Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням комп’ютерних та видавничих технологій

Чубей О.О. /_____/

(підпис)

«___»_____2022 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

освітньо-кваліфікаційного рівня «молодший спеціаліст»

зі спеціальності 122 «Комп`ютерні науки»

на тему: «Інформаційна система для виявлення шкідливого програмного коду»

Студент групи КН-41 Шовдра Б.М. _____

(підпис)

Керівник проєкту Івасьєв С.В. _____

(підпис)

Консультанти:

з техніко-економічного обґрунтування Меленчук Л.І. _____

(підпис)

нормоконтролер Кульчинська Н.З. _____

(підпис)

Тернопіль – 2022

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп`ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням
комп’ютерних та видавничихтехнологій

Чубей О.О. /_____/
(підпис)
«__» _____ 2021 р.

ЗАВДАННЯ

на дипломне проєктування

на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»
студенту Шовдрі Богдану Миколайовичу
(прізвище, ім’я та по-батькові студента)

1. Тема проєкту «Інформаційна система для виявлення шкідливого програмного коду»
затверджена наказом по коледжу від “ ____” _____2021 р., № 178 - н
2. Термін здачі студентом завершеного проєкту “_____” _____2022 р.
3. Вихідні дані до проєкту _____
_____
_____
_____
4. Перелік питань, які повинні бути розроблені в проєкті: _____
а) основна частина _____
_____
_____
_____
б) техніко-економічного обґрунтування _____
_____
_____
_____
_____
5. Перелік графічного матеріалу_____
_____
_____

6. Консультанти проєкту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ (вчена ступень, звання П.І.Б. _____ консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проєктування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування.	20.11.21 р.	06.12.21 р.
2.	Огляд типових рішень та написання відповідного розділу ПЗ	06.12.21 р.	26.01.22 р.
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	26.01.22 р.	14.02.22 р.
4.	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написаннявідповідного розділу ПЗ	14.02.22 р.	02.03.22 р.
5.	Встановлення на налаштування середовища реалізації та написання відповідного розділу ПЗ	02.03.22 р.	16.03.22 р.
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.22 р.	17.04.22 р.
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	17.04.22 р.	03.05.22 р.
8.	Доопрацювання модулів	05.05.22 р.	18.05.22 р.
9.	Тестування на налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.22 р.	01.06.22 р.
10.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	20.05.22 р.	05.06.22 р.
11.	Робота над оформленням пояснювальної записки	05.06.22 р.	12.06.22 р.
12.	Попередній захист дипломного проєкту, доопрацювання	12.06.22 р.	
13.	Підготовка до захисту дипломного проєкту	15.06.22 р.	22.06.22 р.
14.	Захист дипломного проєкту	25.06.22 р.	26.06.22 р.

7. Дата видачі завдання ”____” _____ 2021 р.

Керівник _____ / _____

Завдання прийняв до виконання_____ / _____

Реферат

Інформаційна система для виявлення шкідливого програмного коду Дипломний проєкт Шовдра Богдан Миколайович. Галицький фаховий коледж імені В`ячеслава чорновола, відділення комп`ютерних та видавничих технологій. Спеціальність 122 «Комп'ютерні науки». 63 сторінок, 50 рисунки, 2 таблиці, 8 джерел, 3 додатки, 3 блок-схема.

Тема дипломної роботи «Інформаційна система для виявлення шкідливого програмного коду». Актуальність дипломного проєкту полягає в застосуванні сучасних методів виявлення шкідливого програмного забезпечення, пошуку по хеш сумах файлів та знешкодження шкідливого програмного коду, що забезпечить ефективний механізм захисту ПК. Об'єкт дослідження – процеси проєктування та розробки програмного додатку для пошуку і знешкодження шкідливого програмного коду, заснованих на методах пошуку за хеш сумою.

Об`єкт дослідження – є інформаційна система для виявлення шкідливого програмного коду.

Мета роботи – розробка програмного додатку для виявлення шкідливого програмного коду, що забезпечує, використовуючи методи пошуку за хеш сумою, ефективний пошук та знешкодження вірусів. В роботі було проведено дослідження класів існуючих вірусів та огляд сучасних антивірусних програм, здійснено дослідження і вибір методів виявлення шкідливих програм, виконано моделювання користувацького інтерфейсу, розроблено додаток для виявлення шкідливого програмного коду. Практична цінність роботи полягає в розробці додатка, який забезпечує виявлення та знешкодження шкідливого програмного коду, що може стати першою ланкою в створенні антивірусного програмного забезпечення в подальшому.

Ключові слова : SHA256, Keccak, Absorbing, Md5.

Abstract

Information system for detecting malicious code Diploma project Shovdra Bohdan Mykolayovych. Vyacheslav Chornovil Galician professional College, Department of Computer and Publishing Technologies. Specialty 122 "Computer Science". 64 pages, 50 figures, 2 tables, 8 sources, 3 appendices, 3 block diagrams.

Thesis topic "Information system for detecting malicious code". The relevance of the diploma project is the use of modern methods of detecting malicious software, searching for hash sums of files and neutralizing malicious code, which will provide an effective mechanism for PC protection. Object of research - the processes of designing and developing a software application for finding and neutralizing malicious code based on hash sum search methods.

Object of research - there is an information system for detection of malicious program code.

The aim of the work is to develop a software application for detecting malicious code, which provides, using hash search methods, effective search and elimination of viruses. The study of existing viruses and review of modern anti-virus programs, research and selection of malware detection methods, user interface modeling, developed an application for detecting malicious code. The practical value of the work lies in the development of an application that detects and neutralizes malicious software, which may be the first step in creating anti-virus software in the future.

Keywords: SHA256, Keccak, Absorbing, Md5.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – програмний інтерфейс.

ОС – операційна система.

ПЗ – програмний засіб.

ПК – персональний комп`ютер.

UI – Інтерфейс користувача.

MBR – Головний завантажувальний запис.

IDE – Інтегроване середовизе розробки.

GUI – Графічний інтерфейс користувача.

ВСТУП

Захист від комп'ютерних вірусів є одним із найпопулярніших напрямків в розробці програмного забезпечення. Навіть попри те, що боротьба із шкідливим програмним забезпеченням існує вже більше десятиліття, проблема лишається актуальною і дотепер.

Одною із особливостей протидії зараженим програмам є те, що навіть їх визначення трактуються по різному в джерелах, таких як нормативно-правові акти України та інших держав, визначення, зроблені технологічними компаніями та державні стандарти.

Серед senior-програмістів не існує одностайної думки про те, від чого захищає антивірус. Єдність відсутня навіть в значенні слова «шкідливий», так як в деяких документах застосовується значення «зловмисний код».

Одним з найдієвіших способів боротьби із шкідливими програмами є постійне дослідження поведінки системи і контроль за сигнатурами програмних кодів на прояв аномальної активності на надання про це інформації.

Таким чином, відповідно до вище сказаного, робимо висновок, про необхідність безперервного оновлення антивірусного програмного забезпечення та введення додаткових програмних модулів протидії шкідливим програмам

Об'єктом дипломного дослідження є процес протидії шкідливим програмам.

Предметом дипломного дослідження є програмний засіб для виявлення шкідливого програмного коду.

Мета дипломного дослідження полягає в розгляді основних методів моніторингу та виявлення аномалій мережевого трафіку, які спричинено шкідливими програмами.

У дипломному проєкті розглядаються існуючі види загроз безпеки а також побудова комплексної системи захисту від вірусних загроз.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Дослідження об’єкта інформатизації

Комп’ютерний вірус — це тип комп’ютерної програми, яка при виконанні реплікується, змінюючи інші комп’ютерні програми та вставляючи власний код. Якщо ця реплікація вдається, уражені ділянки називаються «зараженими» комп’ютерним вірусом, метафорою, що походить від біологічних вірусів. Комп’ютерні віруси зазвичай потребують хост-програми. Вірус записує власний код у хост-програму. Коли програма запускається, спочатку виконується написана вірусна програма, що спричиняє зараження та пошкодження. Комп’ютерному хробаку не потрібна хост-програма, оскільки це незалежна програма або фрагмент коду. Тому він не обмежений головною програмою, але може працювати самостійно та активно здійснювати атаки.

Автори вірусів використовують обмани соціальної інженерії та використовують детальні знання про вразливості безпеки, щоб спочатку заразити систему та залишити в ній вірус. Більшість вірусів націлені на системи під керуванням Microsoft Windows, використовують різноманітні механізми для зараження нових хостів і часто використовують складні стратегії антивиявлення/невидимки, щоб уникнути антивірусного програмного забезпечення. Мотиви створення вірусів можуть включати пошук прибутку (наприклад, за допомогою програм-вимагачів), бажання надіслати політичне повідомлення, особисту розвагу, продемонструвати, що в програмному забезпеченні існує вразливість, для саботажу та відмови в обслуговуванні або просто тому, що вони хочуть досліджувати питання кібербезпеки, штучний інтелект та еволюційні алгоритми.

Комп’ютерні віруси щорічно завдають економіці мільярди доларів шкоди.

Сьогодні відомі більше десяти тисяч різних комп'ютерних вірусів. Проте кількість вірусів, які відрізняються за типом поширення та принципом

дії є досить обмеженою. Крім того, існують комбіновані віруси, які одночасно відносяться до кількох видів. Основною і найбільш поширеною класифікацією комп'ютерних вірусів є класифікація за середовищем проживання, або за типами об'єктів комп'ютерної системи, в які впроваджуються віруси.

1.1.1 Завантажувальні віруси

Вірус завантаження (також відомий як інфікатор завантаження, вірус MBR або вірус DBR) націлює та заражає певний фізичний розділ комп’ютерної системи, який містить інформацію, важливу для належної роботи операційної системи (ОС) комп’ютера.

Хоча завантажувальні віруси були поширеними на початку 90-х років, вони стали набагато рідкішими після того, як більшість виробників материнських плат комп’ютерів додали захист від таких загроз, заборонивши доступ до основного завантажувального запису (найпоширенішого компонента) без дозволу користувача.

Проте останніми роками з’явилося більш складне зловмисне програмне забезпечення, яке знайшло способи обійти цей захист і перенацілити MBR (наприклад, Rootkit:W32/Whistler.A). Вірус у завантажувальному записі представлено на рисунку 1.1.

Рисунок 1.1 – Вірус у завантажувальному записі

1.1.2 Файлові віруси

Вірус, що заражає файли, – це тип зловмисного програмного забезпечення, яке заражає виконувані файли з наміром завдати постійних пошкоджень або зробити їх непридатними для використання. Вірус, що заражає файли, перезаписує код або вставляє заражений код у виконуваний файл. Цей тип вірусу може вразити ряд операційних систем, включаючи Macintosh, Windows і Unix. Вірус, що заражає файли, також може бути відомий як файл-інжектор [1].

Файлолвий вірус є одним із найпоширеніших типів вірусів. Як правило, він заражає файли з розширеннями .exe або .com. Під час доступу до зараженого файлу або його виконання він може бути частково або повністю перезаписаний вірусом. Вірус, що заражає файли, також може поширюватися по системі та по мережі, заражаючи інші системи. Крім того, важка форма вірусу, що заражає файли, також може повністю переформатувати жорсткий диск. Win32.Sality.BK – популярний вірус, що заражає файли, який у 2011 та 2012 роках входив до 10 найбільш поширених шкідливих програм [2].

На рисунку 1.2 показана дія вірусу у файлі mouse.com.

