МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

Інформаційно-аналітична система моніторингу загрози ракетних ударів

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402. 21910221

Виконав студент 4-го курсу, групи 402
_____ А.О. Савчук
« 19 » червня 2023 р.

Керівник: ст. викладач
_____І. С. Бурлаченко
« 19 » червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20___ р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Савчуку Антону Олександровичу.

1. Тема кваліфікаційної роботи «Інформаційно-аналітична система моніторингу загрози ракетних ударів».

Керівник роботи Бурлаченко Іван Сергійович, ст. викладач.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: огляд існуючих аналогів програм зі схожою тематикою використання, як і в системі моніторингу ракетних ударів, технологічні вимоги, вимоги до безпеки, вибір інструментів та технологій для реалізації програмного забезпечення.

Очікуваний результат роботи: Інформаційно-аналітична система моніторингу загрози ракетних ударів.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– розгляд існуючих аналогів програм зі схожою тематикою використання та їхні характеристики;

– технології, що використовуються у сфері програмування мовою C# та фреймворка .Net;
– огляд типових онлайн карт, які будуть використовуватися у дослідженні;
– вибір методів програмування, які будуть реалізований у роботі;
– аналіз використовування методу Монте-Карло у системі прогнозування ракетних ударів, та інтерпритація цього методу до нашої програми;
– опис архітектури системи та використання технологій (наприклад C#,.NET);
– розробка методів для взаємодії формул, карт та отримання даних для аналізу;
– розробка системи моніторингу ракетних ударів та аналізу отриманих результатів.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Аналіз організації робочого місця», «Вимоги до пожежної безпеки», «Вимоги до пристрою відображення інформації», «Встановити основні принципи техніки безпеки».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	А. Л. Боженко викладач	

Керівник роботи __ст. викладач, Бурлаченко І.С._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Савчук А.О. _____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» ___листопада___ 2022_ р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Інформаційно-аналітична система моніторингу загрози ракетних ударів

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	27.10.2022	27.10.2022	Виконано
2	Отримання завдання на виконання БКР	21.11.2022	21.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	05.12.2022	05.12.2022	Виконано
4	Отримання завдання на переддипломну практику	25.04.2023	29.04.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	11.05.2023	14.05.2023	Виконано
7	Виконання БКР: аналіз алгоритмів, огляд існуючих рішень, формування задачі, розробка інформаційної системи	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробив студент __А. О. Савчук_____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ ст. викладач, І. С. Бурлаченко _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_09_» _____12_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили

Савчука Антона Олександровича

Тема: «Інформаційно-аналітична система моніторингу загрози ракетних ударів»

Актуальність роботи – в умовах війни, багато людей думають про свою безпеку та все частіше звертаються до різноманітних застосунку, що проінформують їх про загрозу.

Об'єкт роботи – програмна система моніторингу ракетних ударів на базі WPF.

Предмет роботи – метод Монте Карло для прогнозування загрози ракетних ударів.

Метою бакалаврської кваліфікаційної роботи є моніторинг загрози ракетних ударів та розрахунок вірогідності попадання ракети за допомогою метода Монте-Карло.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі розкрито теоретичні засади для інформаційно-аналітичної системи моніторингу загрози ракетних ударів. Проаналізовано схожі додатки такі, як «Мапа Тривог» та «Deep State».

У другому розділі обрані методи та технології для створення застосунку. Було обрано фреймворк .Net та та мову програмування C#, та середовище розробки Visual Studio, що надає можливість гнучко та точно працюати з елементами інтерфейсу через методи та внутрішні функції, включая математичні.

У третьому розділі описані алгоритми і методи для реалізації застосунку. Було построєно модель та алгоритм програми у виді блок-схем. Визначено рівняння ідеальної ракети та обрано метод Монте-Карло, для статистичного розрахунку через випадкові події в кількості 100 шт.

У четвертому розділі описано проектування та програмну реалізацію розробленої системи. Створення інтерфейсу програми та функціональних частин за допомогою бібліотеки GMap.Net та GMap.WindowsForms.Net. Програмно обраховано формули для визначення області та часу польоту ракети та метода Монте-Карло за допомогою функцій та методів мови програмування C#.

Бакалаврська кваліфікаційна робота містить ___ сторінок, ___ рисунків, ___ джерел, ___ додатків.

Ключові слова: Ракетна загроза, моніторингова система, онлайн карта, фреймворк.

ABSTRACT

Bachelor's qualification work

of the student of 402 group of Petro Mohyla Black Sea National University

Savchuk Anton Oleksandrovych

Title: Information and analytical system for monitoring the threat of missile strikes

The relevance of work is in the minds of war, there are a lot of people thinking about their own safety, and more and more often they turn to various zastosunka to inform them about the threat.

Robotic object is a software system for monitoring missile strikes based on WPF.

The subject of the work is the Monte Carlo method for predicting the threat of missile strikes.

The method of bachelor's qualification work is monitoring the threat of missile strikes and the investigation of the probability of hitting a rocket for the help of the Monte Carlo method.

An explanatory note is made up of the entry, chotirokh rozdiliv, vysnovkiv and dodatkiv.

At the first division, a theoretical ambush was opened for the information-analytical system for monitoring the threat of missile strikes. Similar additions were analyzed, such as "Mapa Trivog" and "Deep State".

At another, they developed methods and technologies for setting up a zastosunka. The .Net framework, and the C# programming language, and the Visual Studio development environment, were developed, which allow it to be easily handled with interface elements through methods and internal functions, including mathematical ones.

The third section describes the algorithms and methods for the implementation of the zastosunka. The model and algorithm of the program were built in the form of block diagrams. The equalization of the ideal racket was determined and the Monte Carlo method was selected, for a statistical distribution through the vertical slopes in a quantity of 100 pcs.

The fourth section describes the design and software implementation of the divided system. Creation of a program interface and functional parts for the help of the GMap.Net library and GMap.WindowsForms.Net. Programmatically developed formulas for the purpose of the region and the time of using the Rocket and the Monte Carlo method for additional functions and methods of moving C # programming.

The thesis contains ___ pages, ___ figures, ___ sources, ___ appendices.

Keywords: Missile threat, monitoring system, online map, framework.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПК	– персональний комп’ютер
C++	– мова програмування сімейства сі
IDE	– набір програмних засобів, для розробки програмного забезпечення
ОС	– операційна Система
.NET	– платформа від Microsoft, яка дозволяє створювати програмні застосунки

ВСТУП

Комп'ютерна програма - це набір інструкцій, написаних у спеціальній мові програмування, призначений для виконання певних завдань на комп'ютері. Програма визначає послідовність операцій, які комп'ютер повинен виконати для досягнення певної мети.

Комп'ютерні програми можуть мати різні цілі і функції. Наприклад, програми можуть бути створені для обробки даних, виконання математичних операцій, керування апаратними пристроями, створення графіки, роботи з мережами, розробки веб-сайтів, ігор та багато іншого.

Програми можуть бути написані розробниками за допомогою різних мов програмування, таких як C++, Java, Python, JavaScript тощо. Після написання програми вона компілюється або інтерпретується виконавчим середовищем, що дозволяє комп'ютеру виконувати інструкції, вказані в програмі.

Комп'ютерні програми можуть бути розповсюджуватися у вигляді виконуваних файлів (які можна запустити на комп'ютері), веб-застосунків, бібліотек, плагінів та інших форматів. Вони грають важливу роль у різних сферах, включаючи бізнес, науку, освіту, розваги та інші.

Програми бувають прикладними – спрямованими на виконання поставлених завдань: написання текстів, створення креслень і таблиць. Це ті програми, які використовуються в повсякденній роботі за комп'ютером.

Системні комп'ютерні програми. Вони забезпечують підтримку комп'ютера і створюють основу роботи прикладних програм.

Системи програмування - це деякі програми для розробки, налагодження та впровадження програмних продуктів.

При виконанні роботи програми, комп’ютером обробляється певний набір інструкцій, якщо в цих інструкціях буде будь-який недолік, наприклад, помилка

або неправильне тлумачення даних, то застосунок буде працювати нестійкий або не буде запускатися взагалі, після чого показувати помилку.

Головна ідея програмування як професії, це рішення різних завдань, які представляють собою щось на зразок головоломки. Але це здасться простим тільки тоді, коли з’явиться розуміння коду та інших аспектів програмування.

Інструменти типового програміста найчастіше складаються з наступних речей:

– комп'ютер;

– операційна система;

– інтегроване середовище розробки (IDE): IDE - це програмне забезпечення, яке надає зручне середовище для написання, відлагодження і тестування програм;

– компілятори та інтерпретатори: Залежно від мови програмування, програміст може використовувати компілятори або інтерпретатори для перетворення свого коду в виконувану форму. Наприклад, для мови C++ використовують компілятори, а для Python – інтерпретатор;

– дебагери. Дебагери дозволяють програмістам виявляти та виправляти помилки у своєму коді. Вони дозволяють встановлювати точки зупинки, крокувати по коду та аналізувати значення змінних;

– бібліотеки та фреймворки: Багато програмістів використовують готові бібліотеки і фреймворки, які надають готові рішення для певних завдань. Наприклад, у веб-розробці популярні фреймворки, такі як Django для Python або Ruby on Rails для Ruby.

Актуальність**:** в умовах війни, багато людей думають про свою безпеку та все частіше звертаються до різноманітних застосунків, що проінформують їх про загрозу, але і військові що займаються нашою обороною потребують знань та методів для збереження життя людей.

Об'єкт роботи – програмна система моніторингу ракетних ударів на базі WPF.

Предмет роботи – метод Монте Карло для прогнозування загрози ракетних ударів.

Метою бакалаврської кваліфікаційної роботи є моніторинг загрози ракетних ударів та розрахунок вірогідності попадання ракети за допомогою метода Монте-Карло.

Відповідно до мети виділені наступні завдання бакалаврської кваліфікаційної роботи:

– проаналізувати проєкти які схожі на наш;
– визначити спосіб створення програми;
– розробити структурну та компонентну базу проєкту.

Створити діючу програму інформаційно-аналітичної системи моніторингу загрози ракетних ударів.

1 ТЕОРЕТИЧНІ ЗАСАДИ МОДЕЛЮВАННЯ СИСТЕМИ МОНІТОРИНГУ РАКЕТНИХ УДАРІВ

Комп'ютерний застосунок – це представлена в об'єктивній формі сукупність даних і команд, яка призначена на виконання пристроєм управління комп'ютером, а також застосунки є компонентами програмного забезпечення.

Кожний застосунок відповідає за конкретну ділянку роботи. Деякі застосунки допомагають користувачеві створювати графіку, текст і навіть музику. Деякі, абсолютно спокійно, допомагають користувачеві наводити порядок на жорсткому диску. Існує також вид програм, який допомагає користувачеві працювати в мережі Internet. Нижче наведена класифікація існуючого списку застосунків.

Першу категорію займають, звичайно ж, системні програми. Вони створюються для того щоб забезпечити нормальну роботу, настройку та обслуговування комп'ютера. До даного виду програм відноситься «ОС» і ряд допоміжних утиліт.

Операційна система – один з найголовніших компонентів, який зв'язує між собою апаратне забезпечення і програмне забезпечення. Без операційної системи комп'ютер не в змозі обробити не одну поставлену йому команду.

Утиліта – допоміжна комп'ютерна програма, спрямована на виконання спеціальних завдань і забезпечують доступ до параметрів, налаштувань або установкам, які без застосування утиліт не доступні. А також автоматизують процес деяких параметрів. Правильно підібрані утиліти можуть значно полегшити процес користування ПК.

Прикладні програми – це програми, робота яких спрямована на створення і обробку інформації. У світі таких програм випускається на багато більше, ніж системних.

Це залежить від того, що ці програми випускаються безпосередньо для користувача. А так як користувач по своїй натурі істота дуже примхлива, то і виробник в свою чергу намагаються надати безліч прикладних програм на вибір. До них відносяться:

– офісні програми;
– фінансові та бухгалтерські програми.

Офісні програми – це як правило, і є ті програми, заради яких користувачі і купують комп'ютер. Робота даних програм спрямована на створення і редагування документів (електронна таблиця, текст або зображення).

Найпопулярніший і найповніших пакет офісний пакет – це MICROSOFT OFFICE. До цього програмного продукту входять наступні пакети:

– текстовий редактор MICROSOFT WORD;
– електронна таблиця MICROSOFT EXCEL;
– підготовка презентацій MICROSOFT POWERPOINT;
– керування базами даних MICROSOFT ACCESS;
– поштовий клієнт MICROSOFT OUTLOOK.

Фінансові та бухгалтерські програми – це низка програм, спрямована на допомогу користувачу в підрахунку бухгалтерії як робочої, так і домашньої. Лідером в бухгалтерських програмах є комплект програм 1С бухгалтерія. Як показує статистика, домашня бухгалтерія набуває популярності. Так само в цю групу входять фінансові утиліти і електронні таблиці.

Професійні програми – це досить специфічна група програм, адже до неї можна віднести програми будь-якої групи. Все залежить від навиків і потреби в них невеликого кола людей. До цієї групи відносять:

– інструменти для програмування;
– системи проєктування (CAD).

До першого підвиду відносяться системи програмування, компілятори, а також багато іншого. Для програмістів такі програми невід'ємна частина роботи, а ось для домашнього використання вони не підходять.

Другий підвид програм спрямований на інженерні організації для розробки і креслення складних професійних креслень і блок-схем. До таких програм відноситься відома програма AutoCAD.

Наступний вид – розважальні та освітні програми. До них належать:

– освітні мультимедійні програми;
– енциклопедії;
– відеоігри.

Ще один вид застосунків, що користується великим попитом, – це мультимедійні програми. Мультимедіа – це сукупність усіх видів інформації з використанням технічно розвинених і програмних засобів, які об'єднують звук, текст, відео, графіку, фото в одному цифровому напрямку. Це визначення зародилося ще в ті часи, коли комп'ютер обробляє тільки текстові види інформації, коли використання інформаційних засобів комп'ютера не було поширеним.

До даного виду програм відносяться:

– програми для створення і редагування зображень;
– програми для створення та обробки звуку;
– програми плеєри та програми в’ювери;
– редактори тривимірної графіки.

Усі підтипи застосунків окрім програм-плеєрів можна віднести до професійних, адже вони потребують навичок, при цьому є здебільшого дорогими та витрачають багато ресурсів комп’ютеру.

Комп'ютерний експеримент - це експериментальне дослідження або моделювання, яке виконується за допомогою комп'ютерних програм або комп'ютерних симуляцій. Він передбачає використання обчислювальної техніки

для створення віртуальної моделі або системи, щоб дослідити її поведінку, провести чисельні розрахунки, виконати аналіз даних та вивести висновки.

