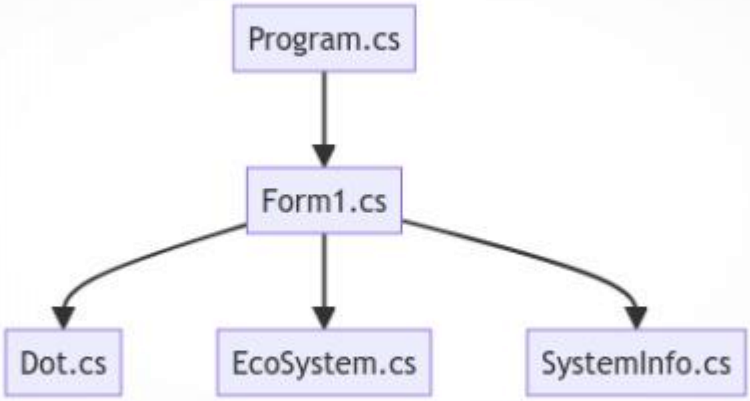
```
class EcoSystem
{
      public double x; // жертва
      public double y; // хижаки
      public double t; // час
      public double a; // швидкість зростання жертв
      public double b; // смертність жертв
      public double c; // зростання хижаків
      public double d; // смертність хижаків

     public void doEcoSystem()
     {
         double dx = (a * x) - (b * x * y);
         double dy = (c * x * y) - (d * y);
         int dt = 1;

         x = x + dx;
         y = y + dy;
         t = t + dt;
     }
{
```