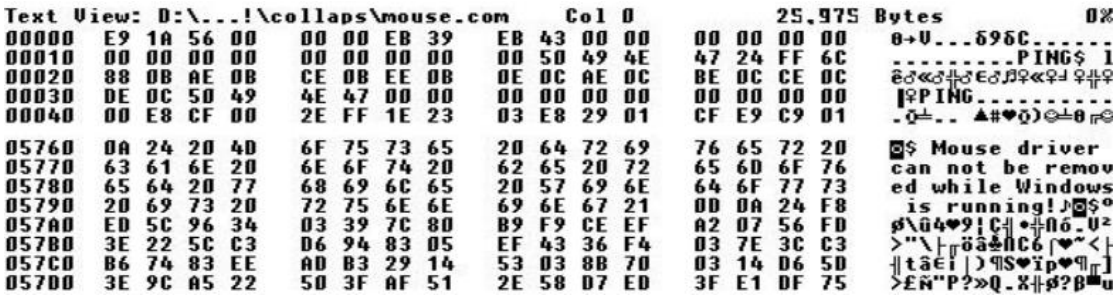

Рисунок 1.2 – Вірус у файлі mouse.com

1.1.3 Поліморфні віруси

Поліморфні віруси — це складні інфікатори файлів, які можуть створювати модифіковані версії самих себе, щоб уникнути виявлення, але зберігати ті самі основні процедури після кожного зараження. Щоб змінювати свій фізичний склад файлів під час кожного зараження,

поліморфні віруси шифрують свої коди і кожен раз використовують різні ключі шифрування. На рисунку 1.3 показаний список найпоширеніших поліморфних вірусів.

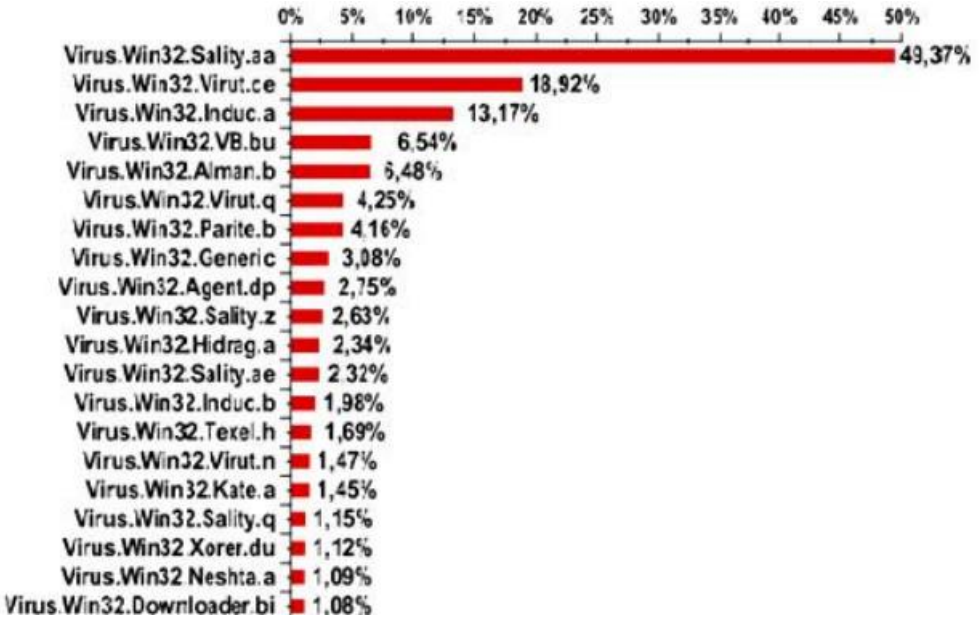

Рисунок 1.3 – Найпоширеніші поліморфні віруси

Поліморфні віруси стійкі до сигнатурного сканування та використовують ріного виду пакувальники і архіїватори.

1.1.7 Троянські коні

У обчислювальній роботі троянським конем є будь-яке зловмисне програмне забезпечення, яке вводить користувачів в оману щодо його справжніх намірів.

Троянські програми зазвичай поширюються за допомогою певної форми соціальної інженерії; наприклад, коли користувача обманюють, щоб виконати вкладений файл електронної пошти, замаскований таким чином, щоб він не виглядав підозрілим. Багато сучасних форм діють як бекдор, зв’язуючись з контролером, який може отримати несанкціонований доступ до ураженого комп’ютера [3]. Атаки програм-вимагачів часто здійснюються за допомогою трояна.

1.2 Дослідження технологій реалізації та аналіз існуючих рішень

У сьогоднішіх реаліях найважливішим завданням будь-якого користувача є пошук та встановлення антивірусної програми, адже тільки вона може захистити ПК від шкідливого коду[4]. Ця частина проєкту присвячена дослідженню антивірусів, які дадуть змогу уникнути зараженню шкідливими програмами, пошкодженню важливих даних та забезпечити стабільну роботу комп’ютера. По використовуваних технологіях антивірусного захисту вони поділяються на:

– класичні антивірусні продукти (продукти, які застосовують тільки сигнатурний метод детектування);

– продукти проактивного антивірусного захисту (продукти, якізастосовують тільки проактивні технології антивірусного захисту);

– комбіновані продукти (продукти, які застосовують як класичні,сигнатурні методи захисту, так і проактивні).

За функціоналом продуктів:

– антивірусні продукти (продукти, що забезпечують тільки антивірусний захист);

– комбіновані продукти (продукти, що забезпечують не тільки захист від шкідливих програм, але і фільтрацію спаму, шифрування та резервне копіювання даних та інші функції);

– за цільовими платформами: Windows, * NIX (до даного сімействавідносяться ОС BSD, Linux, etc), MacOS, для мобільних платформ(Windows Mobile, Symbian, iOS, BlackBerry, Android, Windows Phone).

По об'єктах захисту:

– для захисту робочих станцій, файлових і термінальних серверів та серверів віртуалізації.

Далі розглянемо шість найпопулярніших антивірусів: Avast Free Antivirus, Panda Antivirus Pro, AVG Anti-Virus Free, ESET NOD32 Smart Security, Dr.Web Antivirus.

Avast Free Antivirus – добірний безкоштовний антивірус, яким задоволені мільйони користувачів по всьому світу завдяки перевіреному захисту від шкідливих програм. В останній версії Avast з'явився оновлений інтуїтивно зрозумілий інтерфейс, крім того, поліпшений швидкодією та декілька унікальних функцій, таких як AutoSandbox, Intelligent Scanner і т.д. Також в оновлення входить база вірусів, яка щодня поповнюється [5].

На рисунку 1.4 бачимо головне меню антивірусу.

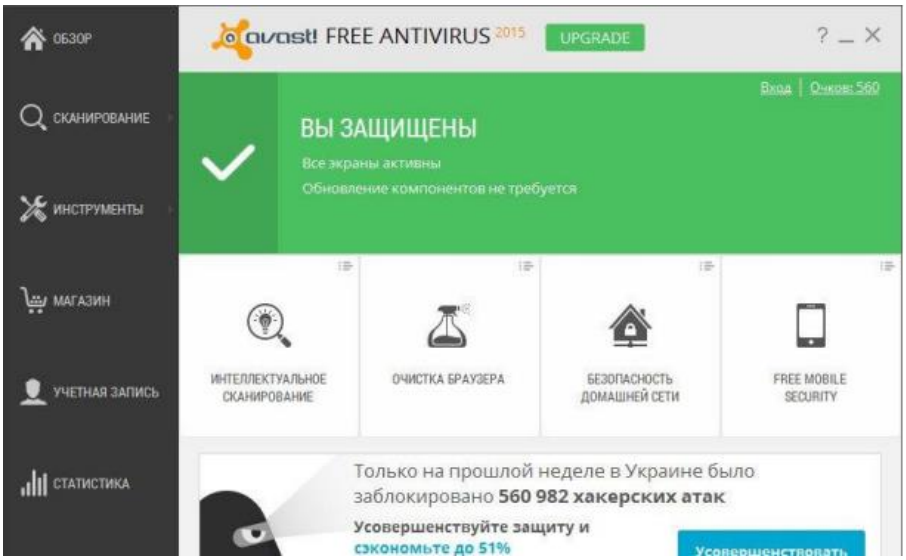

Рисунок 1.4 – Головне меню Avast Free Antivirus

Avast Free Antivirus пропонує антивірусний захист, який отримує чудові бали в практичних тестах і незалежних лабораторних тестах. Що стосується бонусних функцій, то він пропонує набагато більше, ніж багато конкуруючих комерційних продуктів, включаючи сканер мережевої безпеки, засіб оновлення програмного забезпечення тощо.

AVG Internet Security блокує віруси та зловмисне програмне забезпечення, забезпечує безпеку ваших електронних листів, захищає ваші особисті файли, паролі та веб-камеру від хакерів, а також дає змогу без турбот здійснювати покупки й банківські операції в Інтернеті [6].

Перший етап захисту AVG Internet Security для вашого ПК:

– розширений антивірус: сканує ваш комп’ютер на наявність вірусів, програм-вимагачів, шпигунських програм та інших типів шкідливих програм;

– Behavior Shield: надсилає сповіщення, якщо на вашому ПК виявлено підозрілу поведінку програмного забезпечення, виявлення штучного інтелекту: проактивно визначає зразки шкідливого програмного забезпечення, щоб захистити вас від нових загроз;

– CyberCapture: блокує нові загрози, завдяки тому, що наше антивірусне програмне забезпечення автоматично завантажує їх для аналізу. На рисунку 1.5 можна побачити головне меню антивірусу.

Рисунок 1.5 – Головне меню AVG Internet Security

Panda Security — іспанська компанія, яка має значний досвід антивірусних інновацій. Від запуску щоденних оновлень підписів у 1998 році до впровадження моніторингу поведінки в 2004 році та хмарного сканування в 2007 році, Panda була залучена до безлічі технологій, які ми зараз сприймаємо як належне.

Panda Antivirus Pro пропонує антивірусний захист із фільтрацією URL-адрес, щоб блокувати шкідливі посилання та веб-сайти. Пакет виходить за рамки основ із простим двостороннім брандмауером для антивірусного захисту, системою «контролю програм», яка точно визначає, які програми

можуть запускатися на вашому ПК, тощо. Набір функцій не зовсім відповідає конкурентам, таким як Bitdefender – немає спеціального банківського захисту, немає менеджера паролів чи подрібнювача файлів – але він все одно забезпечує трохи більше, ніж ви могли очікувати. На рисунку 1.6 можна побачити головне меню антивірусу

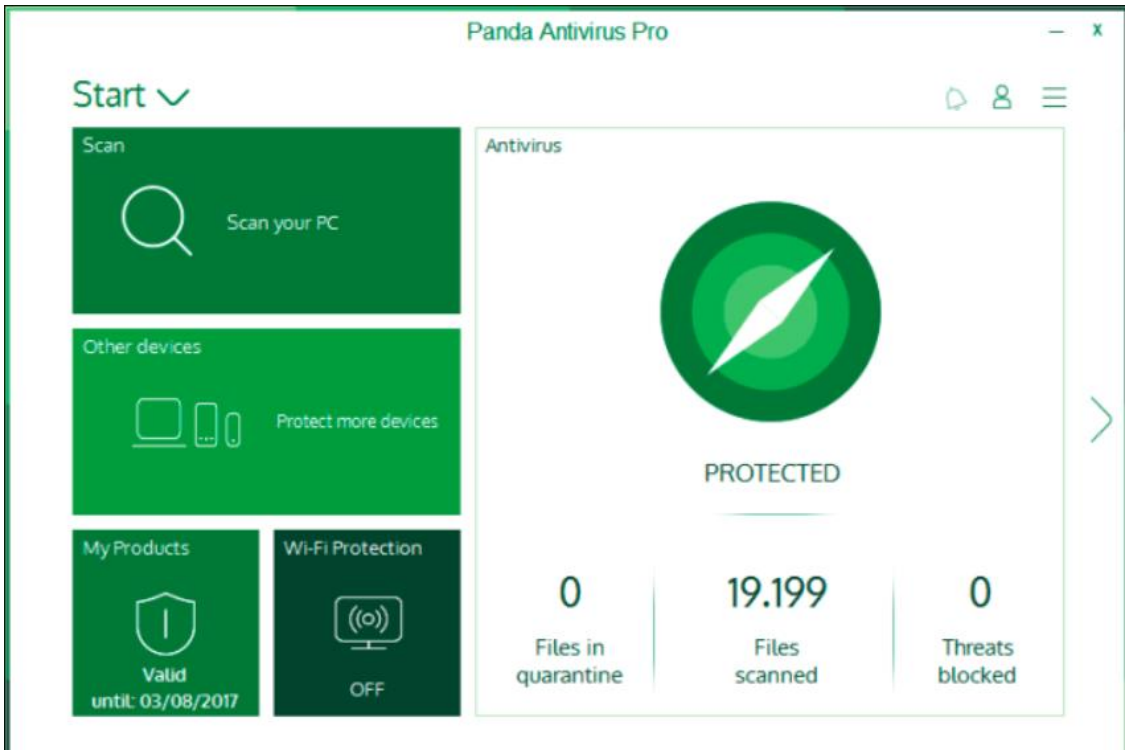

Рисунок 1.6 – Головне меню Panda Antivirus Pro

ESET NOD32 Antivirus, широко відомий як NOD32, — це пакет антивірусного програмного забезпечення словацької компанії ESET. ESET NOD32 Antivirus продається у двох випусках: Home Edition та Business Edition [7]. Пакети Business Edition додають ESET Remote Administrator, що дозволяє розгортати сервер і керувати ним, дзеркальне відображення оновлень бази даних сигнатур загроз і можливість встановлення в операційних системах Microsoft Windows Server. На рисунку 1.7 можемо побачити головне меню антивірусу.