Комп'ютерні експерименти широко застосовуються в різних галузях, включаючи науку, інженерію, медицину, фізику, економіку та інші. Вони дозволяють дослідникам віртуально відтворити складні фізичні процеси або системи, які можуть бути недоступні або небезпечні для прямого експериментування. Комп'ютерний експеримент дозволяє випробувати різні сценарії, змінювати параметри, виконувати повторні експерименти та аналізувати результати.

Основні переваги комп'ютерних експериментів включають швидкість виконання, повторюваність, здатність контролювати умови експерименту та здатність вивчати системи, які можуть бути недоступні для дослідження у реальному світі. Комп'ютерні експерименти також допомагають виявляти та аналізувати складні взаємодії та закономірності, що впливають на систему.

В нашому разі це буде програма для системи Windows, яка є інформаційно-аналітичною системою моніторингу загрози ракетних ударів за допомогою мови програмування C# та фреймворку .Net, тому буде використаний комп’ютерний експеримент для розробки програми інформаційно-аналітичної системи моніторингу загрози ракетних ударів.

1.1 Системи моніторингу загрози ракетних ударів у суспільних застосунках

За останні кілька років суспільність все більше користується застосунками як в своїх телефонах, так і на комп’ютерах. Ця сфера досягла небувалих результатів розвитку, тепер використовування гаджетів та комп’ютерних програм, це невід’ємна частина життя. Ця сфера торкнулася як простих людей, так і програмістів, та тенденція зараз набуває все більших розмахів. Ще вчора програми

розроблялися як частина ПО, а тепер це самостійні застосунки на різних мовах програмування та для різних платформ, які покривають сфери продуктів, навігації, покупок та багато іншого, і також сферу безпеки, що в наш час є дуже важливою.

Наш застосунок буде використовувати за основу багато інших рішень, які вже були створені, але для більш точного і кращого результату потрібно проаналізувати усі компоненти нашої майбутньої програми, та розглянути вже існуючі альтернативи.

Комп'ютерна програма (також відома як програмне забезпечення або просто програма) - це колекція інструкцій, написаних мовою програмування, які виконуються на комп'ютері. Вона представляє собою послідовність команд, які керують поведінкою комп'ютера та виконанням різних завдань.

У системному програмуванні, формальне визначення програми може варіюватись залежно від контексту. Однак, в загальному розумінні, програма може бути визначена як колекція інструкцій або операцій, які керують функціонуванням комп'ютерної системи або операційної системи.

Системна програма, зазвичай, виконує більш низько рівневі завдання, пов'язані з управлінням апаратними ресурсами комп'ютера, операційною системою та взаємодією з пристроями. Вона може включати драйвери пристроїв, ядра операційної системи, системні бібліотеки та інші компоненти, необхідні для ефективної роботи комп'ютерної системи.

Формальне визначення програми в системному програмуванні може бути пов'язане зі специфікацією інструкцій процесора, взаємодією з пам'яттю, операціями введення-виведення, обробкою переривань та іншими низько рівневими аспектами комп'ютерної системи.

Налагодження коду програми (англ. debugging) - це процес ідентифікації, виправлення та видалення помилок (багів) або неправильної роботи програмного коду. Коли розробник під час тестування або виконання програми виявляє

неправильну поведінку або отримує неправильні результати, налагодження використовується для встановлення причини проблеми та її виправлення.

Процес налагодження може включати кроки, такі як аналіз коду для виявлення потенційних помилок, використання спеціальних інструментів для відстеження виконання програми (наприклад, відладчиків), встановлення точок зупинки (breakpoints), спостереження за значеннями змінних, виведення додаткової інформації для аналізу та інше.

Під час налагодження розробник аналізує код програми, перевіряє його логіку, переконується в правильності алгоритмів та виявляє помилки, які можуть бути причиною неправильної роботи програми. Розробник може виправити помилки шляхом внесення змін до коду, покращення логіки, усунення неправильних умов або викликів функцій, та інших корекцій.

Налагодження є важливою складовою розробки програмного забезпечення, оскільки дозволяє виявити та виправити помилки, підвищуючи якість та надійність програми. Цей процес допомагає розробникам впевнитися, що програма працює правильно і задовольняє вимоги та очікування користувачів.

Комп'ютерні програми використовуються в багатьох сферах діяльності, як у бізнесі, так і в повсякденному житті.

Бізнес та управління: комп'ютерні програми застосовуються для автоматизації бізнес-процесів, обліку та аналізу даних, управління проєктами, управління відносинами з клієнтами (CRM), фінансового планування та багатьох інших аспектів бізнесу.

Охорона здоров'я: у медичних закладах використовуються програми для електронної медичної документації, управління пацієнтами, планування розкладу, діагностики та лікування, аналізу медичних даних та багато іншого.

Освіта: комп'ютерні програми застосовуються в освітніх закладах для електронного навчання, створення інтерактивних навчальних матеріалів, онлайн-курсів, управління навчальними планами та оцінками.

Розваги та ігри: комп'ютерні ігри, онлайн-платформи для потокової передачі відео, музичні програми, застосунки для читання книг та перегляду фільмів - все це приклади застосування комп'ютерних програм у сфері розваг та ігор.

Транспорт та логістика: у цій сфері застосовуються застосунки для керування логістичними процесами, відстеження вантажів, планування маршрутів, керування транспортними засобами та оптимізації логістичних операцій.

Соціальні мережі та комунікації: популярні соціальні мережі та месенджери, такі як Facebook, Instagram, WhatsApp, використовують комп'ютерні програми для обміну повідомленнями, спілкування, публікації контенту та підтримки зв'язку з друзями та колегами.

Фінанси: комп'ютерні програми застосовуються у банківській сфері для інтернет-банкінгу, електронних платежів, управління інвестиціями, аналізу фінансових ринків та торгівлі акціями.

Виробництво та промисловість: у цій сфері комп'ютерні програми використовуються для управління виробничими процесами, автоматизації виробництва, моніторингу та аналізу даних виробничих операцій.

Перші застосунки для комп'ютера з'явилися разом з розвитком ранніх комп'ютерних систем. Одним із перших застосунків був програмний пакет для наукових розрахунків, розроблений у 1940-х роках для використання на комп'ютерах Mark I та ENIAC. Однак на той час поняття "застосунок" не використовувалося в сучасному розумінні.

З появою персональних комп'ютерів у 1970-х та 1980-х роках стали з'являтися перші комерційні програми для широкої аудиторії. Такі програми, як електронні таблиці (наприклад, VisiCalc) та текстові редактори (наприклад, WordStar), стали першими успішними програмами для домашніх комп'ютерів.

З часом та з розвитком технологій створення програмного забезпечення тенденції у створенні програм для комп'ютера змінилися. Ось деякі з них:

– веб-програми: з розвитком Інтернету та веб-технологій стали популярними веб-програми, які працюють у браузері та доступні через Інтернет. Це дозволяє користувачам використовувати програми без необхідності інсталяції на свої комп'ютери;

– мобільні програми: з появою смартфонів та планшетів стала популярною розробка мобільних застосунків. Мобільні програми спеціально розроблені для роботи на мобільних пристроях та пропонують широкий спектр функцій та сервісів, включаючи ігри, соціальні мережі, банківську справу, охорону здоров'я, подорожі тощо;

– хмарні програми: з розвитком хмарних технологій стало можливим розробляти та використовувати програми, які зберігають дані та виконують обчислення у хмарі. Це дозволяє користувачам отримувати доступ до програм та даних з будь-якого пристрою з підключенням до Інтернету;

– штучний інтелект та аналіз даних: з розвитком штучного інтелекту та аналізу даних програми стали більш інтелектуальними та здатними обробляти великі обсяги даних. Приклади включають програми для машинного навчання, обробки природної мови, комп'ютерного зору та інших областей штучного інтелекту.

інтернет речей (IoT): зі зростанням числа підключених пристроїв та мереж IoT стали популярними програми, які взаємодіють із пристроями та збирають дані для аналізу чи керування.

1.2 C# та фреймворк .Net

Мова програмування C# використовується для розробки різноманітних типів програм, включаючи десктопні застосунки, веб-застосунки, мобільні застосунки, серверні програми, ігри та багато іншого. Вона має потужну екосистему інструментів, включаючи редактори коду, інтегровані середовища розробки

(наприклад, Visual Studio), фреймворки та бібліотеки, що сприяють швидкій та ефективній розробці програмного забезпечення.

C# це мова що має Сі-подібний синтаксис, а він у свою чергу є близьким до таких мов як C ++ і Java. Тому, якщо ви програмували на одній з цих мов, то опановувати C# буде набагато легше.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C++. Наприклад, C# підтримує всі правила об’єктно-орієнтованого програмування, такі як поліморфізм, успадкування класів, перевантаження операторів та статичну типізацію. Об'єктно-орієнтований підхід надає можливість вирішити завдання з побудови дуже великих, але в той же час дієвих, адаптивних застосунків. C# продовжує активно розвиватися, і з кожною оновленою версією з'являється все більше цікавих можливостей цієї мови програмування, такі як, наприклад, асинхронні методи, динамічне зв'язування, лямбда вирази та інші.

Коли ви чуєте назву C#, ви часто думаєте про технології платформи .NET. І, навпаки, дуже часто буває так, що говорять .NET маючи на увазі мову C#. Однак, хоча вони і пов'язані, невірно буде вважати їх одним і тим самим. Мова C# була створена для роботи з фреймворком .NET, але саме поняття .NET набагато ширше ніж у мови C#, що робить цей фреймворк самостійним.

Фреймворк .NET є популярною та потужною платформою для програмування з численними перевагами. Основні риси, які роблять фреймворк .NET привабливою для розробки програмного забезпечення, включають:

– кросплатформенність: Фреймворк .NET пропонує кросплатформенну підтримку через .NET Core. Це означає, що ви можете розробляти програми для різних операційних систем, таких як Windows, mac OS і Linux. Це дозволяє створювати кросплатформенні застосунки, які можуть працювати на різних пристроях та платформах;

– мовна розмаїтість: Фреймворк .NET підтримує багато мов програмування, зокрема C#, Visual Basic.NET, F# та інші. Це дає розробникам

можливість вибирати мову, яка найкраще відповідає їхнім потребам та вподобанням. Крім того, багатомовна підтримка дозволяє використовувати наявні знання та навички розробника в різних мовах програмування;

– багата бібліотека класів: Фреймворк .NET має велику бібліотеку класів, яка надає готові рішення для багатьох типових задач розробки програмного забезпечення. Це включає роботу з мережами, роботу з базами даних, обробку XML, шифрування, роботу з графікою та багато іншого. Використання цих готових класів дозволяє прискорити розробку та забезпечити високу якість програмного забезпечення;

– інструменти розробки: Фреймворк .NET має потужні інструменти розробки, такі як Visual Studio, яке надає розширені можливості для створення, налагодження та керування проєктами .NET. Ці інструменти надають розробникам багатофункціональне середовище з підсвічуванням синтаксису, автодоповненням, відлагодженням, управлінням версіями та багатьма іншими корисними функціями, що сприяють ефективній розробці програмного забезпечення;

– розширена безпека: Фреймворк .NET має вбудовані механізми безпеки, які допомагають уникати багатьох типових проблем безпеки, таких як переповнення буфера, недостатня автентифікація, атаки з використанням вразливостей та інші. Він надає можливості для реалізації механізмів автентифікації, авторизації, шифрування даних та інших безпекових заходів.

Варто відзначити, що .NET довгий час розвивався головним чином як платформа для Windows під назвою .NET Framework. В 2019 вийшла остання версія цієї платформи - .NET Framework 4.8.

.NET Updates Across Windows Versions								
.NET Framework / OS	Windows 8.1 and earlier	Windows 10						
		version 1507 (LTSC)	version 1607 & Server 2016	version 1703	version 1709	version 1803	version 1809 & Server 2019	version 1903
.NET 3.5	.NET Rollup	Windows Cumulative Update	Windows Cumulative Update	Windows Cumulative Update	Windows Cumulative Update	Windows Cumulative Update	.NET Update	.NET Update
.NET 4.6 through 4.7.2	.NET Rollup	Windows Cumulative Update	Windows Cumulative Update	Windows Cumulative Update	Windows Cumulative Update	Windows Cumulative Update	.NET Update	Not applicable
.NET 4.8	.NET Rollup	Not applicable	New .NET 4.8 standalone update	New .NET 4.8 standalone update	New .NET 4.8 standalone update	New .NET 4.8 standalone update	.NET Update	.NET Update

= No change
= New update
= Not applicable

Рисунок 1.1 – Діаграма змін у .Net Framework

У 2014 році Microsoft оголосила про розпочаття роботи над новою версією ASP.NET, відомою як ASP.NET Next. Ця ініціатива мала на меті перегляд та модернізацію фреймворку ASP.NET для розробки веб- застосунків.

У червні 2016 року був випущений перший стабільний реліз .NET Core 1.0. Це був значний крок у напрямку створення кросплатформної версії платформи .NET, яка може працювати на різних операційних системах. Випуск .NET Core був супроводжений відкриттям джерела коду платформи .NET. Microsoft перенесла розробку .NET Core на відкриту платформу GitHub, де розробники могли співпрацювати, вносити свої внески та приймати участь у процесі розробки. У травні 2019 року Microsoft оголосила про об'єднання .NET Core і Xamarin в єдину платформу під назвою ".NET 5". Це спрямовано на усунення фрагментації та створення єдиної кросплатформної платформи для розробки застосунків на .NET.

1.3 Огляд та аналіз наявних аналогів та публікацій

У сучасний час існує багато програм та систем схожих на нашу, але має зовсім іншу сферу використання, за винятку систем цивільної та військової

безпеки, тому згідно за аналізом подібних систем моніторингу загрози ракетних ударів їх можна поділити на два види, перший - це системи моніторингу в іграх, другий - системи цивільної та військової безпеки.

Щоб дізнатися чи є у моєї програми майбутнє, треба знати скільки систем та програм мають такий же функціонал, як і у мене. У цій частині ми розглянемо аналоги щоб дізнатися про їх переваги та недоліки, що покращить наш кінцевий продукт.

На початку краще розібрати системи цивільної та військової безпеки, які на зараз виконують велику роль у забезпеченні безпеки у різних країнах всього світу. Їх існує багато видів, але ми розглянемо основні з них:

– система попередження про ракетні удари (Missile Warning System - MWS) - це система, яка виявляє запуски ракет і відстежує їх траєкторії. Ця інформація використовується для визначення точного місцезнаходження ракет та розрахунку часу їхнього прилету. Наприклад, MWS використовується в літаках та гелікоптерах, щоб запобігти потенційним ракетним атакам;

– система раннього попередження про ракетну небезпеку (Early Warning System - EWS) - це система, яка виявляє запуски ракет на великих відстанях та надає попередження про потенційну небезпеку. Наприклад, EWS використовується для виявлення ракетних запусків з іноземних країн та надання попереджень національним військовим та цивільним органам;

– система розпізнавання та ідентифікації ракет (Missile Defense System - MDS) - це система, яка використовується для розпізнавання та ідентифікації типу та класу ракет. Ця інформація використовується для вибору найбільш ефективної стратегії знищення ракети. Наприклад, MDS використовується в системах протиракетної оборони для захисту від балістичних ракет;

– система наведення та ведення обстрілу (Fire Control System - FCS) - це система, яка використовується для наведення та ведення обстрілу зброєю на ціль.