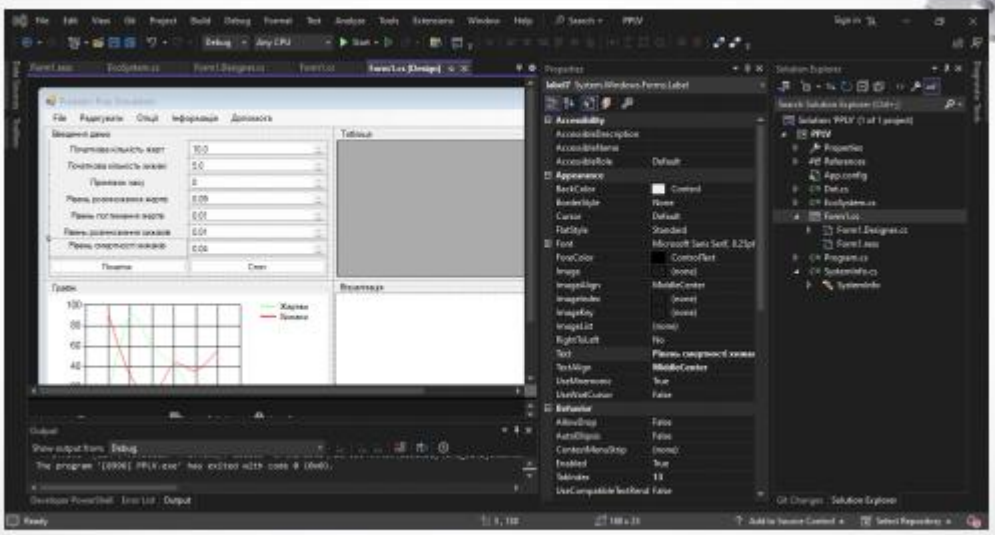


Архітектура застосунку



Модулі застосунку

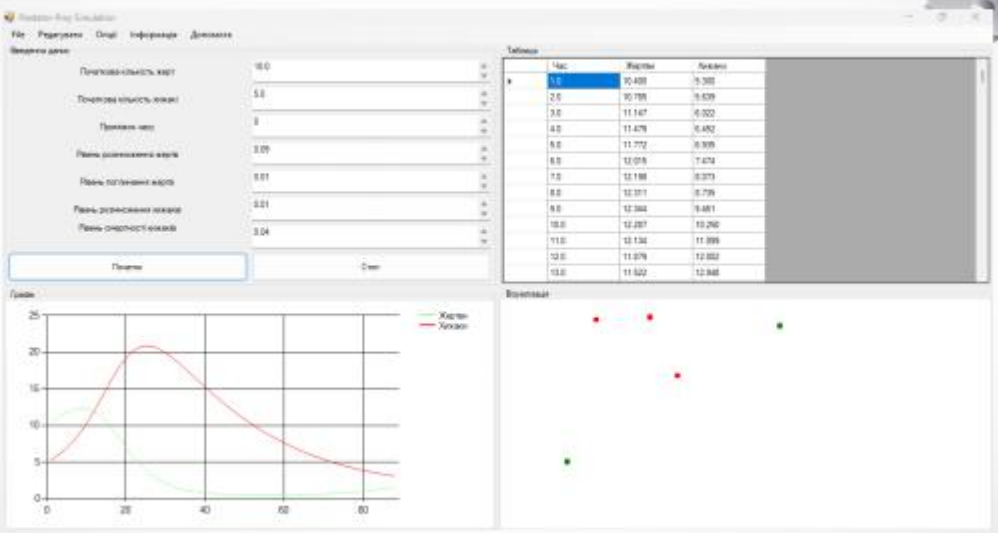
Процес розробки



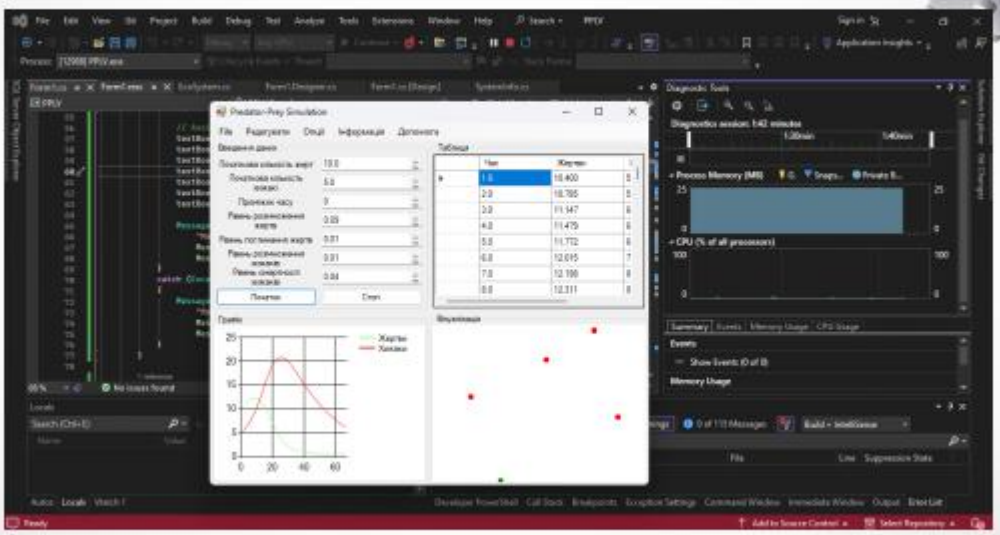
Особливості застосунку

- Адаптивний графічний інтерфейс
- Оновлення всіх даних в реальному часі
- Наявніть зберігання вхідних параметрів у файл
- Наявність таблиці з результуючими даними
- Наявність графіку з результуючими даними
- Наявність візуалізації з результуючими даними
- Супутній функціонал, наприклад, очищення всіх полів

Демонстрація симуляції



Тестування швидкодії



Дякую за увагу!

Кара Є. М., 4РП-07

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Кара Євгена Миколайовича

(прізвище, ім`я та по батькові)

Спеціальність: *121 «Інженерія програмного забезпечення»*

Освітня програма: *«Розробка програмного забезпечення»*

Тема дипломного проекту: *Розробка програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 77 сторінок. У пояснювальній записці описано етапи розробки програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів засобами C# та Windows Forms. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Кара Євген поступово та послідовно виконував всі етапи, проявляв ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Кара Євген під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Кара Євген показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS, C#, Windows Forms, MS PowerPoint, MS Visio та ін.*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові керівника дипломного проекту *Іванова Лілія Вікторівна*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Кари Євгена Миколайовича

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Іванова Лілія Вікторівна*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів*

Обсяг розрахунково-пояснювальної записки *76* сторінок

Обсяг графічної (презентаційної) частини *10* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі вивченню стохастичних процесів та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 10 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скриншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано програмну підтримку моделі "Хижак-Жертва", що дозволяє моделювати стохастичні процеси.*

Програма підтримка моделі має тонкі налаштування вхідних параметрів та деталізовану візуалізацію вихідних даних в реальному часі.

д) основні недоліки дипломного проекту

1. У роботі не представлено жодних блок-схем алгоритмів для створеної програмної моделі;

2. Анімація візуалізації відображає лише загальну кількість хижаків та жертв, але не відображає як хижаки переслідують жертв;

3. Наявні помилки оформлення тексту пояснювальної записки

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *Царьов Роман Юрійович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, ст. викладач, зав. кафедри комп'ютерної інженерії та інформаційних систем*

Підпис:

« 18 » 06 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ТА ЗВ'ЯЗКУ
Факультет інформаційних технологій та кібербезпеки
43997335
УКРАЇНА

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016353726

Дата перевірки:
12.06.2024 19:32:54 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 20:17:43 EEST

ID користувача:
100011688

Назва документа: 4РП-07_Кара_Є

Кількість сторінок: 46 Кількість слів: 6692 Кількість символів: 51536 Розмір файлу: 1.31 MB ID файлу: 1016157668

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.19%
Схожість

Найбільша схожість: 2.17% з Інтернет-джерелом (https://dspace.susu.ru/xmlui/bitstream/handle/0001.74/10988/2016_41.

7.19% Джерела з Інтернету | 687 ..... Сторінка 48

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | 7 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Кара Євген Миколайович,
здобувач освіти гр. 4РП-07, та

Іванова Лілія Вікторівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмної моделі "Хижак-Жертва" для вивчення стохастичних процесів» (автор роботи – Кара Є.М., керівник роботи – Іванова Л.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Кара Є.М. /

Керівник _____ / Іванова Л.В. /

«10» червня 2024 р.