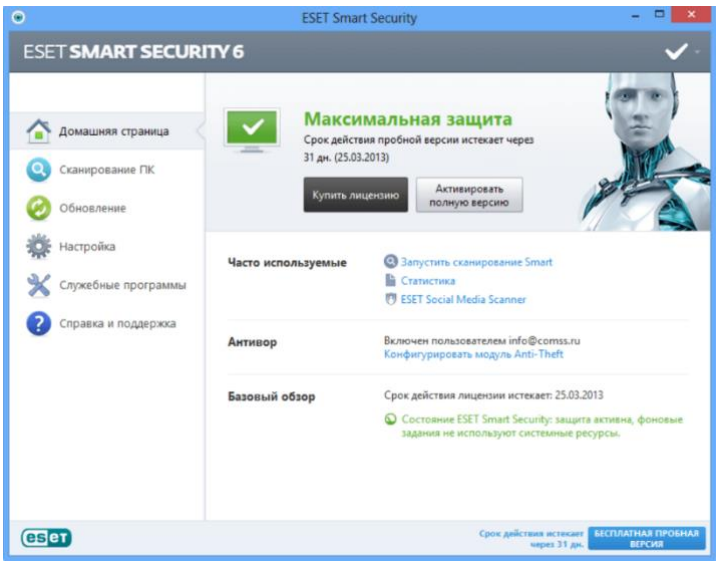

Рисунок 1.7– Головне меню ESET NOD32 Smart Security

Dr.Web — це пакет програм, розроблений російською компанією «Доктор Веб», що займається захистом від шкідливих програм. Вперше випущений у 1992 році, він став першим антивірусним сервісом в Україні.

Компанія також пропонує рішення для боротьби зі спамом і використовується для сканування вкладень електронної пошти. Він також містить доповнення для всіх основних браузерів, яке перевіряє посилання з онлайн-версією Dr Web. На рисунку 1.8 можна побачити головне меню антивірусу [8].

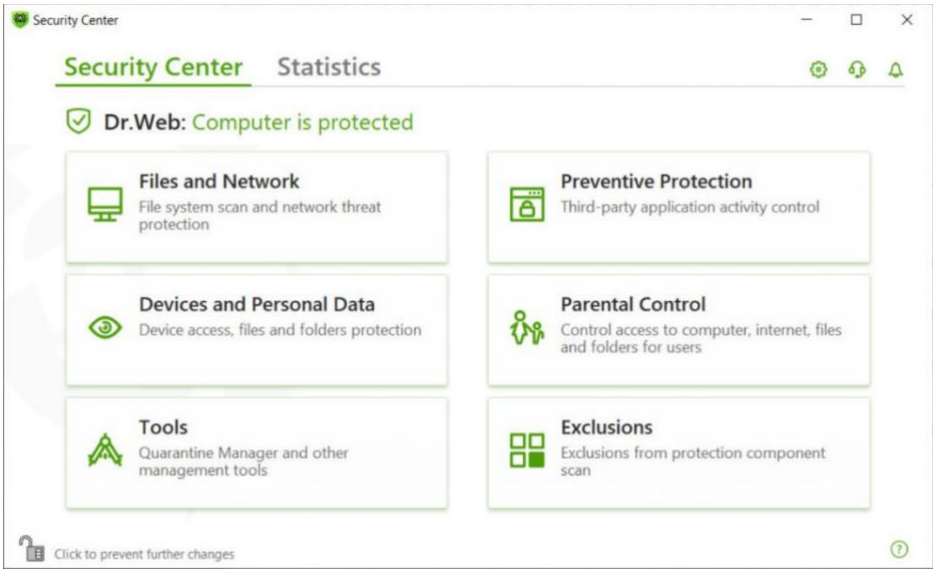

Рисунок 1.8 – Головне меню Dr.Web Antivirus

При оформленні результатів з пошуку вірусів була використана інфорація тестування антивірусів з Інтернет-ресурсу (таблиця 1.1).

Таблиця 1.1 – Порівняльна таблиця антивірусів

	Avast	Panda	AVG	ESET NOD32	Dr.Web
Час завантаження системи з антивірусом	<1 хв	<1 хв	<1 хв	<1 хв	<3 хв
Час сканування системних папок	>10 хв	>20 хв	>10 хв	>10 хв	>115 хв
Використання процесора	2,5%	3%	16, 5%	10%	20%
Використання пам’яті	40 мб	40 мб	130 мб	110 мб	120 мб
Рейтинг	8	9	7	8	5

Згідно з порівняльним аналізом можна сказати, що Panda Antivirus Pro є одним з найкращих антивірусів, що неможливо сказати про Dr.Web.

1.3. Дослідження технологій реалізації

Кількість антивірусних програм, як і кількість вірусів, постійно зростає, загальне визначення та класифікація антивірусного програмного забезпечення постійно розширюється.

Антивірусна програма - програма для виявлення та лікування шкідливих об'єктів або заражених файлів, а також для попередження зараження файлу або операційної системи шкідливим кодом. Антивірусне програмне забезпечення складається з підпрограм, які намагаються виявляти та запобігати розмноженню, видаляти комп’ютерні віруси та інші шкідливі програми. Багато сучасних антивірусних програм можуть виявляти та видаляти також трояни та інші шкідливі програми.

Через велику кількість шкідливих програм, розроблених під платформу Windows від компанії Microsoft, найпоширенішим антивірусним програмним забезпеченням є антивіруси для ОС цього сімейства.

Хоча, навіть попри славу UNIX-подібних систем, як найнадійніших, зараз збільшується попит на антивірусні програми і для інших платформ, таких як Linux і Mac OS. Збільшення кількості антивірусів під ці ОС в свою чергу викликано поширенням шкідливих програм на них.

Користувачі мобільних пристроїв, таких як Windows Mobile, Symbian, iOS, BlackBerry, Android, Windows Phone та ін. також не захищені від шкідливого програмного забезпечення. Саме для них створюються окремі антивірусні програми.

Розрізняють такі типи антивірусних програм:

– Програми - детектори: ведуть пошук властивої конкретного вірусу сигнатури в оперативній пам'яті й у файлах, коли знаходять, повідомляють про це користувача за допомогою повідомлення. Недолік таких антивірусних програм полягає в тому, що вониздатні шукати тільки віруси, відомі творцям програм детекторів;

– програми-лікарі: аналогічно до програмам-вакцин не тільки знаходять файли, заражені вірусами, а й «Лікують» ці файли, тобто ліквідують тіло програми-вірусу з файлу, повертаючи файли в первісний стан. Спочатку фаги ведуть пошук вірусів в оперативній пам'яті, ліквідуючи їх, і тільки потім приступають до «лікуванню» файлів. Середфагіввиділяютьполифаги, тобто програми-доктори, призначені для пошуку і знищення великої кількості вірусів;

– програми-ревізори: є одними з найбільш надійних засобів захисту від вірусів. Ревізори (інспектора) сканують дані на диску з метою виявлення вірусів-невидимок, перевіряють, чи не увійшов чи вірус в файли, чи немає несанкціонованих змін реєстру Windows. При цьому інспектор може і не скористатися засобамиопераційної системи для звернення до дисків, отже, у вірусу немає вийде перехопити це звернення;

– програми-фільтри: являють собою невеликі резидентні програми, які призначені для виявлення підозрілих дій при роботі ПК;

– програми-вакцини: відносяться до резидентних програм, які запобігають зараженню файлів. Вакцини використовують, якщо у користувача немає в наявності програм-докторів, які «лікують» цей вірус, аналогічно з програмами-детекторами, вони здатні нейтралізувати тільки вірусні програми, які відомі творцям антивірусне програмне забезпечення. вакцина змінює програму або диск так, щоб це не відбивалося на їх роботі, а вірус буде сприймати їх як заражені. В даний час програми-вакцини мають обмежене застосування.

Найпоширенішими алгоритмами є: CRC32, MD5 і SHA-1, показано на рисунку 1.9.

```
/* Расчет таблицы корректирующих коэффициентов */
make_crctable( void ) {
 int i, j;  DOUBLE r;
 for (i = 0; i <= 255; i++) {
  r = i;
  for (j = 8; j > 0; j--) if (r & 1)  r = (r >> 1) ^ 0xEDB88320;  else  r >>= 1;
  crctable[i] = r;
 }
}

/* Расчет CRC-32 для буфера buf длиной len */
DOUBLE crc32(unsigned char *buf, DOUBLE len) {
  DOUBLE crc = 0xFFFFFFFF;
  while (len--)  crc = (crc >> 8) ^ crctable[(crc ^ *buf++) & 0xFF];
  return crc ^ 0xFFFFFFFF; // Для облегчения аппаратной реализации
}
```

Рисунок 1.9 – Приклад реалізації для швидкого обчислення CRC

Основні методи протидії вірусам, які застосовуються антивірусами:

– Сигнатурне сканування – це найпоширеніша методика виявлення специфічних вірусів, за допомогою якої код сканується на наявність сигнатур будь-якого з набору відомих вірусів.

– Метод пошуку за контрольною сумою. Контрольна сума — це невеликий блок даних, отриманий з іншого блоку цифрових даних з метою виявлення помилок, які могли виникнути під час його передачі або зберігання.Метод пошуку по HEX рядку в заданій позиції.

– Поведінковий блокатор - це тип програми, яка запобігає виконанню певних дій. Блокувальник поведінки може перешкоджати запису програми в реєстр, завантажувальний сектор або файли. Іноді технології блокування поведінки вбудовуються в програми, які також мають інші можливості

– Евристичний аналіз

1.4. Постановка задачі

Мета дипломного дослідження полягає в розгляді та аналізі основних методів моніторингу та виявлення аномалій мережевого трафіку, які спричинено шкідливими програми, розробка програмного засобу, що буде виконувати пошук та знешкодження вірусів з відомих сигнатур, що є наявними в базі даних системи. Для досягнення цієї мети були визначені задачі, які необхідно вирішити:

– провести дослідження та аналіз відомих класів вірусів;

– здійснити огляд сучасного антивірусного програмного забезпечення;

– провести дослідження та здійснити вибір методів виявлення шкідливих програм;

– розробити програмний засіб, який повинен містити базу вірусів і антивірусний сканер, який повинен здійснювати пошук тих вірусів, сигнатури яких зазначені в базі даних;

– додаток повинен забезпечити внесення до карантину заражених вірусами файлів;

– антивірус повинен використовувати сигнатурний метод SHA-256 пошуку вірусів;

– інтерфейс розробленого програмного додатку для виявлення шкідливого програмного коду повинен бути виконаний в інтуїтивно-зрозумілому для користувача стилі;

– програмний додаток повинен бути протестований на вірусах та шкідливих програмах.