FCS використовується в системах протиракетної оборони для наведення зброї на літаки або ракети, які можуть бути загрозою;

– система обміну даними та координації (Data Exchange and Coordination System - DECS) - це система, яка забезпечує обмін даними та координацію між різними системами моніторингу загрози ракетних ударів. Ця система забезпечує ефективну взаємодію між всіма компонентами системи моніторингу загрози та забезпечує швидкий обмін інформацією для прийняття рішень;

– система дії в разі нападу (Attack Response System - ARS) - це система, яка розроблена для швидкого реагування на загрозу ракетного нападу. ARS забезпечує збір та обробку інформації, розробку планів дії та надання відповідних команд для захисту від ракетних ударів.

Ці системи моніторингу загрози ракетних ударів використовуються в різних військових та цивільних застосуваннях, включаючи захист від балістичних ракет, протиракетну оборону, охорону національної безпеки та захист від терористичних актів.

Далі розглянемо більш детально системи моніторингу ракетних ударів на прикладі застосунків що вже існують, та визначимось з недоліками та перевагами щодо нашої програми.

1.3.1 Застосунок "Мапа Тривог"

"Мапа Тривог" – це карта, на якій можна побачити, в яких районах чи областях України зараз є повітряна тривога. У програмі можна побачити всю карту України, а також наближати її для більш детального перегляду. Області та райони, в яких зараз є повітряна тривога, забарвлені у червоний колір, можна переключити режим на список, і дивитися в яких регіонах є повітряна тривога у вигляді списку.

Рисунок 1.2 – Інтерфейс застосунку "Мапа Тривог"

Переваги:

– моментальне відображення тривоги;
– можливість детального розгляду мапи;
– відображення причини тривоги та час її тривалості;
– дуже простий інтерфейс.

Недоліки:

– мало взаємодій з мапою;
– має тільки інформаційну структуру;
– відображає тільки райони тривоги;
– немає інформації о швидкості та типі ракети.

1.3.2 Застосунок "Deep State"

"Deep State" - це карта, яка показує у реальному часі стан проведення бойових дій на території України, яку було створено групою волонтерів команди DeepStateUA team на початку російського вторгнення в Україну у лютому 2022 року з метою інформування про хід воєнних дій. У її основі лежить усім відома мапа усього світу від компанії Google – Google Map.

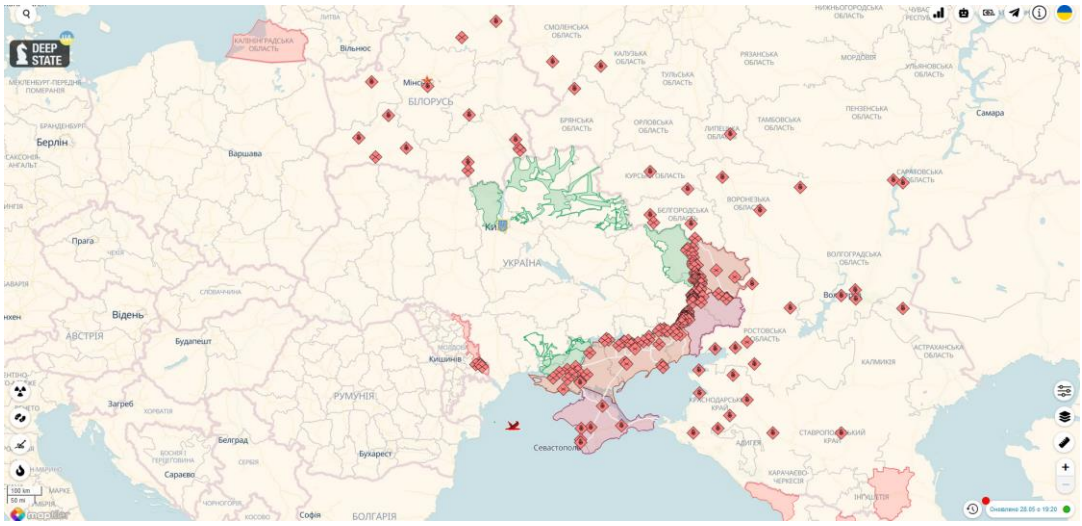

Рисунок 1.3 – Інтерфейс застосунку "Deep State"

Переваги:

– передає точне положення об’єктів та інших структур;
– ваказує з якої сторони проходять атаки;
– надає дані о відстані знаходження цілі;
– повністю зрозумілий інтерфейс;
– можливість детального розгляду мапи.

Недоліки:

– немає інформації о швидкості та типі ракети;
– мало взаємодій з мапою;
– має тільки інформаційну структуру.

1.4 Технічне завдання

Актуальність даної теми роботи полягає в тому, що створення програми надає студенту, чи іншому користувачу, простий шлях до вивчення параметрів різних ракет та вираховування їх можливих цілей ураження виходячи з дальності ,та ця програма в перспективі допоможе учням військових вузів. Дані з середовища

будуть оновлюватися поточно без збереження, що дає можливість в економії пам’яті та сприяє швидкодії програми.

Мета бакалаврської кваліфікаційної роботи полягає у покращенні вивчення та моніторингу ракетних ударів завдяки вхідним параметрам.

Завдання: Розробити застосунок на основі C# Net Framework для моніторингу ракетних ударів.

Для досягнення поставленої задачі необхідно виконати наступні завдання:

– проаналізувати аналоги у цій області;
– визначити технологію створення застосунку для Windows;
– розробити алгоритм за яким виконується програма;
– створити програму.

Висновки до розділу 1

Все частіше питання безпеки становить велику важливість серед людей, що насамперед залежить від навичок тих чи інших програм чи людей, тому їх підготовка є важливою частини безпеки. У планах розробити просту але діючу програму, що не потребує великих системних потреб. Мова програмування C#,що використовується у розробці, стоїть на 5-тому місці по популярності, що надає впевненості у надійності продукту та надає подальшої можливості для вдосконалення чи перенесення на інші платформи такі як Linux,Android,IOS.

Розібравши фреймворк .Net, було визначено його переваги, такі як: кросплатформеність, мовна різноманітність, багата бібліотека класів, розширена безпека та багата кількість інструментів для розробки. Важливо зауважити що фреймворк навіть зараз має програмну підтримку та майже кожен рік отримує новлення до свого пакету, що розширює функіонал та практичність, наприклад цього року було добавлена така функція, як розрив строки, яка робить код більш читабельним та легшим у виправленні помилок.

Проаналізувавши схожі програми та фізичні розробки, такі як різни системи протиповітрянної безпеки було рішено зупинитись на програмних розробках та вилучено чіткі потреби до створення програми та обрано інтерфейс, який у подальшому буде опрацьований за допомогою додаткових бібліотек сумісних с Windows Forms.

Тому можна зробити висновок, що програмні засоби та технології ,які було обрано нами, є кращим рішенням щодо створення нашого застосунку, через свою гнучкість, сумісність з іншими мовами програмування та середами розробки, та через великий інструментарій з налаштування інтерфейсу.

2 МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ СТВОРЕННЯ ЗАСТОСУНКУ

Інформаційна технологія - це процес який використовує декілька методів і засобів збору, передачі і обробки даних, щоб отримати інформацію про стан процесу, об’єкта чи явища. Мета інформаційної технології полягається в виробництві інформації. Аналіз проводиться безпосередньо людиною і прийняття на основі цього аналізу рішення, з виконання будь якої дії.

Впровадження персонального комп'ютера в інформаційну сферу та використовування телекомунікаційних засобів зв'язку почали новий етап розвитку та поширення інформаційної технології у світі. Нова інформаційна технологія - це технологія з так званим «дружнім» інтерфейсом роботи користувача, який використовує персональні телекомунікаційні засоби і комп'ютери. Нова інформаційна технологія базується на таких базових принципах:

– режим роботи з комп'ютером у діалоговому вікні;
– інтегрованість з іншими програмами;
– гнучкість щодо зміни даних і постановки завдань.

В якості інструментів в інформаційних технологіях використовують найпоширеніші види програмних продуктів: видавничі системи, текстові процесори, електронні таблиці, електронні календарі, системи управління базами даних та інформаційні системи функціонального призначення.

2.1 Засоби розробки застосунків на мові C#

C# – мова програмування, що поєднує у собі концепції об'єктно-орієнтованого та аспектно-орієнтованого мов програмування. Ця мова розроблена у 1998-2001 роках невеликою компанією інженерів під керівництвом Андерса Хейлсберга на території компанії Microsoft як мова розробки програмних

застосунків для платформи Microsoft .NET та .NET фреймворк. Компілятор з мови C# входить до стандартної установки самої платформи, тому створювати і компілювати програми на ньому можна навіть без сторонніх програмних застосунків, таких як Visual Studio.

C# є мовою з Сі-подібним синтаксисом та має багато спільного з C++ і Java. Взявши багато від попередніх мов - C#, спирається на практику їх використання, виключаючи деякі моделі поведінки, що погано зарекомендували себе при розробці програм та систем. Тому, C# не підтримує множинне успадкування класів.

C# розроблявся як мова прикладного рівня програмування для CLR, тому функціонал мови напряму залежить від можливостей самої CLR. Це стосується системи типів C#, що відображає FCL. Наявність або відсутність виразних особливостей мови обумовлюється тим, чи може ця мовна особливість бути переведеною в відповідності до конструкції CLR. Тому з розвитком CLR, C# став набагато продуктивнішим і потужнішим. По прогнозам такого росту слід було чекати і в подальшому часі. Однак цей зв’язок був порушений з виходом C# 3.0, що є розширення до мови, але тепер він не спирався на розширення які відносяться до платформи .NET. CLR надає можливість для C#, як і всім іншим мовам що зв’язані з .NET, якої позбавлені інші мови програмування. Наприклад, прибирання так званого сміття (це код який використовує пам’ять, але не робить нічого) не передбачена у самому C#, а проводиться CLR для програм, написаних на цій мові програмування так само, як для програм на інших мовах.

C# має дуже багатий, але і простий та зручний у вивченні, функціонал. Характерні фігурні дужки при написанні коду на мові C# миттєво пізнаються усіма, хто знайомий з мовами Сі. Розробники, які знайомі з будь-якою з цих мов, зазвичай дуже швидко розбираються як працювати з C#. Синтаксис C# виправляє багато складностей мови C ++, але при цьому є потужним інструментом програміста з таким ж функціями. C# підтримує стандартні методи і типи, які надають дуже високий рівень безпеки і продуктивності, а також різні ітератори, що дозволяють

визначати в класах колекцій власну поведінку, це можна легко застосувати в коді. Вирази LINQ створюють мовну конструкцію, що є на наш час дуже простою, у порівнянні з іншими мовами, та використовується для строго типізованих запитів.

C# є об'єктно-орієнтованою мовою, а це означає що вона підтримує інкапсуляцію, успадкування і поліморфізм. Це всім відомі правила ООП. Всі змінні і методи, включаючи головний метод під назвою Main,в якому найчастіше відбувається виклик методів класу та створення екземплярів цих класів, який представляє собою місце входу у застосунок, інкапсулюється в визначення класів. Клас успадковується від одного батьківського класу, але може реалізовувати будь-яке число методів та інтерфейсів, як своїх так і успадкованих. Методи що були перевизначені, скасовують методи батьківського класу, та містять ключове слово override, щоб виключити випадкове перевизначення. У мові C# структура коду схожа на полегшений вид класу: це тип, що знаходиться в стеці, який реалізує інтерфейси та методи, але не підтримує множинне спадкування.

Крім цих основних принципів ООП, C# пропонує ряд нових, навіть інноваційних, мовних конструкцій, що покращують розробку програмних частин та повністю цілих застосунків:

– інкапсульовані частини методів, що називаються делегатами, які дозволяють реалізувати безпечне повідомлення про події;

– властивості, що грають роль функції акцесорів для закритих змінних-членів;

– атрибути, які надають метадані про типи змінних під час компілювання;

– коментарі для XML-документації у середині комірки;

– linq, який надає вбудовані можливості для створення запитів до різних джерел даних.

Архітектура платформи .NET Framework. Програми з використанням мови C# виконуються на платформі .NET Framework, на вбудованій частині системи Windows, яка включає у себе віртуальну систему виконання коду, звану

підтримкою загальномовного середовища виконання програми, і універсальний набір бібліотек та класів. Середа CLR є аналогом комерційної реалізації міжнародного стандарту Common Language Infrastructure, що служить базою щодо створення середовищ виконання і розробки, які надають можливість спільно використовувати різні мови програмування та підключення до них бібліотек.

Код, створений за допомогою мови програмування на мові C#, виконується у проміжний мові, що відповідає потребам CLI. Код, написаний мовою IL та ресурси, разом з піксельними малюнками та рядками, зберігаються на диск у форматі початкового файлу (найчастіше він має розширення .exe або .dll). Подібний файл називають складанням. До збірки входить маніфест із даними про типи, версії, вимоги безпеки, мови і регіональні параметри для даної збірки.

У процесі виконання програми C# Середа CLR завантажує збірку та виконує низку дій в залежності від відомостей, що збережені у маніфесті. У разі виконання всіх вимоги безпеки, середа CLR здійснює JIT-компіляцію з коду на мові IL в інструкції машинної мови. Також середовище CLR виконує інші операції. До них належать: автоматична збірка сміття, обробка винятків та управління ресурсами. Код, що виконується середовищем CLR, інколи називають "керованим кодом". Таку назву застосовують задля підкреслення відмінностей даного підходу від "некерованого коду", що відразу компілюється у машинну мову для певної системи. На наступній схемі зображені зв'язки між файлами вихідного коду C# з бібліотеками класів .NET Framework, збірками та середовищем CLR, що існують у процесі зборки і під час її виконання.