Розроблений в рамках виконання дипломного проєкту програмний засіб повинен забезпечувати виявлення та знешкодження шкідливого програмного коду, що може стати в подальшому першою ланкою в створенні потужного антивірусного програмного забезпечення.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Формалізація вимог

В результаті постановки завдання проєкту, було визначено необхідність проєктування та розробки програмного засобу, який дозволить виявляти та знешкоджувати шкідливий програмний код.

Контрольна сума використовується для порівняння набору даних та для виявлення вірусів, при цьому контрольна сума файлу і хешу з бази вірусів будуть не рівні. На рисунку 2.1 можна побачити блок-схему SHA-256 алгоритму пошуку хешу файлів, використовується для того, щоб видобути з файлу його хеш,що дасть змогу в подальшому порівняти його з базою вірусів.

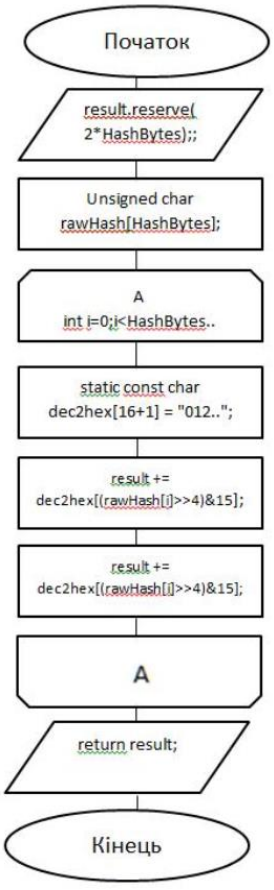

Рисунок 2.1 – Блок схема алгоритму SHA-256

Як правило, антивірус – це декілька компонентів, які відповідають за виявлення вірусів на певній ділянці своєї роботи.

Протягом розвитку склад ці компоненти змінювались, відповідно до вимог часу, додавалися нові і віддалялися ті, що не відповідають їм.

Основним методом, за допомогою якого будуть працювати компоненти моєї антивірусної є метод пошуку за контрольною сумою. .

Також підключення біблотеки дає змогу шукати хеші файлів за таким алгоритмом, як keccak, це нова хеш-функція з довжиною виходу в 224, 256, 384 і 512 біт, в основі хеш функції keccak лежить конструкція під назвою губка, цю конструкцію можна глянути на рисунку 2.2.

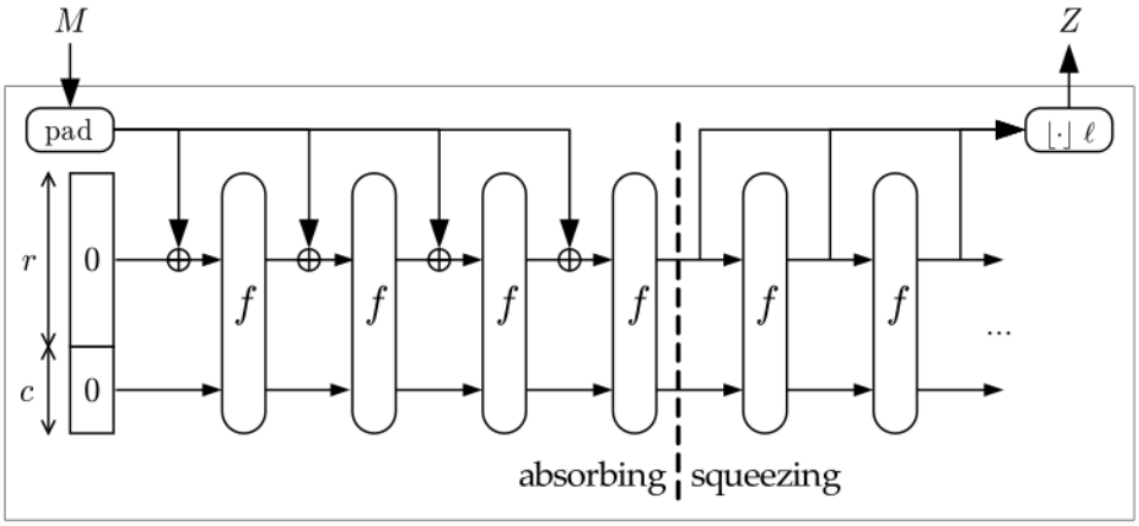

Рисунок 2.2 – Конструкція keccak

Тобто, щоб получити хеш файлу нам потрібно зробити наступні дії:

– Взяти вихідне повідомлення M та доповнити його до довжини кратної r. Правила доповнення полонять своєю простотою. Як формули їх можна зобразити так: M=M||0x01||0x00||..||0x00||0x80. Або російською мовою, до повідомлення дописується одиничний байт, необхідна кількість нулів і весь цей ансамбль завершує байт зі значенням 0x80.

– Додавання по модулю 2 з першими r-бітами набору початкових станів S. Перед початком роботи функції всі елементи S дорівнюватимуть нулю.

– N разів застосовуємо до отриманих в результаті даних функції f.

Набір початкових станів S для блоку Mi+1 буде результатом останнього раунду блоку Mi.

– Після того як всі блоки Mi закінчаться взяти підсумковий результат і повернути його як хеш-значення.

Блок-схема алгоритм пошуку keccak зображено на рисунку 2.3.

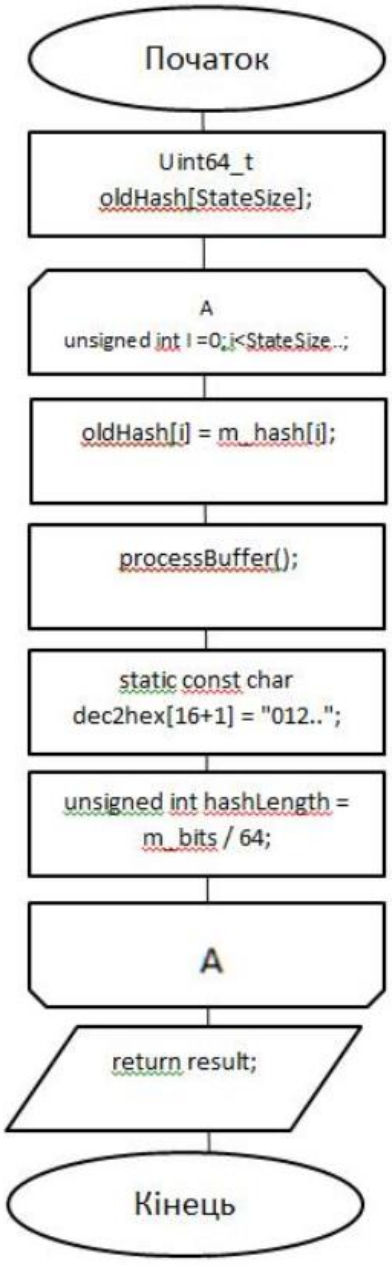

Рисунок 2.3 – Блок схема алгоритму keccak

На етапі Absorbig проводиться обчислення хеш значення. А на етапі Squeezing виведення результатів доки не буде досягнуто необхідної довжини хешу.

2.2. Проєктування структури системи

Мета дипломного дослідження полягає в розгляді основних методів моніторингу та виявлення аномалій мережевого трафіку, які спричинено шкідливими програми. Для досягнення мети дипломного проєкту у моєму програмному засобі будуть розроблятися такі компоненти:

- Сканування певного файлу на зараження вірусом;
- перевірка цілої папки файлів та вкладених файлів;
- сканування всієї системи та переведення заражених файлів у карантин;
- видалення заражених файлів;
- моніторинг завантаженості ПК;
- оновлення бази даних вірусів на основі результатів сканування.

Виходячи із компонентів програми, складається бачення взаємодії об'єктів впорядкованих за часом. Приклад зображено на рисунку 2.2.

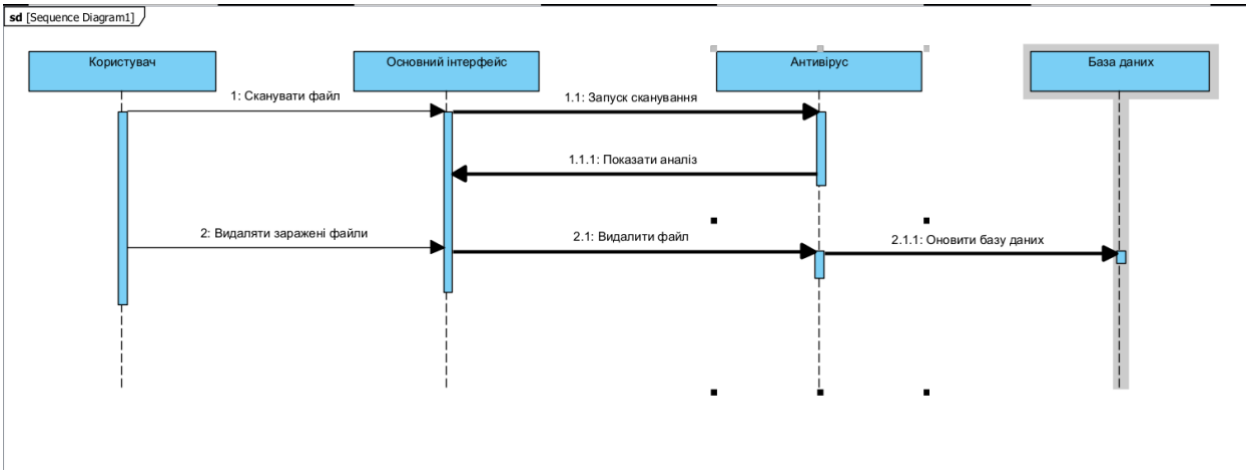

Рисунок 2.2 – Діаграма послідовності

На діаграмі зображено «цикл» програмного засобу, який починається із сканування файлу (запуску сканування, показу його аналізу та видалення зараженого файлу) та закінчується оновленням бази даних вірусів.

2.3 Дослідження алгоритму побудови Геш значень

Геш-функція – це функція перетворення масиву вхідних даних довільної довжини в вихідний бітовий рядок фіксованої довжини, що виконується за допомогою певного алгоритму. Функція однонаправлена, тому що маючи вихідне значення, тобто число певної довжини, неможливо визначити, які дані були подані на вхід. Функція вважається криптографічно стійкою в тому випадку, коли її можна атакувати лише «грубою силою», тобто перебирати всі можливі варіанти вхідних значень і намагатися знайти вихідне значення, що відповідає заданому. Схематично принцип роботи гешфункції буде виглядати, як зображено на рисунку 2.3.

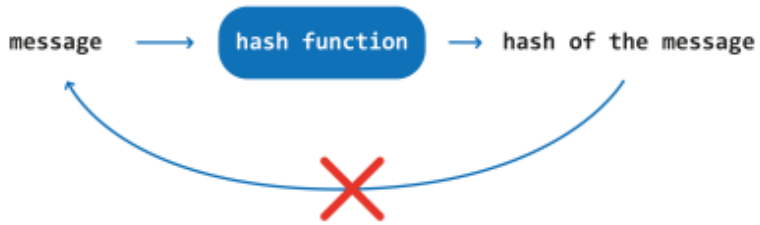

Рисунок 2.3 – Принцип роботи Геш-функції

На вхід геш-функція може приймати повідомлення практично необмеженого розміру (будь-які дані), а на виході – унікальний ідентифікатор – геш-значення. Маючи геш-значення, отримане за допомогою криптографічно стійкої геш-функції, неможливо визначити, які дані були подані їй на вхід. Більш того, в Bitcoin використовується кілька різних алгоритмів гешування.

Функцію гешування можна зрозуміти на прикладі такої функції, яка підраховує суму цифр у вхідному повідомленні. Така функція має наступні особливості.

По-перше, в цьому випадку, результат обчислюється досить швидко за рахунок простоти алгоритму підрахунку.

По-друге, така функція дійсно є однонаправленою, так як вихідне значення містить дані тільки про суму цифр у повідомленні, але не про їх

розташування. Тобто складність відновлення вихідного повідомлення зростає з його довжиною, рисунок 2.4.

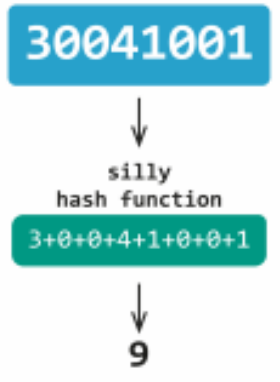

Рисунок 2.4 – Вхідне повідомлення

Однак використання такої функції не є безпечним з точки зору простоти знаходження колізій, так як в цьому випадку дуже просто замінити вхідне повідомлення (тобто подати на вхід функції повідомлення з тими ж числами, що вказані на рисунку 2.5.

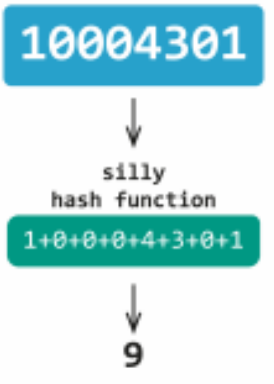

Рисунок 2.5 – Приклад колізії геш-функції

Давайте ж розберемося, які властивості повинна мати гешфункція, щоб її використання було максимально безпечним для користувача.

Стійкість геш-функції до колізій означає, що в ній відсутній алгоритм, що дозволяє знаходити колізії за відносно короткий час. Колізією називають ситуацію, коли існує пара значень, подавши які на вхід геш-функції ми отримаємо один і той же вихідний результат, показано на рисунку 2.6.

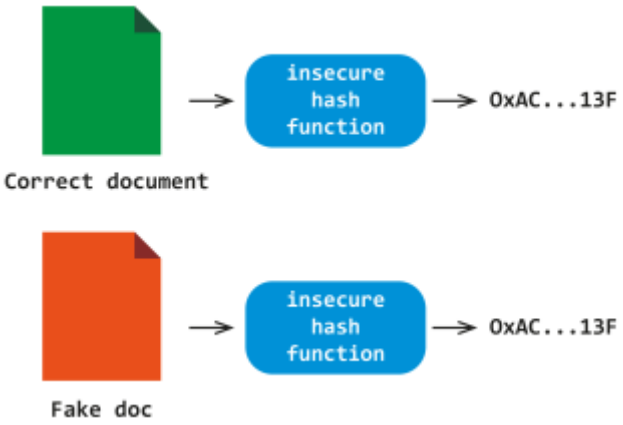

Рисунок 2.6 – Стійкість геш-функції

Стійкість до пошуку першого прообразу (незворотність) є вимогою до геш-функції, при виконанні якої неможливо відновити вхідне повідомлення за реальний час, знаючи тільки відповідне йому геш-значення. Стійкість до пошуку другого прообразу є вимогою, що має на увазі, що при її виконанні сторона, що має вихідне повідомлення і відповідне йому геш-значення, не може створити ще одне повідомлення, що на виході геш-функції надасть той же самий результат. Також криптографічно стійка геш-функція повинна мати властивість, згідно з якою зміна одного біту на вході повинна призводити до зміни близько половини вихідних бітів. Проілюструвати на прикладі подібну властивість можна наступним чином рисунок 2.7. Допустимо, що ми формуємо транзакцію, в якій вказано, що ми пересилаємо Алісі 10 біткоінів, і обчислюємо відповідне геш-значення. Якщо хтось забажає підмінити транзакцію і дописати, наприклад, що переводиться не 10 біткоінів, а 100, то факт підробки транзакції буде виявлений відразу

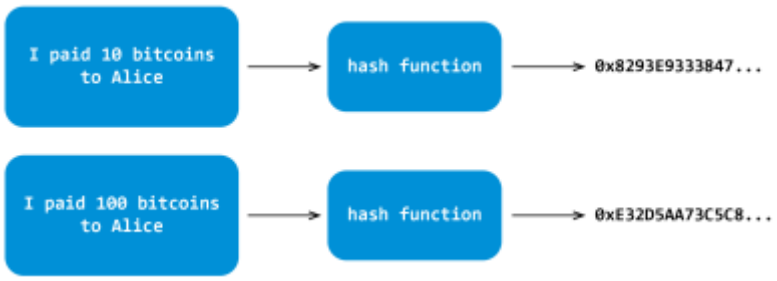

Рисунок 2.7 – Зміна вихідного значення геш-функції в залежності від вхідного

Як приклад однієї з достатньо сучасних геш-функцій, принцип функціонування якої досить просто пояснити, наведемо SHA-1. Фактично, робота цієї функції полягає в циклічному перемішуванні (80 циклів) і використанні основних бітових операцій (And, Xor, Rot, Add, Or) з використанням вхідних даних. На рисунку 2.8 представлений один цикл роботи геш-функції SHA-1.

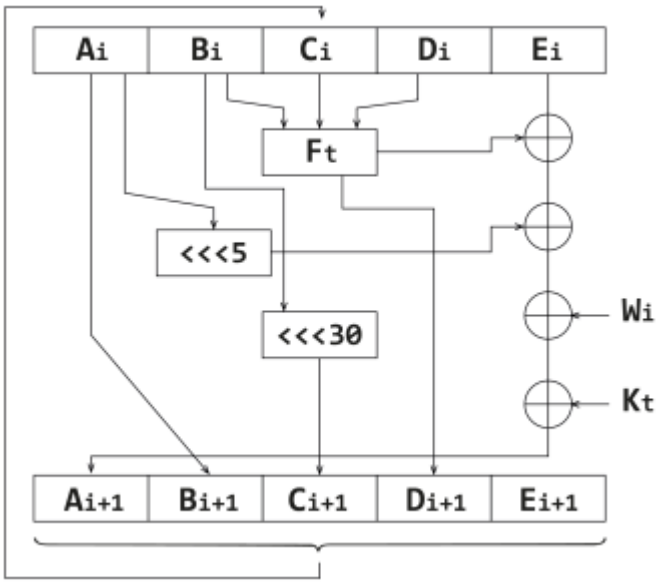

Рисунок 2.8 – Конструкція SHA-1

Kt – константа

Ft – перемінна функція(змінюється кожні 20 циклів)

Wi – модифікований елемент вихідного повідомлення (4 байта)

<<<x – циклічний зсув вліво на x позицій

Очевидно, що гешування дуже зручно використовувати для отримання унікального ідентифікатора набору даних. У разі Bitcoin, роль геш-функції полягає у timestamping і зв'язуванні блоків, а також зберіганні виключно доказу даних, а не самих даних. Як наслідок, геш-функції набули широкого поширення. Розглянемо основні варіанти їх застосування. Геш-значення використовуються в якості контрольних сум при передачі даних. Для того щоб перевірити, що повідомлення не було випадково порушено через якихось шумів у каналі передачі даних, сторона-отримувач може повторно

обчислити геш-значення від отриманих даних і порівняти його з вже наявними. Крім цього, такі функції використовуються для пошуку дублікатів при зберіганні або для порівняння великих масивів даних. Щоб не порівнювати великі обсяги даних безпосередньо, можна зберігати відповідні їм значення геш-функцій і порівнювати тільки ці значення. Якщо значення геш-функції для різних наборів даних збігаються, значить з дуже великою ймовірністю і самі дані збігаються. Це значно прискорює процес, наприклад, при формуванні цифрового підпису. Зазвичай при підписанні документа підписуються не самі дані повідомлення, а їх геш-значення. При цьому вважається, що це гешзначення передається разом з повідомленням і підписом, щоб спочатку одержувач міг перевірити цілісність повідомлення. А потім коректність цифрового підпису.

Концепція побудови таких дерев вперше була опублікована в 1979 році Ральфом Мерклем. Дерева Меркла представляють собою структуру даних, яка дозволяє зв'язати окремі фрагменти даних в 3. Вступ до криптографії та управління ключами єдине кореневе значення та після довести, що певний блок даних дійсно має відношення до конкретного кореневого значення. Дерево Меркла містить у собі такі компоненти (рис. 2.9): листя дерева (Merkle leaves); вузли дерева (Merkle nodes); корінь дерева (Merkle root).

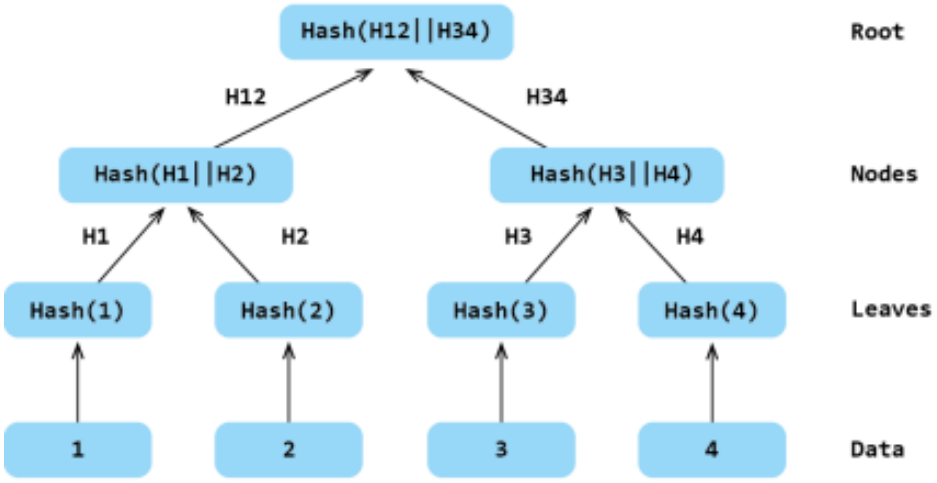

Рисунок 2.9 – Структура дерева Меркла

Листя дерева Меркла – геш-значення від блоків даних, які необхідно зібрати в структуру. Вузол дерева є значенням, яке було отримано в результаті конкатенації та подальшого гешування двох дочірніх вузлів або листів. Корінь дерева Меркла також являє собою вузол, що знаходиться на вершині дерева.

2.4 Проєктування бази даних

Для виявлення в файлах вірусів застосовується, як згадувалось раніше, метод пошуку за контрольною сумою. Для цього була розроблена база даних вірусів.

Зазначена база даних складається із хешів вірусів, захешованих алгоритмом SHA-256.

Хешування – це процес перетворення заданого ключа в код. Хеш-функція використовується для заміни інформації щойно згенерованим хеш-кодом. Точніше, хешування – це практика отримання рядка або ключа введення, змінної, створеної для зберігання описових даних, і представлення її за допомогою хеш-значення, яке зазвичай визначається алгоритмом і становить набагато коротший рядок, ніж оригінал.

Алгоритм SHA-256 є одним із різновидів SHA-2 (Secure Hash Algorithm), який був створений Агентством національної безпеки в 2001 році як наступник SHA-1. SHA-256 – це запатентована криптографічна хеш-функція, яка виводить значення довжиною 256 біт.