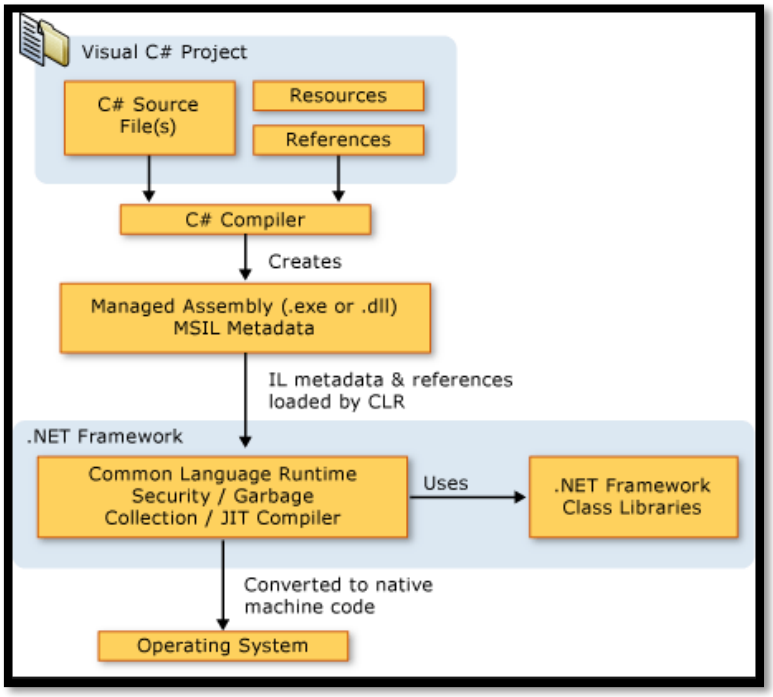

Рисунок 2.1 – Робота з фреймворком .Net

Крім стандартних служб, платформа .NET Framework також має свою велику бібліотеку до якої належить трохи більше за 4000 класів, організованих у просторі імен, які надають різні функції що є корисними та спрощують роботу для всіх операцій: починаючи з введення і виведення інформації з файлів до управління рядками при проходженні аналізу XML і управляючих елементів Windows Forms. Насамперед застосунок C# широко використовує сукупність класів .NET Framework для простих завдань по підключенню.

2.2 Вибір середовища для програмування

Visual Studio - це інтегроване середовище розробки (IDE), розроблена компанією Microsoft. Воно надає розробникам широкий набір інструментів і можливостей для створення різних типів програмного забезпечення, включаючи десктопні застосунки, веб-застосунки, мобільні застосунки, хмарні рішення та інші.

.

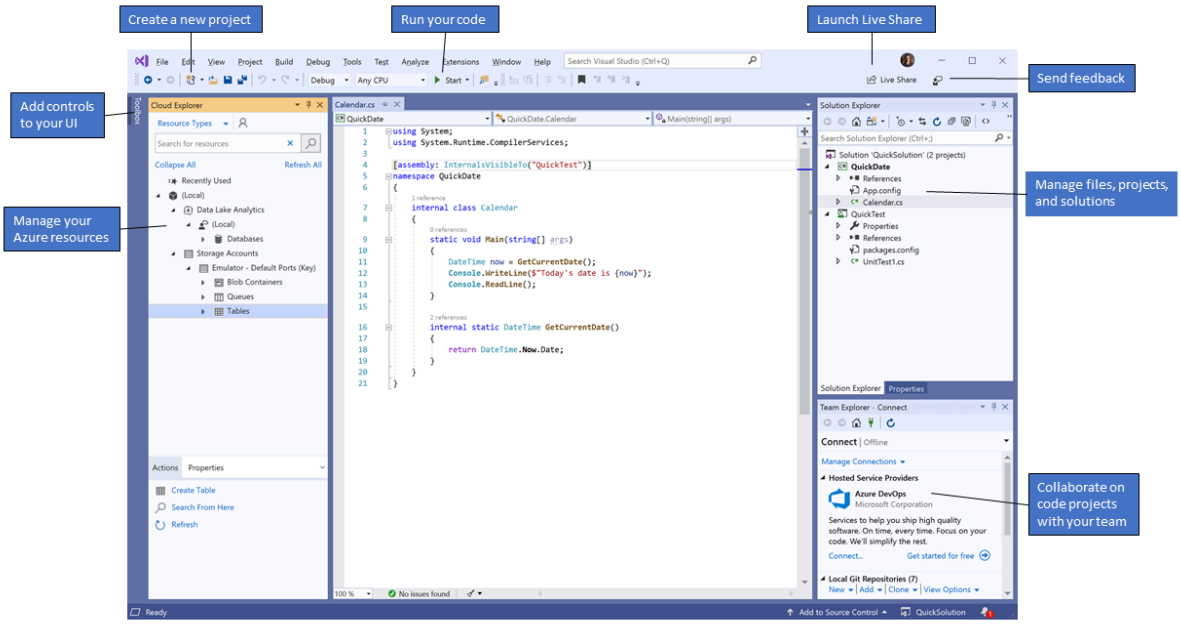

Рисунок 2.3 – Елементи Visual Studio

Існує три випуски Visual Studio 2019: Community, Professional і Enterprise. Ми працюємо з версією Professional.

Але чому ми обираємо саме Visual Studio? У цього середовища є багато плюсів, а саме:

– кодування та редактор: Visual Studio має потужний текстовий редактор з підсвічуванням синтаксису, автодоповненням коду, перевіркою помилок та багатьма іншими функціями, які полегшують процес написання коду;

– відлагодження: Візуальна студія надає розширені можливості для відлагодження програм, що дозволяє розробникам крокувати по коду, спостерігати значення змінних, аналізувати стек викликів та виконувати інші дії, щоб знайти та виправити помилки;

– управління проєктами: Visual Studio дозволяє створювати та керувати проєктами різних типів. Він надає шаблони проєктів, інструменти для додавання, видалення та керування файлами проєкту, а також можливості налаштування залежностей та конфігурацій проєкту;

– інтеграція з Git та іншими системами контролю версій: Visual Studio підтримує інтеграцію з популярними системами контролю версій, такими як Git, TFS (Team Foundation Server) та SVN (Subversion). Це дозволяє розробникам керувати версіями свого коду, виконувати коміти, злиття та інші операції безпосередньо з інтерфейсу Visual Studio;

– візуальне проєктування інтерфейсу користувача: Visual Studio має вбудовані інструменти для візуального проєктування інтерфейсу користувача для десктопних застосунків, мобільних застосунків та веб-сторінок. Він надає можливість перетягування та випадання елементів у користувацькому інтерфейсі, налаштування властивостей елементів та перегляд результатів в реальному часі;

– розширюваність: Visual Studio дозволяє розробникам створювати власні розширення та застосунки, що розширюють функціональність середовища. Це дозволяє адаптувати Visual Studio до власних потреб та розробляти інструменти, які полегшують роботу розробників.

Усі ці засоби та інші можливості Visual Studio у сукупності дають нам можливість гнучкого редагування коду та цілу базу допоміжних засобів. Саме тому багато хто з програмістів використовує Visual Studio і ми не виключення.

2.3 Технологія створення коду та зв’язку між компонентами програми

Windows Forms - це фреймворк для розробки графічних інтерфейсів користувача (GUI) для Windows- застосунків у середовищі розробки .NET. Він надає програмістам інструменти для створення віконних застосунків з використанням готових елементів управління, таких як кнопки, текстові поля, списки, таблиці, меню та інші.

Windows Forms працює на основі об'єктно-орієнтованої парадигми програмування та використовує мову C# або інші мови .NET, такі як VB.NET, для написання коду. Він використовує подійно-орієнтовану модель програмування, де

дії користувача, такі як натискання кнопок або введення тексту, викликають відповідні події, на які можна реагувати кодом програми.

Основні компоненти Windows Forms включають форми (Form), елементи управління (Control), діалогові вікна (Dialog), макети (Layout), графічні елементи (Graphics) та багато інших. Ви можете створювати власні елементи управління, наслідуючи від базових класів або використовуючи сторонні компоненти, що розширюють функціональність.

Завдяки Windows Forms ви можете створювати різноманітні Windows-застосунки, включаючи редактори тексту, калькулятори, системи управління базами даних, графічні програми, застосунки для обробки даних та багато інших. Він надає багато можливостей для налаштування зовнішнього вигляду, взаємодії з користувачем та обробки подій, що робить його потужним інструментом для розробки Windows- застосунків.

Windows Forms надає простий і зрозумілий спосіб розробки графічного інтерфейсу користувача. Його API є інтуїтивно зрозумілим, має багатий набір готових елементів управління (кнопки, текстові поля, списки тощо), і легко використовується навіть для початківців у програмуванні. Ще він має великий набір готових елементів управління, які можна використовувати для створення багатофункціональних застосунків. Це включає кнопки, тексти, таблиці, списки, вкладки, діалогові вікна та багато інших елементів.

Windows Forms надає розширені можливості налаштування зовнішнього вигляду і стилю застосунків. Ви можете змінювати кольори, шрифти, розміри, стилі, фони тощо, щоб створити унікальний інтерфейс, який відповідає вашим потребам та бренду.

Windows Forms має розширену підтримку подій та обробників подій, що дозволяє взаємодіяти з користувачем та реагувати на дії, такі як натискання кнопок, введення тексту тощо. Це дозволяє створювати динамічні інтерфейси та реалізовувати функціональність відповідно до дій користувача.

Також ви можете перевіряти правильність введення даних, обмежувати значення, встановлювати правила форматування тощо, щоб забезпечити відповідність введених даних заданим критеріям, також Windows Forms добре інтегрується з іншими технологіями Microsoft, такими як бази даних SQL Server, WCF (Windows Communication Foundation), WPF (Windows Presentation Foundation) тощо. Це дозволяє створювати повноцінні застосунки, які взаємодіють з різними системами та ресурсами.

Програмісти розглядають Microsoft .NET найчастіше як .NET Framework class library. Для того, щоб отримати точну картину про ширину і глибину .NET Framework class library, слід уявити MFC на порядок більше. Щоб зробити протиріччя в позначеннях мінімальними і надати організацію багатьом класам, .NET Framework class library розділена на залежні розділи по іменах. Головний розділ, System, визначає основні типи даних, що використовуються всіма програмами написаними на .NET.

Застосунки, які використовують Windows Forms, використовують класи сімейства WinForms. Цей розділ включає у себе такі класи, як Form, який моделює поведінку вікон та форм вашого застосунку, Menu, який і є цим меню, Clipboard, який надає можливість програмі на Windows Forms використовувати буфер обміну. Він також містить дуже велику кількість класів, які надають засоби управління, наприклад: Button, ListView, MonthCalendar та інші. Ці класи можуть бути включені в застосунок двома способами, це або з використанням тільки імені класу до якого вони відносяться, або з використанням повного імені елемента, наприклад: System.WinForms.Map().

В основі майже кожної програми, написаної із застосуванням Windows Forms, лежить похідний клас від System.WinForms.Form. Зразок цього класу представляє головне вікно програми. System.WinForms.Form має безліч властивостей і методів, які мають багатий програмний інтерфейс до форм. Для визначення розмірів клієнтської області форм у Windows необхідно викликати

функцію API GetClientRect. У Windows Forms необхідно використовувати властивості ClientRectangle або ClientSize.

Застосунки, засновані на Windows Forms, які використовують кнопки, списки та інші типи компонентів Windows, застосовують класи управління System.WinForms, значно спрощують процес програмування управління. Наприклад, можна без проблем створити стилізовану кнопку з зображенням у вигляді фону. Для цього необхідно включити потрібне зображення в об'єкт System.Drawing.Bitmap і призначити його властивості кнопки Background Image. Також наявна можливість управління кольором, наприклад – налаштування кольору фону текстового поля. Потрібно лише присвоїти колір властивості BackColor.

Іншим важливим "будівельним" блоком програми, який використовує Windows Forms є клас System.WinForms що називається Application. Цей клас містить метод Run, який і завантажує програму та показує вікно з виконаним кодом.

Багато які класи мають методи що називаються віртуальними та які можна перевизначити. Наприклад, System.WinForms.Form містить метод OnPaint, який викликається, коли клієнтська область цієї форми потребує оновлення і він є віртуальним. OnPaint – один з множини віртуальних методів, який користувач може перевизначити в похідному класі, це робиться для формування інтерактивних форм.

Інша важлива частина програмування з використанням Windows Forms – це механізм, який використовується формами для відповіді на введення інформації в меню, засобів управління і інших елементів застосунку. Традиційні Windows-застосунки можуть оброблятися повідомленням WM_COMMAND і WM_NOTIFY, за допомогою події Windows Forms. У C# і на інших мовах, які підтримують .NET події – члени типу похідного класу нарівні з методами, полями і властивостями. Практично всі управляючі класи Windows Forms (а також і багато некерованих класів) створюють події. Наприклад, коли ви натискаєте кнопку, вона створює

подію Click. Форма, яка повинна відповісти на натискання кнопки може використовувати наступний код, щоб з'єднати кнопку на обробником події Click.

```
MyButton.Click += new EventHandler (OnButtonClicked);

...

private void OnButtonClicked (object sender, EventArgs e)
{
    MessageBox.Show ("Click!");
}
```

Рисунок 2.4 – Приклад використання Event Handler

EventHandler – це спеціальний обробник подій, який виконує метод OnButtonClicked, коли MyButton створює подію Click. Перший параметр OnButtonClicked ідентифікує об'єкт, що викликав подію. Другий параметр здебільшого не має не сенсу для події Click, але використовується деякими іншими типами подій, щоб передати додаткову інформацію.

Для покращення інтерфейсу програми, було прийнято рішення використати додаткові бібліотеки, такі як GMap.Net та GMap.NET.WindowsForms, що надають можливість працювати з онлайн мапами від будь якого провайдера, наприклад GoogleMap та інш. Це на сам перед надає можливість роботи з збільшенням масштабу та більш детального розгляду нашого застосунку, для більш точного встановлення місця запуску ракети.

Бібліотека GMap.Net – це бібліотека для роботи з картами різних постачальників, для зручної роботи з мапою. Зручність полягає у включених до бібліотеки спеціальних класів та методів, які надають можливість створення точок з координатами X та Y, де X - широта, а Y - довгота.

Бібліотека GMap.NET.WindowsForms – це інструмент для сполучення GMap.NET та фреймворку Windows Forms, за допомогою спеціальних інструментів. Вона передбачає додавання до панелі інструментів Windows Forms,

спеціального вікна GMapControl, де можна побачити різні види мап, залежно від постачальників. Це зручний інструмент для проєктування власної програми, в основу якої входить сама інтерактивна мапа та інші елементи фреймворку Windows Forms.

Розглянемо декілька методів, що дозволяють працювати з GMap, на прикладі нашої програми.