Під час шифрування дані перетворюються в захищений формат, який неможливо прочитати, якщо одержувач не має ключа. У зашифрованому вигляді дані можуть мати необмежений розмір, часто стільки ж, скільки й у незашифрованому вигляді. У хешуванні, навпаки, дані довільного розміру відображаються на дані фіксованого розміру. Наприклад, 512-бітовий рядок даних буде перетворений на 256-бітовий рядок за допомогою хешування SHA-256.Файл з яким зрівнюються SHA256 знаходиться в проєкті і називається SHA256.txt, можна побачити на рисунку 2.10.

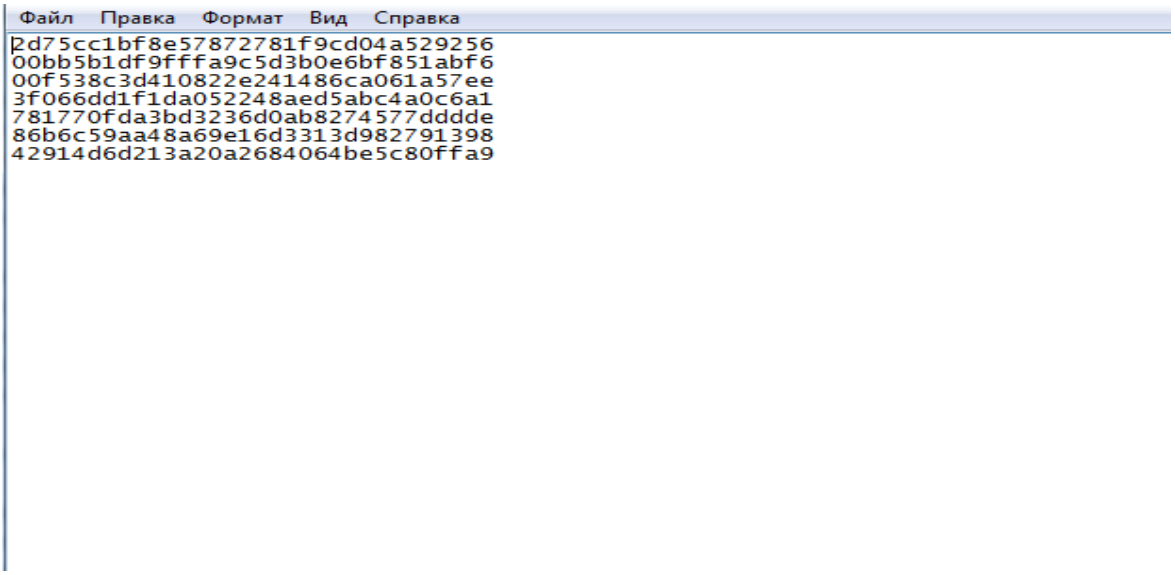

Рисунок 2.10 – Файл з хешами вірусів

Було обрано реалізацію бази даних в вигляді текстового файлу, оскільки зручно формувати базу даних необхідних файлів.

2.5 Проєктування користувацького інтерфейсу

Для побудови користувацього інтерфейсу потрібно проаналізувати основні вимоги до програмного засобу. Отже, функціонал користувача полягає в наступному:

- зручний графічний набір компонентів та кнопок;
- наявність головного меню та діалогових вікон;
- можливість сканування жорсткого диску та порівнянняз геш значенням в базі даних.

Для побудови графічного інтерфейсу доцільно обрати середовище швидкої розробки (RAPID Aplication Development). Для цієї задачі підходить RadStudio.

На рисунку 2.11 приведено діаграму прицедентів для проєктованого програмного засобу.

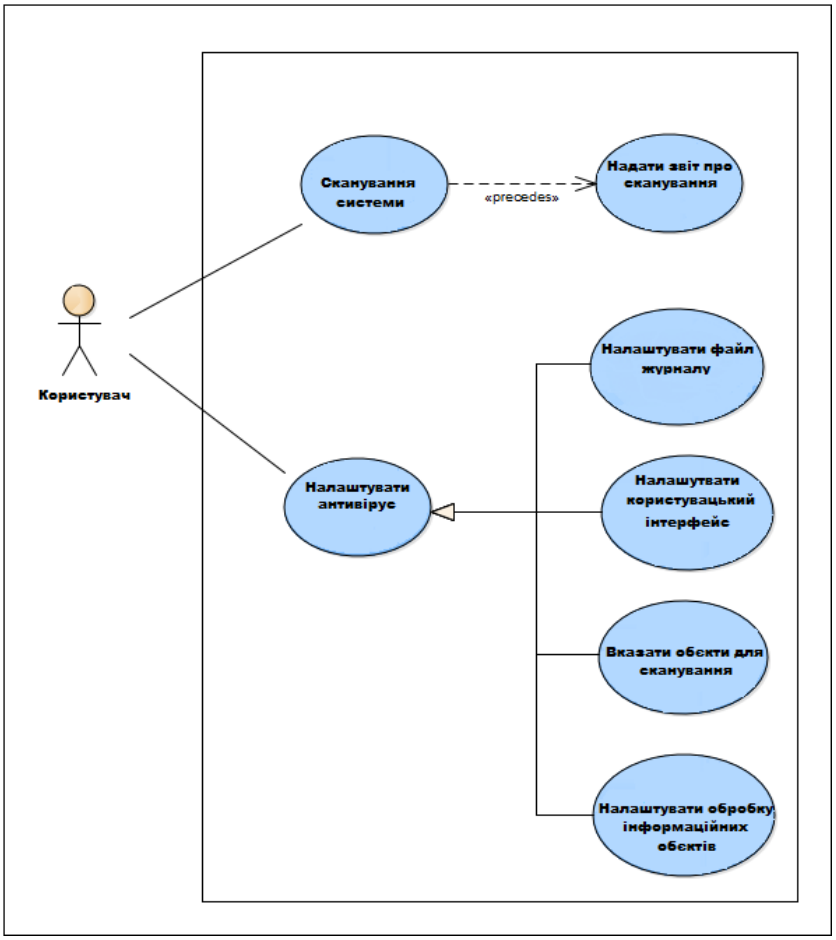

Рисунок 2.11 – Діаграма прецедентів

Із діаграми випливає, що функціями користувача є:

– Сканування системи;

– налаштування антивіруса.

Форма «сканування системи» повинна надавати можливість отримання звіту про сканування.

Форма «налаштування системи» в свою чергу, повинна надавати можливість налаштування файлу журналу, налаштування користувацького інтерфейсу, вказувати об’єкти для сканування та налаштування обробки інформаційних об’єктів.

При проєктуванні програмного продукту було розроблено найпримітивніший інтерфейс, що представлений на рисунках 2.12-2.13, ціль його - переведення програмного коду у виконуваний файл.

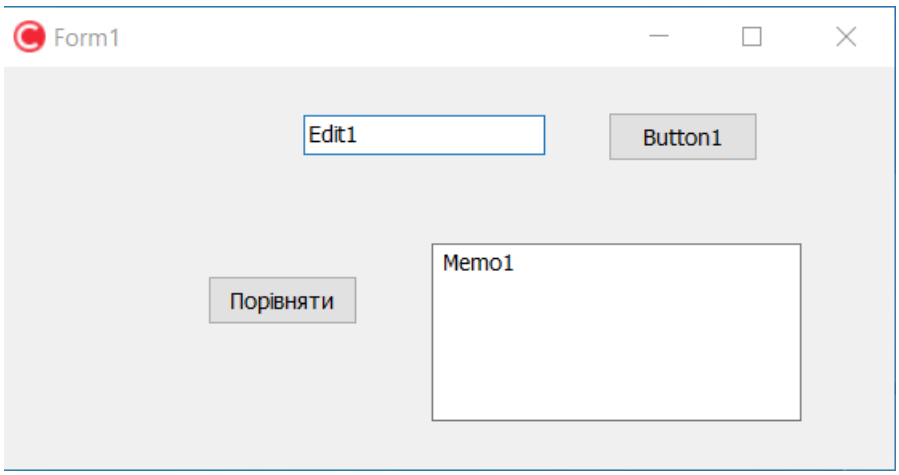

Рисунок 2.12 – Вікно перевірки файлу

Тут було зображено форму, в якій після вибору файлу в поле Edit1 виводиться хеш вибраного файлу, який в подальшому після натискання кнопки порівняти, порівнюється з базою хешів вірусів.

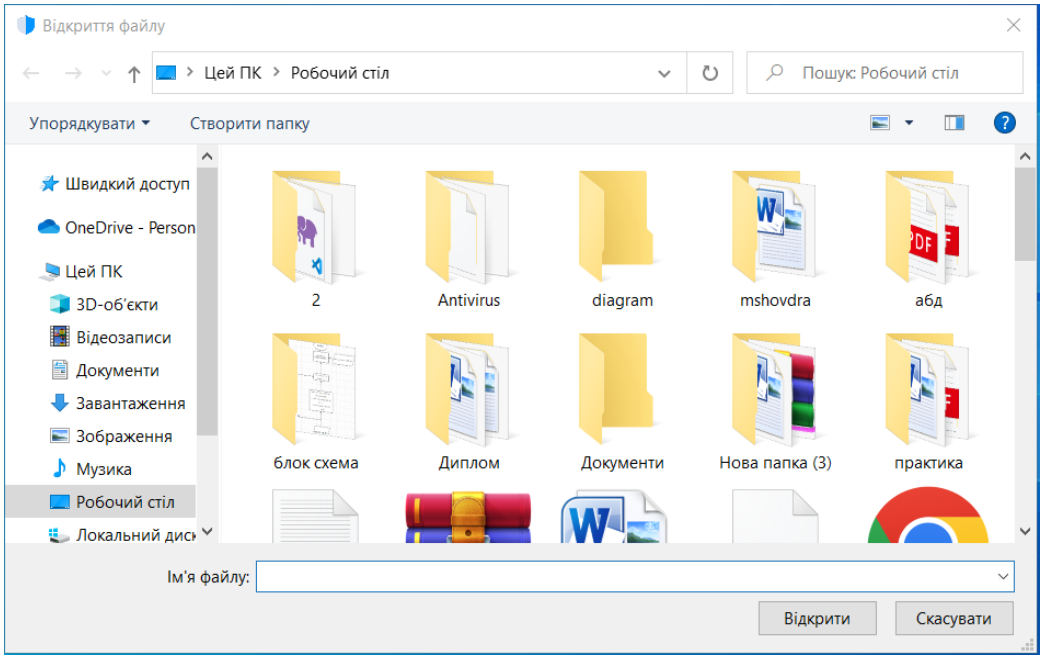

Рисунок 2.13 – Вікно для вибору файлів

На цьому рисунку було зображено файловий провідник, за допомогою якого можна вибирати потрібний файл для перевірки.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Опис технологій та засобів реалізації

Для того, щоб максимально виконати завдання, які були зазначенні на початку виконання дипломного проєкту, серед інших інструментів розробки, було обрано таку мову програмування, як C++. Вона забезпечить можливість надання простого та доступного користувацького інтерфейсу програмного засобу. Одним з найкращих інструментів для створення потрібного користувацького інтерфейсу є бібліотека WinApi (Windows Application programming interfaces).

Для розробки програмного забезпечення було використане таке інтегроване середовище розробки, як Embarcadero C++ Builder, яке було завантажене з офіційного сайту, показаного на рисунку 3.1.

Embarcadero RAD Studio – це середовище розробки десктопних програмних засобів для операційної системи Windows, але також підтримує розробку під Mac OS,Apple IOS та Android.

Рисунок 3.1 – Офіційний сайт Embarcadero

В процесі інсталяції було обрано шлях для встановлення, і компоненти, які зображено на рисунку 3.2. Середовище Embarcadero RAD Studio може працювати в демо – режимі, що дозволяє розробляти невеликі проєкти. Також наявна Comuniti ліцензія.

Рисунок 3.2 – Компоненти для встановлення

На етапі встановлення необхідно обрати платформу для розробки програмного засобу. Наступним кроком буде створення проєкту. Потрібно запустити середовище розробки Rad studio, показано на рисунку 3.3.

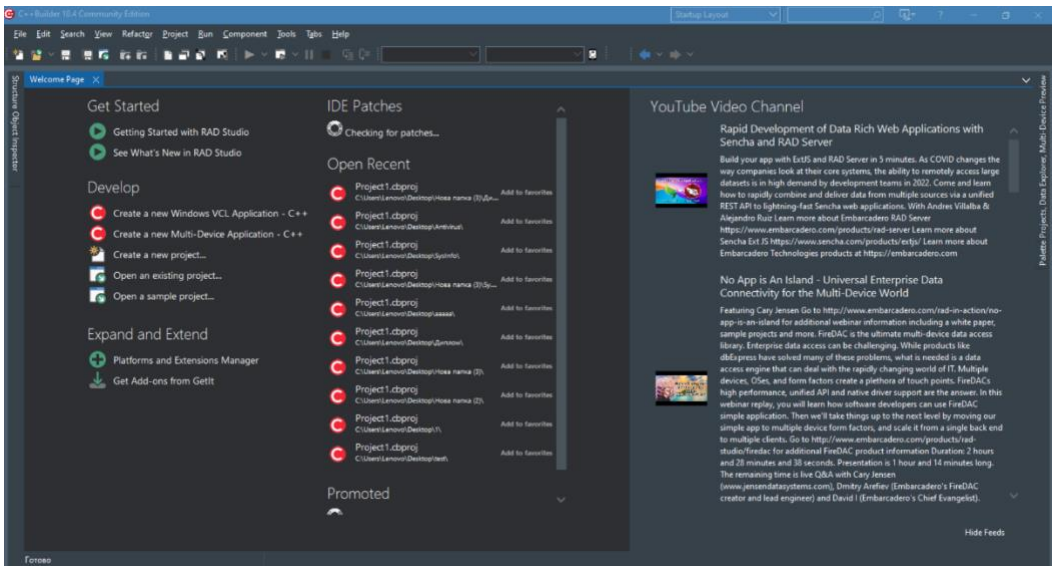

Рисунок 3.3 – Середовище розробки

Після того, як відкрилося середовище розробки потрібно створити проєкт, як на рисунку 3.4.

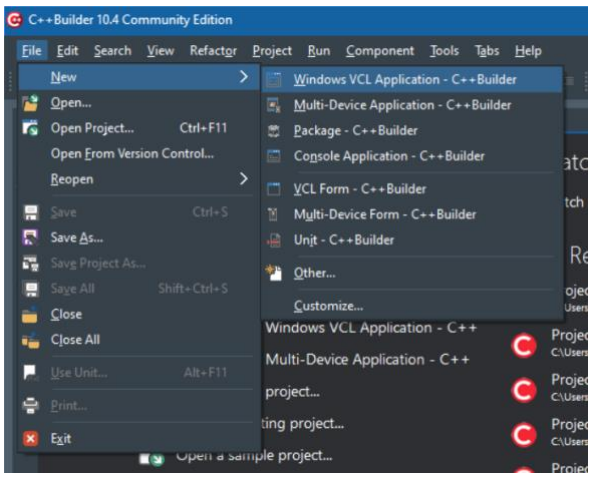

Рисунок 3.4 – Створення нового проєкту

Головне меню файл, надає можливості створення проєктів різних типів. Дальше буде створеня форма, в якій можна буде займатися проєктуванням застосунку, показано на рисунку 3.5.

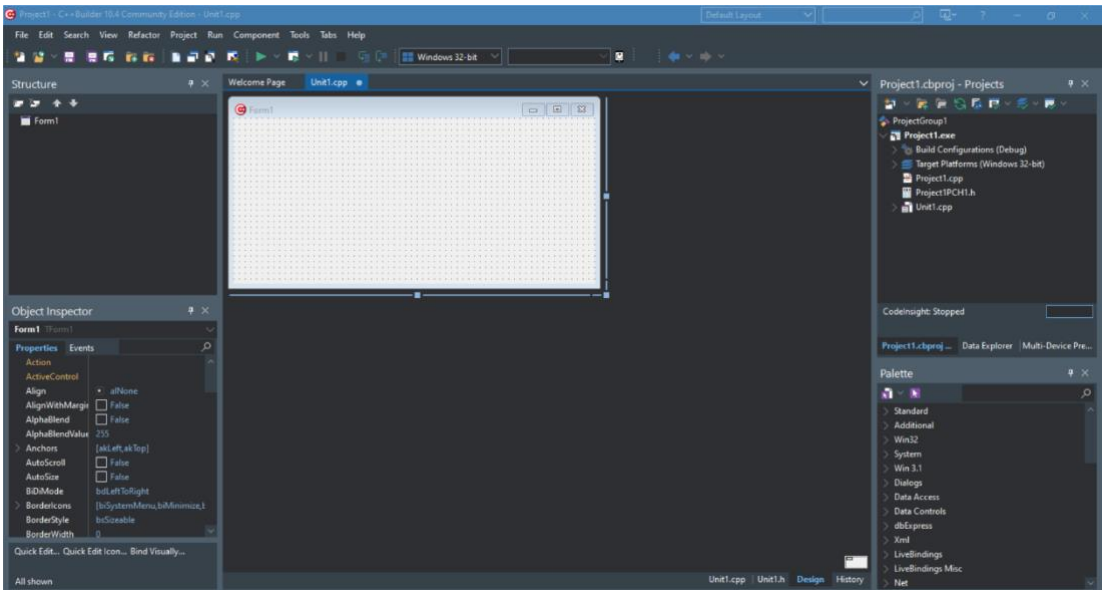

Рисунок 3.5 – Створення форми.

Середовище проєктуваннння надає зручні інструменти для швидкої розробки графічного дизайну. Далі було підключено бібліотеку, в якій знаходяться алгоритми пошуку хешів, як на рисунку 3.6.

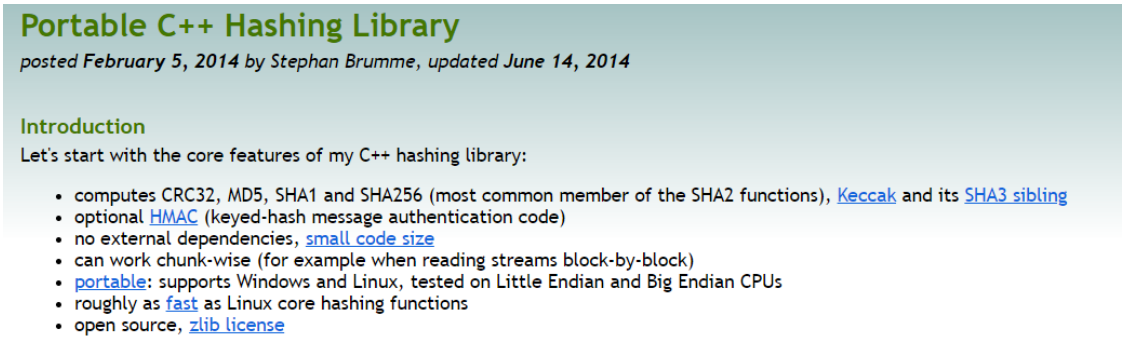

Рисунок 3.6 – Сайт бібліотеки хешів

В результаті завантаження файлу отримуємо архів, в якому зберігаються алгоритми для пошуку хешів, зображено на рисунку 3.7. Для вибору бібліотеки з алгоритмами обчислення хеш функцій обрано бібліотеку написану на мові С. Бібліотека достатньо універсальна та дозволяє виконувати обчислення використовуючи різні алгоритми обчислення Геш значень.

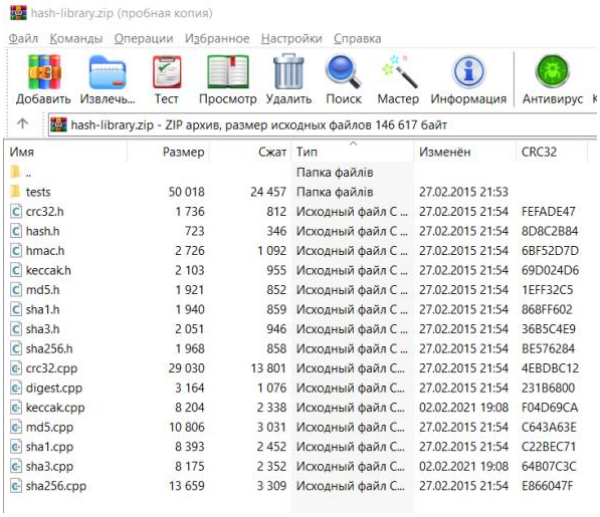

Рисунок 3.7 – Архів з файлами бібліотеки

Кожен алгоритм обчислення Геш функції виконаний в вигляді cpp модуля. Наступним кроком є переміщення файлів з архіву в кореневу папку програми, як видно на рисунку 3.8.

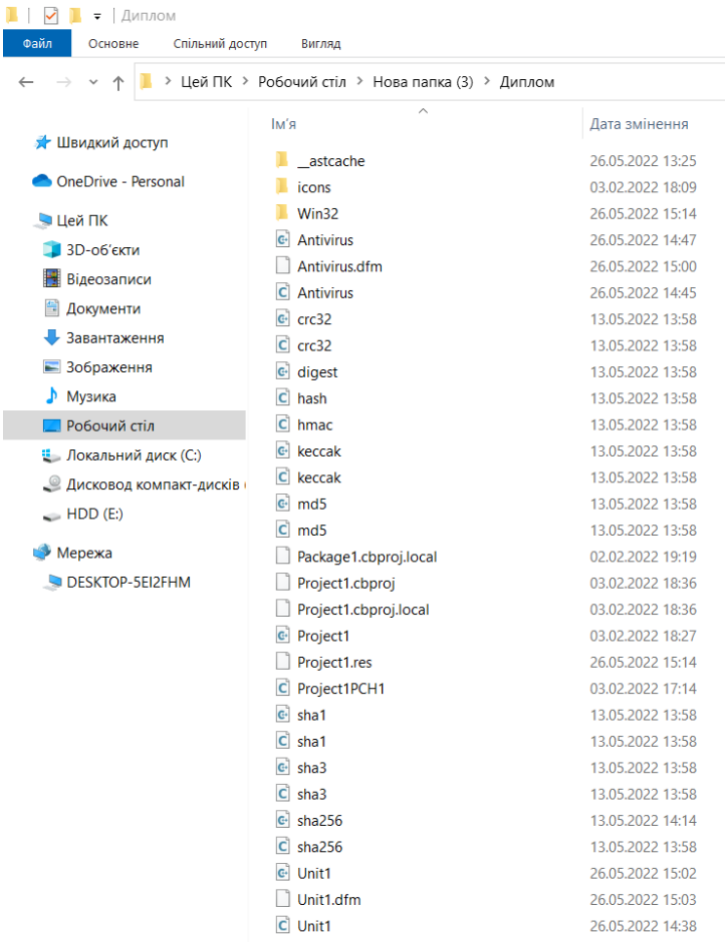

Рисунок 3.8 – Коренева папка програми

Після успішного завантаження файлів потрібно підключити файли з бібліотеки в IDE, як зображено на рисунку 3.9.