Метод gMap_Load, це метод для ініціалізації графічного елемента на якому буде відображатись карта, як можна побачити в ньому задані параметри для цього елемента, розглянемо кожен з них:

– gMap.DragButton = MouseButtons.Left – у цьому параметрі вказується якою кнопкою буде виконуватися перетаскування картки;

– gMap.MaxZoom = 18 – параметр у якому вказується максимальний зум для картки (можливий діапазон зуму від 1 до 18);

– gMap.MinZoom = 2 - параметр у якому вказується мінімальний зум для картки;

– gMap.Zoom = 13 параметр, який визначає який буде зум при відкритті програми;

– gMap.MapProvider=GMap.NET.MapProviders.GMapProviders.GoogleMap – параметр у якому вказується картки від якого постачальник будуть використовуватись (У даному випадку від Google);

– GMaps.Instance.Mode = AccessMode.ServerOnly – параметр у якому визначається в якому режимі будуть працювати карти(Онлайн режим, режим кешу, або змішаний режим);

– gMap.Position = PositionCenter – параметр у якому вказується на якій позиції карта має бути відображена при старті програми, їй передається об'єкт спеціального класу PointLatLng, що входить до бібліотеки GMap.NET, і служить для збереження точки, форматі координат широти та довготи.

Висновки до розділу 2

У цьому розділі було розглянуто основні поняття для вибору предметної області та інструментів що допоможуть з цим. Було визначено які методи програми треба описати, через які конструктори посилатися на об’єкти інтерфейсу, також розглянуті внутрішні елементи Windows Forms та додаткові функції бібліотеки GMap.Net, такі як ініціалізація онлайн мапи та робота з нею.

Було проаналізовано роботи з методами класів, вбудованих в фреймворк .Net, вбудовані функції бібліотеки GMap.Net, ткож опрацьовано методи роботи з обробниками подій, таких як MouseClick() та інш.

Тако було покращено навички роботи у середовищі Visual Studio, з компілятором та сторінкою обробника помилок, що дуже допомагає у знаходженні та виправленні як логічних так і синтаксичних помилок у коді, наприклад відсутність ком чи дужок, або неправильне визначення функції чи посилання до об'єктів класу які не приймають в якості параметру текстове значення, а тільки чисельне.

Для програмування було обрано мову C# разом з середою розробки Visual Studio та Windows Forms, та фреймворк .Net Framework що використовує бібліотеки .Net, та бібліотека GMap.Net.

3 МОДЕЛЮВАННЯ СИСТЕМИ МОНІТОРИНГУ РАКЕТНИХ УДАРІВ

Програмна реалізація це добре, але без алгоритмів та формул мало що вийде зробити. Головною складовою числень є формули, а на сам перед рівняння.

Рівняння - формула, зовнішня (верхня) зв'язка якого є бінарне відношення рівності. Однак важлива особливість рівняння полягає також в тому, що входять до нього символи поділяються на змінні і параметри Наприклад, x 2 = 1- є рівнянням, де x - змінна. Значення змінної, при яких рівність істинно, називаються корінням рівняння: в даному випадку такими є два числа 1 і -1. Як правило, якщо рівняння на одну змінну не є тотожність, то корені рівняння є дискретна, найчастіше кінцева (можливо і пуста) множина.

Система моніторингу обумовлене використанням формул. В нашому випадку найголовніша фізична величина це швидкість та дальність польоту, що на пряму мають вплив на кінцеву ціль нашого проєкту.

3.1 Функціональна модель застосунку

Програміст - це людина, яка займається розробкою комп'ютерних програм. Він створює програми, які виконують певні завдання або розв'язують конкретні проблеми, використовуючи різні мови програмування та інструменти розробки. Найважливіше вміння програміста, це – вміння продумати та скласти алгоритм дій, для точної реалізації програми. Чим краще ти розумієш алгоритм, тим швидше та точніше буде виконане завдання.

Функціональна модель (також відома як функціональна архітектура) - це підхід до проєктування програмного забезпечення, де програма розбивається на набір функцій або функціональних блоків, які виконують певні операції. У цій моделі фокус зосереджений на виконанні функцій та обміні даними між ними.

Основні принципи функціональної моделі включають:

– чистота функцій: Функції повинні бути безсторонніми і не мати побічних ефектів. Це означає, що функції не повинні змінювати стан системи або зовнішніх змінних, а лише приймати вхідні дані і повертати результат;

– розділення даних та функцій: Дані, з якими працюють функції, повинні бути відокремлені від функцій, що з ними працюють. Це сприяє модульності та зручності підтримки коду;

– відсутність стану: Функції не повинні залежати від зовнішнього стану системи або попередніх викликів функцій. Вони повинні бути незалежними та здатними виконувати свою функцію незалежно від контексту;

– рекурсія: Функціональна модель активно використовує рекурсію - виклик функції самою собою. Це дозволяє ефективно вирішувати проблеми, що мають рекурсивну структуру, наприклад, обробка дерев або списків;

– не мутабельність: Дані, з якими працюють функції, повинні бути не мутабельними, тобто не змінюватися після створення. Замість цього, функції створюють нові дані на основі вхідних даних, зберігаючи незмінність вихідних даних.

Функціональна модель часто використовується в функціональному програмуванні і може принести такі переваги, як зменшення складності коду, легкість тестування та розуміння програми, більша стабільність та зручність у використанні. Вона підходить для розробки алгоритмічних і обчислювальних задач, а також для паралельного та розподіленого програмування.

Наша програма, як і всі інші програми, має свій алгоритм, а програма, насамперед повинна відповідати вимогам, які ми для неї визначаємо.

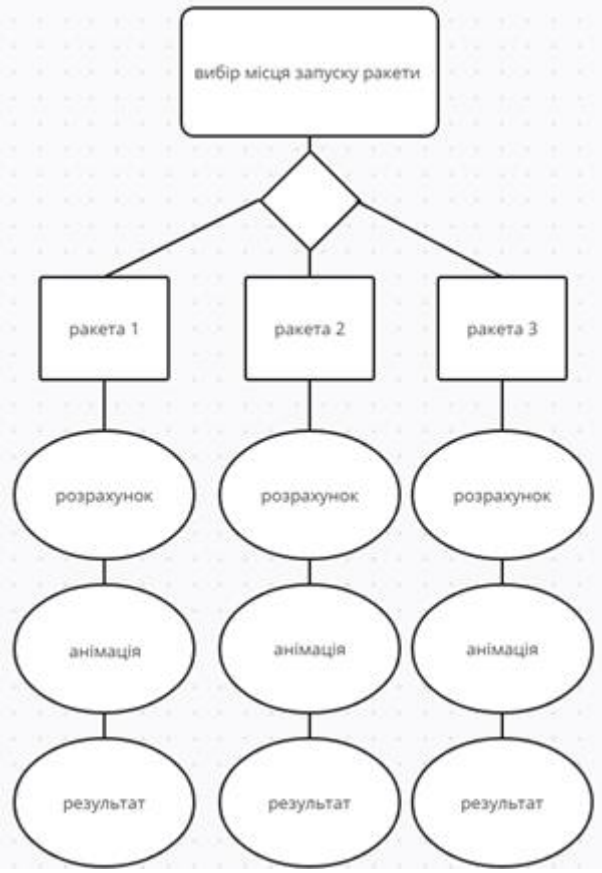

Рисунок 3.1 – Модель програми

Тут зображено просту модель роботи застосунку та скільки застосунків має основних функціональних частин, зв’язок між ними та їх внутрішні функції . Як бачите в теорії нічого складного.

Після дослідження, ми можемо зробити висновки о недоліках, які будуть використані при створенні застосунку.

На початку треба розробити алгоритм дій, щоб знати які частини програми ми повинні реалізувати. Алгоритм буде мати вигляд нашої програми з етапами які буде проходити простий користувач.

Рисунок 3.2 – Алгоритм запуску ракети з обраної точки на карті

Застосунок має бути з простим інтерфейсом для користувача, цікавим та швидким. Код програми треба оптимізувати та виправити всі помилки у розрахунках. Мають бути підказки та інші речі що роблять програму простою у використанні але з найбільшим коефіцієнтом корисної дії.

Вимоги до проєкту та його реалізації:

– відповідність нашим цілям;
– наукова складова;
– проста навігація;
– підсказки;

– точність та швидкість виконання роботи;
– відповідність задачам.

У попередньому пункті ми виявили очевидні недоліки двох програм зі схожими властивостями, наша мета зробити застосунок на прикладі існуючих програм, щоб отримати систему моніторингу ракетних ударів.

3.2 Математична модель застосунку для аналітичної системи моніторингу ракетних ударів

Математична модель - це абстрактна представлення реальної системи, яка використовує математичні концепції, формули, рівняння та методи для опису та розуміння її поведінки. Вона використовується для аналізу, прогнозування та вивчення системи, що досліджується.

Математичні моделі можуть бути застосовані до різних областей, таких як фізика, економіка, біологія, соціологія, інженерія та багато інших. Вони дозволяють абстрагуватись від складності реальної системи та досліджувати її поведінку за допомогою математичних методів.

Математична модель зазвичай складається з математичних рівнянь або систем рівнянь, які описують залежності між різними змінними системи. Ці рівняння можуть включати фізичні закони, статистичні моделі, диференціальні або інтегральні рівняння, алгоритми та інші математичні конструкції.

Математична модель дозволяє розуміти та прогнозувати поведінку системи, а також проводити різні аналізи, такі як чутливість до зміни параметрів, оптимізація, стохастичне моделювання тощо. Вона може бути використана для розробки нових стратегій, вирішення проблем або встановлення взаємозв'язків між різними факторами системи.

Важливо зауважити, що математична модель є спрощеним представленням реальної системи, і вона має свої обмеження. Вона базується на припущеннях та

уявних умовах, які можуть не завжди точно відображати реальність. Тому важливо ретельно аналізувати та перевіряти модель на відповідність реальним даним та результатам експериментів.

Комп'ютерне математичне моделювання включає кілька етапів, які можна умовно розділити наступним чином:

Визначення проблеми: Перший етап полягає в чіткому визначенні проблеми або системи, яку ви бажаєте дослідити чи моделювати. Це може бути проблема з фізики, економіки, біології, соціології тощо.

Наступний крок - це формалізація проблеми за допомогою математичних рівнянь, моделей та параметрів. Ви визначаєте, які фактори впливають на систему, як вони взаємодіють і яким чином вони можуть бути представлені математично.

Розробка математичної моделі. На цьому етапі ви розробляєте математичну модель, яка описує поведінку системи. Це може бути система диференціальних рівнянь, стохастична модель, алгоритм чи будь-який інший математичний формалізм, що відповідає вашій проблемі.

Розробка комп'ютерної програми. На цьому етапі ви переводите математичну модель у комп'ютерну програму. Ви використовуєте мову програмування (наприклад, Python, MATLAB, C++) та відповідні бібліотеки для реалізації математичної моделі та алгоритмів, які необхідні для її розв'язання.

Параметризація та налаштування: На цьому етапі ви визначаєте значення параметрів моделі та налаштовуєте їх таким чином, щоб відповідати умовам вашої проблеми. Це може включати підбір оптимальних значень параметрів або проведення експериментів для збору реальних даних.

Виконання моделювання. Після налаштування параметрів ви запускаєте комп'ютерну програму, яка виконує математичну модель. Вона обчислює поведінку системи та надає результати, які можуть бути графіками, числовими значеннями чи іншими формами.

Аналіз та інтерпретація результатів: Останній етап включає аналіз та інтерпретацію отриманих результатів моделювання. Ви оцінюєте, наскільки вони відповідають вашим очікуванням та поставленим цілям. Це може включати порівняння з експериментальними даними, проведення чутливості аналізу, ідентифікацію трендів та інше.

В основу класифікації математичних моделей можуть бути покладені безліч принципів. Можна розділити моделі по галузями наук. Можна класифікувати за математичним апаратом який був використаний (моделі, які засновані на застосуванні диференціальних рівнянь, диференціальних рівнянь в приватних похідних, дискретних алгебраїчних перетворень та інші). Нарешті, якщо виходити із загальних завдань моделювання в різних науках безвідносно до математичного апарату, найбільш нормальною є така класифікація:

– дескриптивні (описові) моделі;
– оптимізаційні моделі;
– багатокритеріальні моделі;
– ігрові моделі.

Дескриптивні (описові) моделі. Наприклад, моделювання руху космічного тіла, що увійшло у Сонячну систему, яке проводиться з метою передбачення траєкторії польоту цього тіла, його відстані, на якому воно пройде від Землі та інше. У цьому разі, цілі моделювання носять описовий характер, так як немає ніяких можливостей якось змінити рух комети та будь-що інше.

Оптимізаційні моделі, це моделі які використовуються для опису процесів. На них можна впливати при досягненні заданої мети. В цьому випадку до моделі відносяться один або кілька параметрів, на які можна впливати. Наприклад, змінюючи тепловий режим у зерносховищі, можна поставити собі за мету підібрати такий режим, щоб досягти максимального збереження зерна, тобто оптимізувати процес зберігання.

Багатокритеріальні моделі. Нерідко доводиться оптимізувати процес за кількома параметрами одночасно, причому цілі можуть бути досить суперечливими. Наприклад, знаючи ціни на продукти і потреби людини в їжі, потрібно організувати харчування великих груп людей (в армії, дитячому літньому таборі тощо). Фізіологічно правильно і, одночасно з цим, як можна дешевше. Очевидно, що ці цілі зовсім не збігаються, тобто при моделюванні буде використовуватися кілька критеріїв, між якими потрібно шукати баланс.

Усі дії програми регулюються через математичні рівняння, чи функції влаштовані в програму. До реалізації коду, на сам перед треба визначитися з тим які формули треба для використання в нашому застосунку. Так як програма демонструє нам поводження ракети, а саме швидкість та час польоту, треба використовувати математичні формули. На початку нам треба визначити рівняння знаходження часу від швидкості та дальності польоту:

$$t=V / S, \tag{3.1}$$

де t – час польоту ракети;

V – швидкість ракети;

S – радіус дії ракети.

А також визначити рівняння ідеальної ракети:

$$\Delta u = V_{eq} * \ln(\frac{mf}{me}) = V_{eq} * ln * MR = isp * p_0 * lnMR, \tag{3.2}$$

де M – маса ракети;

u – швидкість ракети;

V – швидкість вихлопу ракети;

A – витяжна зона;

p – тиск вихлопу;

p_0 – атмосферний тиск.

Для аналітичної частини роботи, було вирішено використовувати метод Монте-Карло. Цей метод є статистичним чисельним методом, який використовується для апроксимації або оцінки математичних величин за

допомогою випадкових чисел. Цей метод отримав назву в честь відомого казино в Монте-Карло, де випадковість гри в рулетку є ключовим елементом. Він широко застосовується у різних галузях, включаючи фізику, фінанси, комп'ютерну графіку, статистику та оптимізацію.

Основна ідея методу Монте-Карло полягає у використанні випадкових чисел для створення моделей або симуляцій системи чи процесу, а потім аналіз результатів, отриманих за допомогою цих моделей. Процес методу Монте-Карло включає кілька кроків.

Перший крок – формулювання завдання. Необхідно визначити математичну модель чи величину, яку потрібно оцінити чи апроксимувати з допомогою методу Монте-Карло.

Другий крок – генерація випадкових чисел. Необхідно згенерувати послідовність випадкових чисел, які використовуватимуться для моделювання системи чи процесу. Зазвичай використовуються псевдовипадкові числа, які можна одержати за допомогою різних алгоритмів генерації випадкових чисел.

Третій крок – моделювання. Використовуючи згенеровані випадкові числа, створюється модель чи симуляція системи чи процесу. У цій моделі виконуються необхідні обчислення або проводяться експерименти, щоб отримати оцінки величини, що шукається.

Четвертий крок – аналіз результатів. Отримані результати аналізуються, щоб отримати оцінку або апроксимацію шуканої величини. Часто використовуються статистичні методи аналізу результатів, такі як усереднення, варіаційне обчислення і довірчі інтервали. Чим більше випадкових чисел використовується у моделюванні, тим більш точною стає оцінка.

Метод Монте-Карло має кілька переваг. По-перше, він є універсальним методом, який може бути застосований до різних завдань. По-друге, дозволяє моделювати складні системи з урахуванням випадкових впливів. По-третє, метод

не вимагає знання аналітичної формули для оцінки величини і може бути використаний у випадках, коли аналітичне рішення неможливе або надто складне.

Проте метод Монте-Карло також має певні обмеження. Він може бути обчислювально інтенсивним і вимагати велику кількість випадкових чисел для досягнення точних оцінок. Крім того, він може бути витратним при моделюванні систем з великим числом варіантів або високої розмірності.

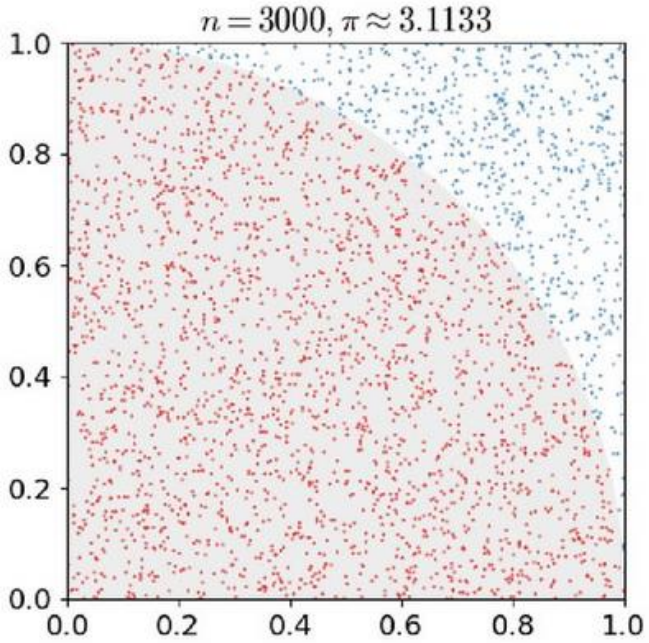

Рисунок 3.3 – Приклад роботи методу Монте-Карло

Застосування методу Монте-Карло різноманітне. Центр військово-морських аналізів розробив симуляцію ракети за допомогою методу Монте-Карло, яка обчислює ймовірність цілі збору (PACQA) і розподілу прибуття ракет разів. Симуляція враховує численні ракети та цільові невизначеності. Цілі можуть мати незалежний рух або корельований рух і мені дозволено змінювати напрямок і швидкість. Для визначення використовується модель руху Random Tour кількість разів, коли ціль змінюється. Скоординовані удари є імітується шляхом моделювання кількох ракет із кількох запусків точки на обстріл групи цілей. Кожна ракета може мати а унікальне поєднання восьми шляхів і режиму пошуку і може бути запущений у будь-який визначений час під час участі. Тактичні застосування

симуляції включають бойові дії планування та виявлення розміру залпу на основі ймовірностей захоплення. Розподіл часу польоту ракет може допомогти у плануванні скоординованих ударів і у вказівці, якщо цільовий захист може бути насиченим. Симуляція керується подіями. Параметри вибіркові кожну ітерацію Монте-Карло, а також час їх появи. Критичні події - це ті, чиї час виникнення визначає часові кроки в моделюванні. Кожен такий час визначається як час події (TOE). все TOES відсортовані в порядку зростання та відмінностей між послідовними TOES використовуються як часові кроки для Ітерації метода Монте-Карло. Вхідні параметри моделювання:

– розташування кожної точки запуску(широта та довгота) ;

– розмір залпової точки пуску Екологічні умови точки старту (вітер, температура, дощ);

– призначена ціль для кожної ракети;

– режим пошуку для кожної ракети;

– траєкторія польоту кожної ракети (початковий курс, виліт шляхові точки).

Для кожної цілі – це розташування (розмір AOU та орієнтація), напрямок, швидкість і пов'язана з ними невизначеність, час запуску У фізиці може використовуватися для моделювання руху частинок, розрахунку властивостей матеріалів чи моделювання фізичних явищ. У фінансах він застосовується для моделювання цін на фінансові інструменти та оцінки ризиків. У комп'ютерній графіці метод використовується для моделювання освітлення, створення реалістичних зображень та симуляції фізичних взаємодій. У статистиці метод Монте-Карло використовується з метою оцінки інтегралів, проведення статистичних випробувань і моделювання випадкових процесів.

Метод Монте-Карло є потужним інструментом чисельного моделювання та вирішення завдань у різних галузях. Він дозволяє отримувати наближені рішення та оцінки за допомогою випадкових чисел та симуляцій.

Для застосування цього методу у нашому застосунку, треба мати декілька значень, такі як область виконання цього методу, кількість циклів виконання та метод для класифікації отриманих даних. Вхідними даними для нас буде область ураження ракети та кількість випадкових подій, у нашому разі це 100 подій. На виході ми отримаємо процентну можливість влучень у області, які знаходяться у зоні можливого ураження ракетою.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Обґрунтування та вибір базових програмних засобів.

Приступаючи до створення проєкту необхідно визначитись з пакетом програм та технологій, які слід обрати для реалізації проєкту. Виходячи з попереднього аналізу цілей, функціоналу та інших особливостей створюваного застосунку було обрано платформу Visual Studio для нашої програми. Visual Studio – це платформа для програмування з обширною базою вибору мови, на сам перед C#.

Ця програмна середа має великий набір інструментів, зручний, багатофункціональний інтерфейс та купу речей для легкого програмування.

Програма незамінна у сфері розробки та налагодження застосунків, а так само для сумісної роботи з іншими продуктами, такі як Unity 3D.

Перевага використання даної платформи полягає в доступності всіх програмних елементів для побудови гнучкого застосунку та для подальшого його тестування, так як програма передбачає використовування компілятора, та листа з помилками.

Розглянемо більш детально користувацький інтерфейс Visual Studio. В даному разі ми використовуємо Visual Studio 2022 року. На початку роботи користувача вітає головна сторінка програми де все і так логічно зрозуміло.

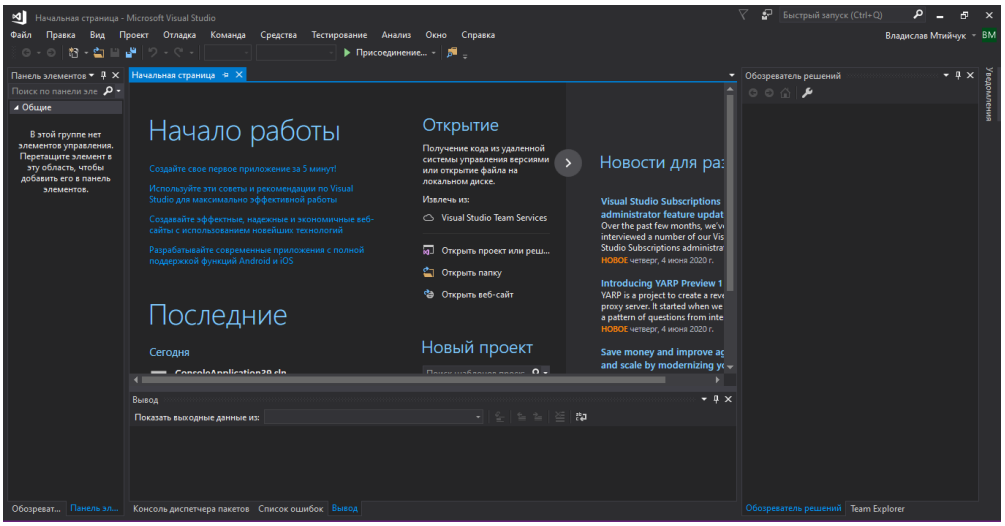

Рисунок 4.1 – Стартове вікно Visual Studio

Після увімкнення Visual Studio ми можемо одразу перейти до меню програми, де як і завжди є вкладника «Файл» та обрати створення проєкту. Після цього з’явиться вікно з вибором мови програмування та засобами для створення проєкту на обраній мові.

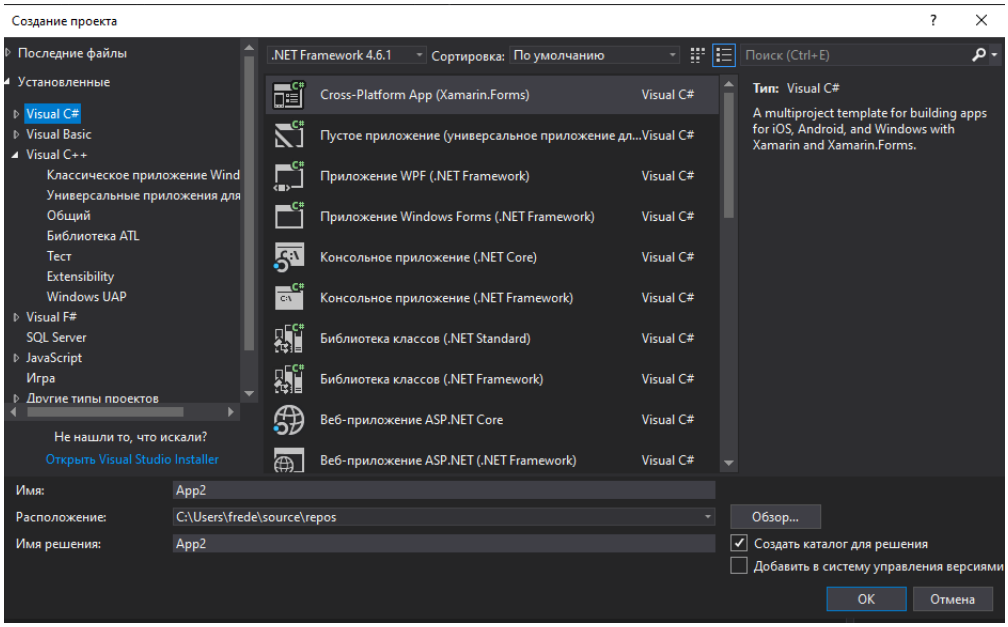

Рисунок 4.2 – компоненти Visual Studio в вкладенці C#

До переліку можливих пунктів, що встановлені у нашому пакеті:

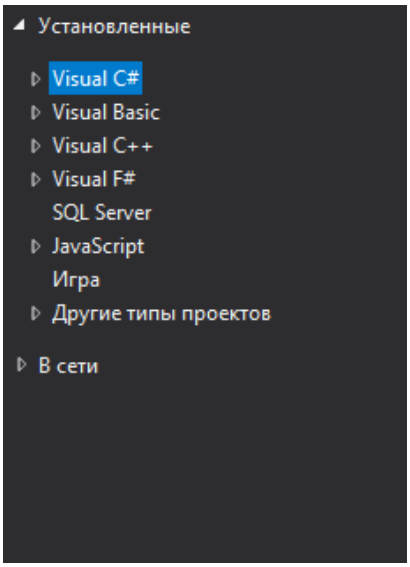

Рисунок 4.3 – Усі компоненти Visual Studio

Як можна побачити, Visual Studio дозволяє різні мови програмування та фреймворки для побудови застосунку, до яких входять C++, C#, JS та інші.

У середовищі розробки Visual Studio з використанням мови C# ви можете керувати елементами проєкту, такими як файли, класи, методи та властивості, за допомогою різних елементів інтерфейсу. Ось декілька основних елементів інтерфейсу, які допомагають керувати елементами проєкту:

– провідник рішень (Solution Explorer): Це вікно відображає структуру вашого рішення та всі його складові елементи, такі як проєкти, файли, папки тощо. Ви можете використовувати провідник рішень для створення нових файлів, перейменування, переміщення, видалення та організації елементів вашого проєкту;

– вікно коду (Code Editor): Це основне вікно, в якому ви пишете код вашого проєкту. Ви можете використовувати вікно коду для створення класів, методів, властивостей та інших конструкцій мови C#. Вікно коду надає можливості автодоповнення, перевірки синтаксису та рефакторингу коду;

– вікно властивостей (Properties Window): Це вікно відображає властивості обраного елемента у вашому проєкті. Наприклад, ви можете використовувати вікно властивостей для налаштування властивостей форми,

елементів керування, класів тощо. Ви можете змінювати розмір, положення, колір, шрифт, стиль та інші властивості елементів за допомогою вікна властивостей;

– вікно майстра (Toolbox): Це вікно містить колекцію доступних елементів керування та інших компонентів, які ви можете використовувати у вашому проєкті. Ви можете перетягувати елементи з вікна майстра до вікна розмітки форми, щоб додати їх до вашого інтерфейсу. Вікно майстра також містить інші корисні компоненти, такі як налагоджувач, контролери даних, компоненти доступу до бази даних тощо.

Якщо користувач зробить яку-небудь дію з формою, або одним з її елементів управління, створюється подія. Ваша програма реагує на ці події за допомогою коду, який ви прописуєте заранне, і обробляє їх при виникненні.

З платформою Visual Studio та Windows Forms можна гнучко налаштувати зовнішній вигляд користувацького інтерфейсу. Вікно для редагування (Рисунок. 4.4) має декілька блоків між якими розподілені налаштування та те, що побачить користувач при виконанні коду. Всі ці елементи можна міняти місцями, загалом налаштувавши своє робоче місце.

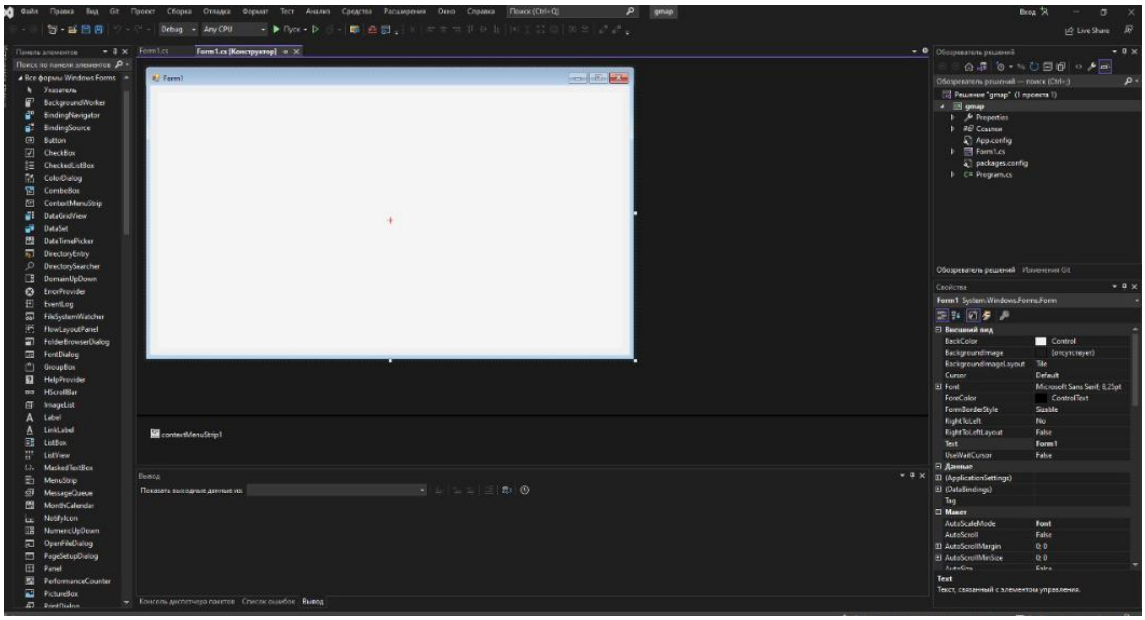

Рисунок 4.4 – Вікно налаштувань Windows Forms C#

У Visual Studio і Windows Forms є кілька способів налаштування користувацького інтерфейсу (UI). Розглянемо деякі основні методи:

– візуальний редактор: Visual Studio надає вбудований візуальний редактор, що дозволяє легко проєктувати та редагувати форми Windows Forms. З використанням редактора ви можете перетягувати та розташовувати елементи у формі, встановлювати їх властивості та настроювати вигляд та поведінку;

– властивості елементів керування: Кожен елемент у Windows Forms має набір властивостей, які можна налаштовувати. Ці властивості включають розмір, положення, колір, шрифт, стиль, зображення та багато іншого. Ви можете настроювати ці властивості за допомогою властивостей вікна властивостей Visual Studio або програмно в коді;

– розміщення та вирівнювання елементів: Visual Studio надає різні засоби для розміщення та вирівнювання елементів на формі. Ви можете використовувати контейнерні елементи, такі як Panel, FlowLayoutPanel або TableLayoutPanel, для групування та організації елементів. Крім того, ви можете встановлювати властивості вирівнювання, такі як Anchor і Dock, для контролю місцезнаходження елементів при зміні розмірів форми;

– обробка подій: Windows Forms дозволяє відповідати на події, що виникають при взаємодії з елементами керування. Ви можете програмно призначати обробники подій для елементів, такі як кнопки, текстові поля або списки, і виконувати відповідні дії при спрацюванні подій, наприклад, натисканні кнопки або зміні значення в полі вводу;

– власні графічні ресурси: Ви можете використовувати власні графічні ресурси, такі як зображення, іконки або курсори, для налаштування вигляду елементів керування. Visual Studio дозволяє додавати ці ресурси до проєкту та використовувати їх у властивостях елементів керування.

В ході розробки застосунку були використані тільки стандартні бібліотеки фреймворку .Net, але у подальшому можна підключати плагіни та бібліотеки що покращують роботу та візуальну частину нашого проєкту.

4.2 Створення моделі системи моніторингу ракетних ударів

При розробці проєкту, треба впевнено себе почувати у просторі даної програми. На сам перед Windows Forms це витриманий та гнучкий дизайн, з адаптивними вікнами та своїми вкладеними функціями, які присутні нашому фреймворку .Net.

Для адаптивності та детального налаштування існує панель «Свойства». Для кожного елементу форми вони унікальні, окрім списку критеріїв під назвою «Макет», який прийме властивості пов’язані з позиціюванням, такі як:

– Anchor - відповідає за прив’язку до комірок вікна;

– AutoSize - функція розтягування елементів;

– AutoSizeMode – розтягувати відносно чого;

– Dock – позиціювання елементів у області, що початково пропонує програма;

– Location – місце де знаходиться елемент;

– Min\MaxSize – мінімальний та максимальний розмір елементу при масштабуванні;

– Margin – позиціювання у процентах;

– Padding – відступи від інших елементів;

– Size – розмір який ми бачимо у редакторі.

Розглянемо всі вікна у проєкті.

Перше вікно це меню, воно дуже просте, тому що не має ніяких зайвих речей, а саме дві кнопки для початку та завершення дії програми.

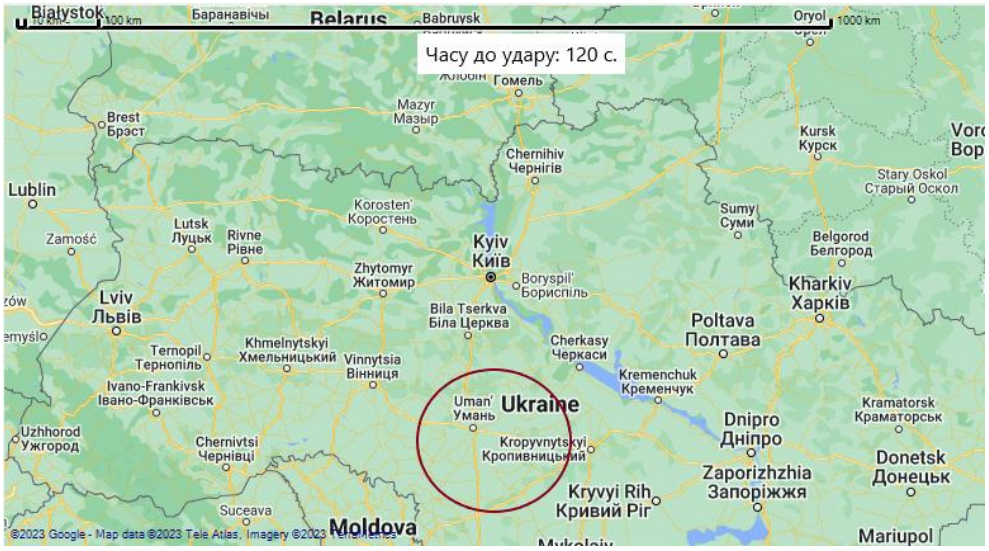

Рисунок 4.5 – Початкове меню

Як можна побачити на рисунку. 4.5, меню програми не має навігаційних елементів, адже все і так інтуїтивно зрозуміло, як і в наших аналогах програм. Але в даному випадку ми маємо відмінний від аналогів функціонал.

```
private void label1_Click_1(object sender, EventArgs e)
{

}

private void timer1_Tick(object sender, EventArgs e)
{
    if (comboBox1.Text == "Ракета Калибр")
    {
        speed = 240;//240 m/s
        range = 500000;//500 km
        seconds += 30;
        label2.Text = "Летит " +comboBox1.Text.ToString();
        comboBox1.Text = "";

    }
    if (comboBox1.Text == "Ракета Кинжал")
    {
        seconds += 50;
        label2.Text = "Летит " + comboBox1.Text.ToString();
        comboBox1.Text = "";

    }
    if (comboBox1.Text == "Ракета X-101")
    {
        seconds +=90;
        label2.Text = "Летит " + comboBox1.Text.ToString();
        comboBox1.Text = "";
```

Рисунок 4.6 – Приклад оформлення конструкторів

В .NET Framework використовується новий інтерфейс - бібліотека класів .NET Framework, яка містить більш ніж 10 000 різних типів: класів, структур, інтерфейсів. Деякі класи FCL містять до 100 методів, властивостей та інших членів. Щоб уміти розробляти програми в .Net Framework, потрібно не тільки знання програмування на деякій мові, а й уміння використовувати бібліотеку FCL. FCL добра тим, що вона повністю об'єктно-орієнтована, і може використовуватися всіма мовами програмування, які працюють на платформі .Net.

Бібліотека FCL містить набір системних типів даних, для яких в конкретних мовах програмування робиться відповідність з типами даних які вони використовують.

Для полегшення використання FCL увесь зміст добре структурований у вигляді організованих груп типів. Кожна група типів називається простором імен. У кожному з них містяться класи та інші типи, що мають деяке загальне призначення. Наприклад, велика частина Windows API для керування вікнами міститься в просторі імен System.Windows.Forms. Тут знаходяться всі можливі класи, що представляють вікна, діалоги, меню та інші елементи, які звичайно використовуються в застосунках з графічним інтерфейсом користувача. Окремий простір - System.Collections - містить класи колекцій і словників, а в просторі імен System. 10 - класи для роботи з даними на зовнішніх пристроях.

Таблиця 4.1 – Простори імен фреймворку .Net

Простір імен	**Вміст**
System	базові типи даних і допоміжні класи
System. Collections	колекції, словники, масиви змінної розмірності і інші контейнери
System.Data і ін.	класи ADO.NET для доступу до даних
System.Drawing	класи для малювання у вікні (GDI +)

Закінчення таблиці 4.1

Простір імен	**Вміст**
System. 10	класи файлового і потокового введення-виведення
System. Net	класи для роботи з мережевими протоколами, наприклад, HTTP
System.Reflection і ін.	класи для читання і запису метаданих
System.Runtime. Remoting і ін.	класи для розподілених застосунків
System. ServiceProcess	класи для створення служб Windows
System.Treading	класи для створення і управління потоками

Завдяки знанням просторів імен, ми можемо використовувати функції які вже придумані замість нас. Тобто бібліотека надає доступ до елементів які знаходяться всередині неї через ключові слова та посилання на елементи класів, що дозволяє програмно дуже просто зробити велику кількість різних речей.

Головна задача нашої роботи це розробка аналітичної системи для моніторингу ракетних ударів. Для цього використовують прості математичні або фізичні рівняння, формули та інше, разом з можливостями простору імен. Для моделювання руху у просторі програми використовують осі **x** та **y,** які в геометрії відповідають за рух вгору, вниз, вправо, вліво та використовує таймер, тому що наш об’єкт пересувається у реальному часі.

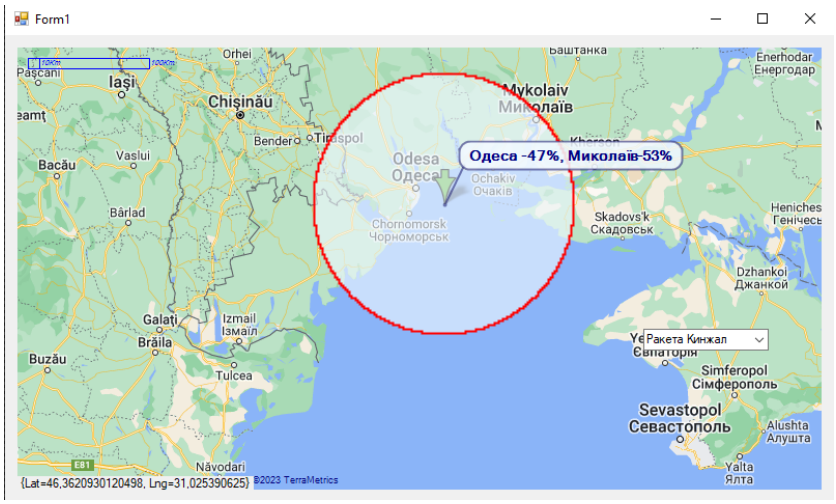

Рисунок 4.7 – Результат виконання програми

Після цих не складних маніпуляцій на виході ми маємо програму, яка за допомогою вичислень, генерую абстрактну точку на мапі, та зону ураження в межах якої буде рухатися ракета, та за допомогою обчислення методом Монте Карло маємо виноску з процентним відношенням можливості влучення ракети до області, яка входить у зону ураження.

Також було проведені розрахунки скорості виконання програми, при різній кількості запущених ракет, та побудовані графік відношення часу обробки функцій програми до кількості випущених ракет, який виглядає так:

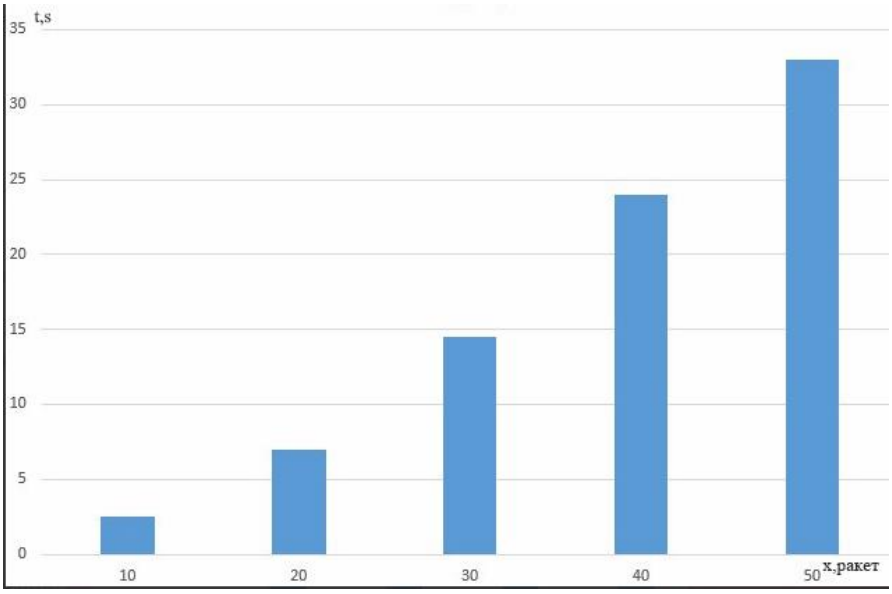

Рисунок 4.8 – Графік залежності часу обробки (Y) вірогідності влучення від кількості ракет (X)

На рисунку чітко видно, що час обробки програми напряму залежить від кількості вхідних значень та паралельно виконуваних процесів. Різниця між 10 та 50 ракетами у часі майже в 15 разів більше.

Висновки до розділу 4

В ході виконання практичної частини даної бакалаврської кваліфікаційної роботи було створено аналітичну систему моніторингу ракетних ударів.

Було визначено програмні засоби, такі як бібліотека Gmap, фреймворк .Net та методи розрахунків через функції та методи різних класів вбудованих в середовище Visual Studio за допомогою яких була розроблена візуальна та технічна частина застосунку. Для створення застосунку було використано платформу Visual Studio та Windows Forms. Завдяки використанню Windows Forms кожен обробник подій на формі повністю відповідає об’єкту інтерфейсу, тому що кожен з об’єктів має свій набір функцій, які можна знайти на вкладці «Події» в середині інтерфейсу Visual Studio, що є не тільки корисним, а і дуже простим у використанні.

Для аналітики можливості ракетних ударів був використаний метод Монте-Карло, який за допомогою випадкових подій вираховує процентну ймовірність влучення ракети до областей, які входять до області ураження ракетою. Цей метод базується на генераторі випадкових чисел у певному діапазоні. Цим діапазоном виступає межи дії ракети через координати X та Y, ці параметри ми отримуємо безпосередньо з вікна GMapControl, який передає нам реальні координати широти та довготи з онлайн мапи.

До інтерфейсу програми було додано інформаційну комірку з точною координатою місця пуску ракет та шкала масштабування карти, для вираховування зони ураження на максимально близькому масштабуванні карти.

ВИСНОВКИ

В ході виконання бакалаврської кваліфікаційної роботи було розроблено систему моніторингу ракетних ударів за допомогою мови програмування C# та фреймворку .Net.

У першій частині роботи було обрано мову програмування C# та фреймворк .Net, та розроблено чіткий план роботи. Було розглянуто схожі програми та проєкти з використанням google - maps. На основі проведених досліджень було обрано мову та інтерфейс програми, згідно з плюсами проаналізованих програм, такі як передача точного положення об’єктів та інших структур, надає дані о відстані знаходження цілі, повністю зрозумілий інтерфейс, можливість детального розгляду мапи .

У другому кроці роботи було сплановано проєктні завдання та створений календарний план з оціночними термінами виконання кожного етапу роботи. Було також проведено установку необхідного ПЗ, це Visual Studio 2022 та встановлені бібліотеки для роботи з мапою, під назвою GMap.Net та GMap.WindowsForms.Net для роботи над проєктом бакалаврської кваліфікаційної роботи.

Для програмування з урахуванням поставлених вимог, спочатку необхідно було розробити процесні та функціональні моделі застосунку. Функціональною моделлю застосунку виступають функції та методи описані у проєкті, такі як метод дозволяючий малювати область ураження ракетою, що масштабується згідно з приближенням на мапі. За основу математичної моделі було узято ідеальне рівняння ракети, через яке розраховується дальність польоту та інші властивості польоту ракети, та метод Монте-Карло, який використовується для статистичного аналізу влучення ракети до того чи іншого регіону країни з візуалізацією інформації за допомогою картографічних сервісів Google.

Виходячи з цих пунктів було спроєктовано застосунок який включає у себе інтерактивну google-maps, з можливістю вибору ракети та точки її запуску, після

чого запускається таймер з підрахунком часу польоту ракети та відображається область ураження ракети.

Для подальшого вдосконалення проєкту можна використати нові бібліотеки та методи статистичного обчислення для більш точного розрахунку поведінки ракети та місць її ураження, також розширити функціонал, наприклад додавши приблизне місце збиття ракети та падіння її уламків, та прив’язати застосунок до систем оповіщення населення.

В результаті виконання даної бакалаврської кваліфікаційної роботи були вирішені наступні завдання: обрана мова та допоміжні засоби програмування, побудовані схеми роботи програми, розроблен інтерфейс програми, розроблені методи для моніторингу ракетних ударів та методи для аналізу вірогідності потрапляння ракети, проаналізована швидкість виконання роботи з великою кількістю ракет одночасно.

Розроблений застосунок дозволяє визначити час та можливе місце ураження ракетою на реальній мапі за допомогою метода Монте-Карло, та допомагає в вивченні параметрів різних ракет, що в перспективі надає нам ще одну можливість забезпечити безпеку людям та надати можливість більш чіткого моніторингу ракет.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "C# 9 and .NET 5 – Modern Cross-Platform Development" автора Mark J. Price. URL: https://www.amazon.com/C-9-NET-5-Cross-Platform/dp/180056810X (дата звернення: 01.04.2023).

2. "Pro C# 9 with .NET 5" автора Andrew Troelsen та Philip Japikse. URL: https://www.apress.com/gp/book/9781484264811 (дата звернення: 05.04.2023).

3. "CLR via C#" автора Jeffrey Richter. URL: https://www.amazon.com/CLR-via-Developer-Reference-Richter/dp/0735667454 (дата звернення: 11.04.2023).

4. "C# in Depth" автора Jon Skeet. URL: https://www.manning.com/books/c-sharp-in-depth-fourth-edition (дата звернення: 12.04.2023).

5. "Effective C#: 50 Specific Ways to Improve Your C#" автора Bill Wagner. URL: https://www.amazon.com/Effective-Specific-Improve-Professional-Computing/dp/0321658701 (дата звернення: 20.04.2023).

6. Microsoft Docs - офіційна документація від Microsoft для C# та .NET Framework. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 25.04.2023).

7. Офіційна документація GMap.NET. URL: https://github.com/judero01col/GMap.NET/wiki (дата звернення: 28.04.2023).

8. GMap.NET на GitHub: URL: https://github.com/judero01col/GMap.NET (дата звернення: 28.04.2023).

9. GMap.NET на Stack Overflow: URL: https://stackoverflow.com/questions/tagged/gmap.net (дата звернення: 28.04.2023).

10. GMap.NET на CodeProject: URL: https://www.codeproject.com/search.aspx?q=GMap.net (дата звернення: 28.04.2023).

11. GMap.NET на NuGet URL: https://www.nuget.org/packages/GMap.NET(дата звернення: 28.04.2023).

12. METANIT.COM сайт о програмуванні. URL: https://metanit.com/sharp/windowsforms/1.1.php (дата звернення: 28.04.2023).

13. Create a Windows Forms app in Visual Studio with C#. URL: https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022 (дата звернення: 11.05.2023).

14. Уроки C# Windows Forms. URL: https://programer.in.ua/index.php/pochatkivtsiu/rozrobka-ihor-dlia-pochatkivtsiv-na-c/200-urok1-c-windows-forms (дата звернення: 12.05.2023).

15. Метод Монте-Карло. URL: https://stud.com.ua/79276/ekonomika/metod_monte_karlo (дата звернення: 21.05.2023).

16. LINQ (Language Integrated Query). URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ (дата звернення: 03.03.2023).

17. Entity Framework: ORM (Object-Relational Mapping) фреймворк для C#. URL: https://docs.microsoft.com/en-us/ef/ (дата звернення: 02.03.2023).

18. ML.NET: Бібліотека машинного навчання для C#. URL: https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet (дата звернення: 09.06.2023).

19. Accord.NET: Бібліотека машинного навчання та чисельних обчислень. URL: http://accord-framework.net/ (дата звернення: 04.06.2023).

20. Microsoft Azure Machine Learning. URL: https://azure.microsoft.com/en-us/services/machine-learning/ (дата звернення: 07.06.2023).

21. Methodology of an event-driven Monte Carlo missile simulation. URL: https://www.sciencedirect.com/science/article/pii/089571779090352N(дата звернення: 07.06.2023).

22. Angle of attack of the missile (Monte Carlo simulation). URL: https://www.researchgate.net/figure/Angle-of-attack-of-the-missile-Monte-Carlo-simulation_fig12_341847280(дата звернення: 07.06.2023).

23. An Introduction to Monte Carlo Methods. URL: https://www.coursera.org/lecture/compmethods1/an-introduction-to-monte-carlo-methods-KvlPi(дата звернення: 07.06.2023).

24. Monte Carlo Methods: Basics and Applications. URL: http://www-personal.umich.edu/~mejn/courses/2006/cmplxsys899/powerpt/intro-mc.ppt(дата звернення: 07.06.2023).

25. Monte Carlo Simulation in C#. URL: https://www.codeproject.com/Articles/861879/Monte-Carlo-Simulation-in-Csharp (дата звернення: 07.06.2023).

26. Monte Carlo Methods with C#. URL: https://www.codeproject.com/Articles/12374/Monte-Carlo-Methods-with-C(дата звернення: 07.06.2023).

27. Monte Carlo Simulation in WPF using C#. URL: https://www.c-sharpcorner.com/article/monte-carlo-simulation-in-wpf-using-c-sharp/(дата звернення: 07.06.2023).

ДОДАТОК А

Код програми з для розрахунку часу та вірогідності влучення ракети

```
using GMap.NET.MapProviders;
using GMap.NET;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using System.Runtime.Remoting.Messaging;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WindowsFormsApp3
{
   public partial class Form1 : Form
   {
      private double Scale;
      public Form1()
      {
         InitializeComponent();
         List<string> roket = new List<string> { };
         roket.Add("Ракета Калибр");
         roket.Add("Ракета Кинжал");
```

```
            roket.Add("Ракета X-101");
            for (int i = 0; i < roket.Count; i++)
            {
                comboBox1.Items.Add(roket[i]);
            }
        }
        private void gMapControl1_Load(object sender, EventArgs e)
        {
            GMap.NET.GMaps.Instance.Mode = GMap.NET.AccessMode.ServerAndCache;
            gMapControl1.MapProvider = GMapProviders.GoogleMap;

            gMapControl1.MinZoom = 2;
            gMapControl1.MaxZoom = 16;
            gMapControl1.Zoom = 6;
            gMapControl1.Position = new GMap.NET.PointLatLng(50.460797, 30.601694);
            gMapControl1.MouseWheelZoomType =
GMap.NET.MouseWheelZoomType.MousePositionAndCenter;
            gMapControl1.CanDragMap = true;
            gMapControl1.DragButton = MouseButtons.Left;
            gMapControl1.ShowCenter = false;
            gMapControl1.ShowTileGridLines = false;
            gMapControl1.MapScaleInfoEnabled = true;
        }
        private List<string> GetAddress(PointLatLng here)
        {
            List<Placemark> placemarks = null;
            var statusCode=GMapProviders.GoogleMap.GetPlacemarks(here, out placemarks);
            if (statusCode == GeoCoderStatusCode.G_GEO_SUCCESS && placemarks==null)
            {
                List<string> addresses= new List<string>();
                foreach(var placemark in placemarks)
                {
                    addresses.Add(placemark.Address);
```

```
            }
            return addresses;
        }
        return null;
    }
        private void Monte_Carlo()
        {
                for (int i= 0; i<100;i++)
                {
                        string names_town[];
                        string names_town2[];
                        Random rdpoint= new Random();
                        PointLatLng here = gMapControl1.FromLocalToLatLng(e.X, e.Y);
                        if ( comboBox1.Text == "Ракета Калібр"){
                                gMapControl1.Position = here;
                                int x1= rdpoint.Next(-600,600);
                                int y1= rdpiont.Next(-600,600);
                                PointLatLng                    mntcrl                    =
gMapControl1.FromLocalToLatLng(e.X+x1, e.Y+y1);
                                var addresses=GetAddress(here);
                                string names_town+=addresses.ToString();
                        outtool.Text ="adres:"+addresses.ToArray();
                        }
                        if ( comboBox1.Text == "Ракета Кинджал"){
                                gMapControl1.Position = here;
                                int x1= rdpoint.Next(-100,100);
                                int y1= rdpiont.Next(-100,100);
                                PointLatLng                    mntcrl                    =
gMapControl1.FromLocalToLatLng(e.X+x1, e.Y+y1);
                                var addresses=GetAddress(here);
                                string names_town+=addresses.ToString();
                        outtool.Text ="adres:"+addresses.ToArray();
                        {
```

```
                }
                for (int i = 0; i < names_town.Length; i++)
                        {
                         if (names_town[i].StrtWith("Town:"))
                                {

                                names_town2[]+=names_town[i];
                                }
                        }

        }
    private List<Placemark> temp()
    {
        List<Placemark> placemarks= new List<Placemark>();
        return placemarks;
    }
    private void AddMarker(PointLatLng pointToAdd, GMarkerGoogleType markerType=
GMarkerGoogleType.arrow)
    {
        var markers = new GMapOverlay("marckers");
        var marker = new GMarkerGoogle(pointToAdd, markerType);
        markers.Markers.Add(marker);
        gMapControl1.Overlays.Add(markers);
    }
    private void gMapControl1_MouseClick_1(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Right)

        {
            if (comboBox1.Text == "Ракета Калібр")
            {
                PointLatLng here = gMapControl1.FromLocalToLatLng(e.X, e.Y);
                label1.Text = here.ToString();
```

```
    gMapControl1.Position = here;
    AddMarker(here);
    int rad = 600;
    int seg = 50000;
          Monte_Carlo();
    Scale = (GetScale() / 100);
    GPoint cirle = gMapControl1.FromLatLngToLocal(here);
    DrawCircle(cirle, rad / Scale, seg);
    gMapControl1.Position = here;
}
else if (comboBox1.Text == "Ракета Кинжал")
{

        PointLatLng here = gMapControl1.FromLocalToLatLng(e.X, e.Y);
        label1.Text = here.ToString();
        gMapControl1.Position = here;
        AddMarker(here);
        int rad = 100;
        int seg = 50000;
                Monte_Carlo();
        Scale = (GetScale() / 100);
        GPoint cirle = gMapControl1.FromLatLngToLocal(here);
        DrawCircle(cirle, rad / Scale, seg);
        gMapControl1.Position = here;
        Bitmap imgshoot = (Bitmap)Image.FromFile("D:\\boom.png");
        GMapMarker kinjalmark = new GMarkerGoogle(here, imgshoot);
        GMapOverlay marker = new GMapOverlay("Shoot");
        marker.Markers.Add(kinjalmark);
        gMapControl1.Overlays.Add(marker);

}
else
{
```

```
            Close();
        }
    }
}
private double GetScale()
{
    GMapRoute route = new GMapRoute("getscale");
    PointLatLng point1 = gMapControl1.FromLocalToLatLng(0, 0);
    route.Points.Add(point1);

    point1 = gMapControl1.FromLocalToLatLng(100, 0);
    route.Points.Add(point1);
    return route.Distance;
}
private void DrawCircle(GPoint here,double radius, int segments)
{
    GMapOverlay overlay = new GMapOverlay("markers");
    List<PointLatLng> gpointList= new List<PointLatLng>();

    double seg = Math.PI * 2 / segments;
    for(int i = 0; i < segments; i++)
    {
        double theta = seg * i;
        double a = here.X +Math.Cos(theta)*radius;
        double b= here.Y +Math.Sin(theta)*radius;

        PointLatLng gPoint = gMapControl1.FromLocalToLatLng((int) a, (int) b);
        gpointList.Add(gPoint);
    }
    GMapPolygon poligon = new GMapPolygon(gpointList, "")
    {
        Stroke = new Pen(Color.Red, 2)
    };
```

```
            var poligons = new GMapOverlay("poligons");
            poligons.Polygons.Add( poligon );
            gMapControl1.Overlays.Add(poligons);
        }

        private void fdfdfdfToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }
    }
}
```