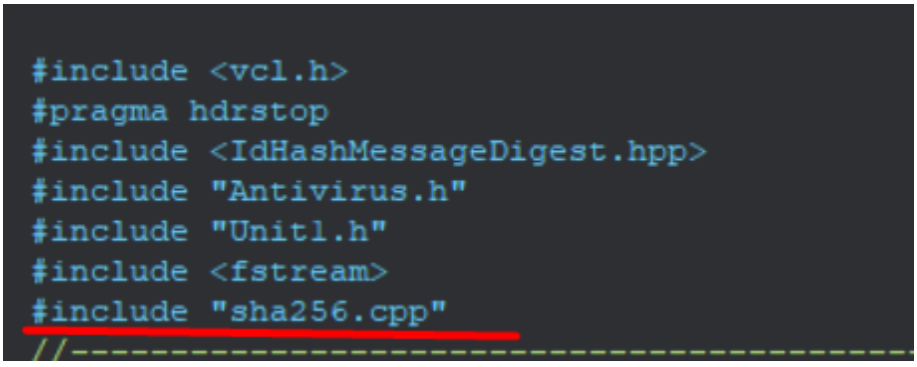

Рисунок 3.9 – Підключення файлу з бібліотеки

Для підключення бібліотеки обчислення хеш значень достатньо використовувати підключення cpp файлу та додати файли в проєкт. Також в проєкт додано бібліотек fstream для роботи з файлами.

3.2 Реалізація основних функцій системи

Для створення програмного засобу необхідно виділити основні функції системи. Для реалізації проєкту в середовищі RAD Studio створено відповідні модулі, функції та класи. Програмний засіб складається з такого функціоналу, як:

– надання інформації про систему;

– файловий менеджер;

– перевірка файлу;

– вихід.

Для більшого розуміння функціоналу програмного засобу, пропоную провести детальний аналіз функцій.

1) Інформація про систему – це окрема форма, в якій подано інформацію про завантаженість системи у відсотках, а також є графік завантаженості ПК;

2) файловий менеджер – спливаючий список, за допомогою якого можна вибрати потрібний файл для подальшого сканування його на віруси;

3) перевірка файлу – використовується для подальшої перевірки вибраного файлу на віруси, якщо хеш вибраного файлу і хеш з бази даних вірусів співпадає, то цей файл заражено, тоді його потрібно перевести в карантин і ліквідувати;

4) вихід – завершення програми.

Описані основні функції надають основний функціонал, що дозволить виконати поставлені завдання.

3.3 Реалізація користувацького інтерфейсу системи

Для втілення функціоналу було створено форму користувацького інтерфейсу програми, яка застосовується для зберігання на ній різних компонентів програми, зображено на рисунку 3.10.

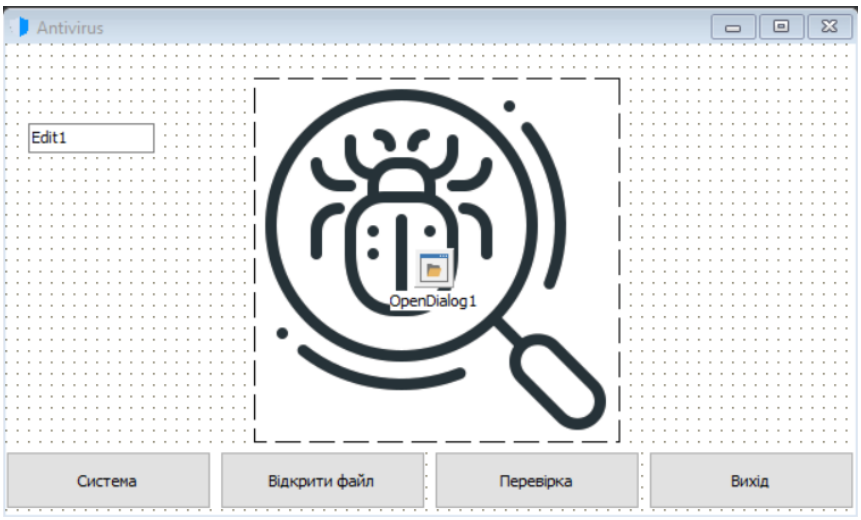

Рисунок 3.10 – Форма компонентів

Доступ до функціональної частини надає панель з кнопками. Наступним кроком було розміщення на формі кнопок для навігації, показано на рисунку 3.11.

Рисунок 3.11 – Кнопки навігації

Пізніше було добавлено логотип на форми і значок програми, рисунок 3.12 – 3.13.

Рисунок 3.12 – Логотип на формі

Було створено іконку для головного вікна, як це показано на рисунку 3.13.

Рисунок 3.13 – Значок програми

Для реалізації файлового менеджера було створено такий компонент, як OpenDialog, показано на рисунку 3.14.

Рисунок 3.14 – Компонент OpenDialog

В подальшому для реалізації пошуку хешу створено такий компонент, як Edit. В Edit буде виводитися хеш вибраної нами програми з файлового менеджера, див рисунок 3.15.

Рисунок 3.15 – Текстове поле Edit

Для відображення інформації про систему було створено окрему форму, яка відображена на рисунок 3.16.

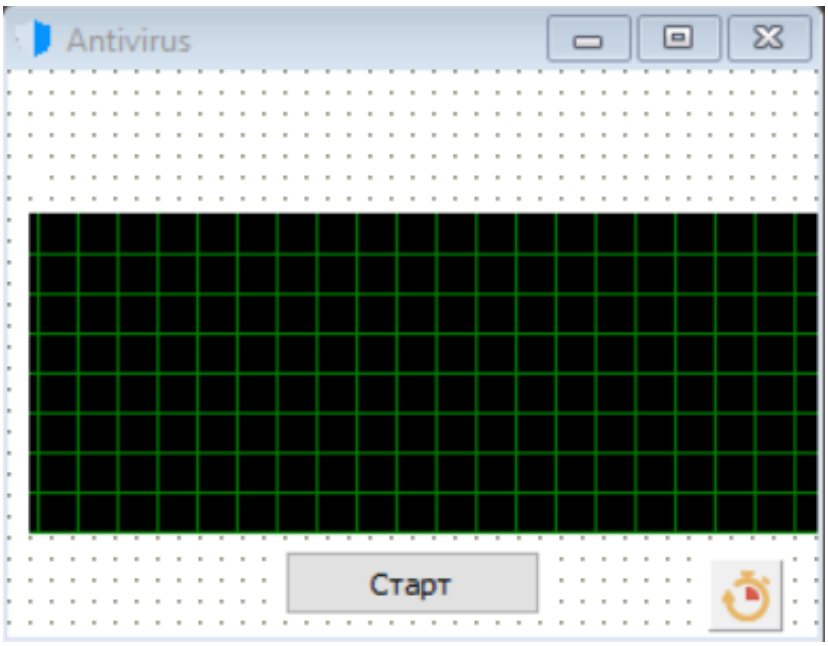

Рисунок 3.16 – Форма для відображення інформації про систему

На цій формі зберігаються такі компоненти, як Label – використовується для отримання інформації про кількість ядер процесора,тип процесора, рисунок 3.17.

кількість ядер : 4 / тип процессора : 586 / OemId : 0

Рисунок 3.17 – Поле Label

Також було використано таки елемент,як PerformanceGraph, за допомогою якого ми можемо отримати графік завантаженості ПК, рисунок 3.18.

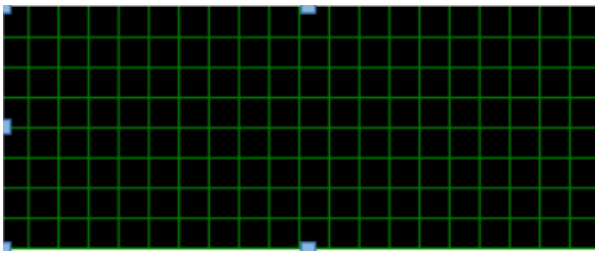

Рисунок 3.18 – PerformanceGraph

Кнопка, яка дозволяє запустити процес для отримання інформації, рисунок 3.19.

Рисунок 3.19 – Кнопка «старт»

Для відображення інформації про завантаженість процесора,використовується такий компонент, як Timer, рисунок 3.20.

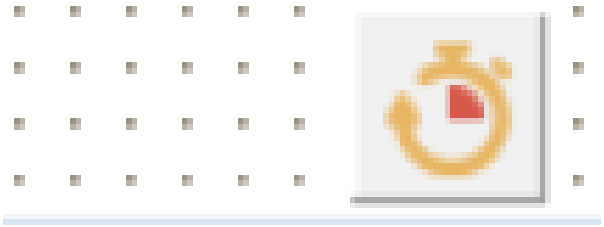

Рисунок 3.21 – Таймер

Компонент Таймер дозволяє відображати в режимі реального часу системні характеристики.

3.4 Тестування програмного засобу

В результаті успішного виконання поставленої мети дипломного проєкту, ми отримали готовий програмний засіб, який виявляє вірусний код у файлах.

Фінальним етапом виконання поставленої мети дипломного проєкту є тестування програмного засобу.

Для початку тестуємо таку функцію, як отримання інформації про систему, для цього на головні формі натискаємо на кнопку «система», а потім у іншій формі на кнопку «старт» .

Спочатку тестуємо функцію, коли ПК знаходиться у стані спокою. Результатом тестування є графік, представлений на рисунку 3.22.

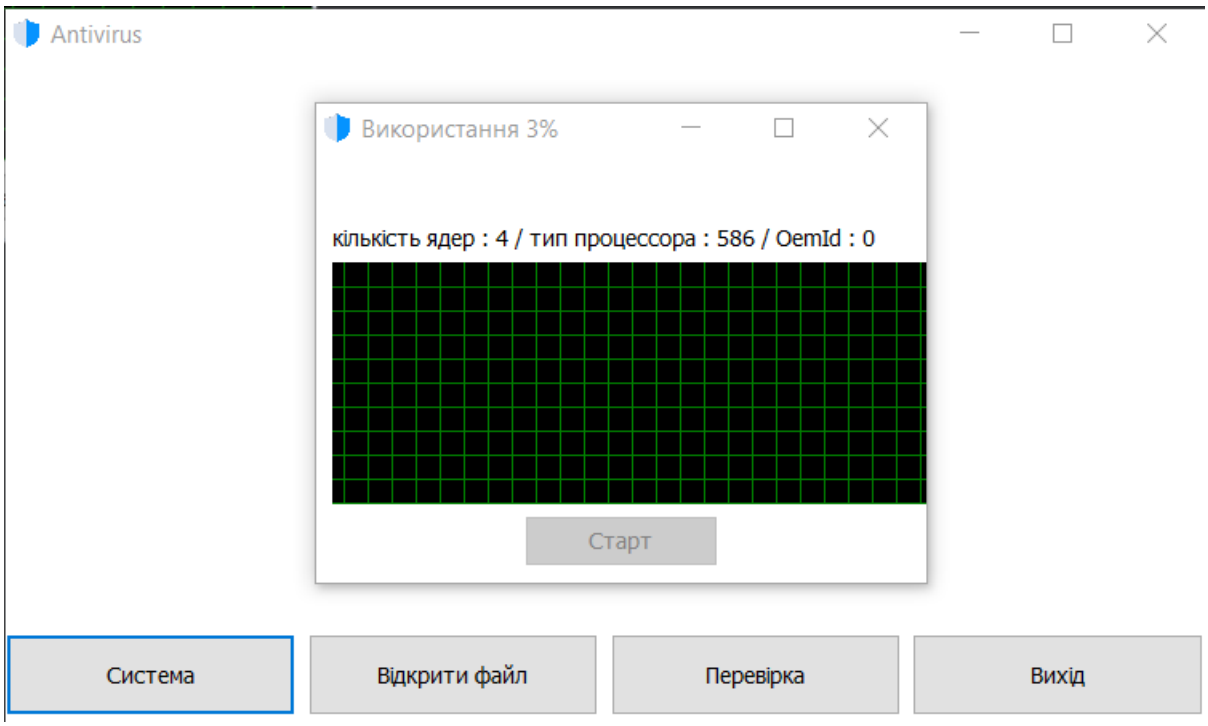

Рисунок 3.23 – Графік системи у стані спокою

Далі виконуємо будь-які задачі, для завантаження ПК, в результаті отримуємо наступний графік (дивись рисунок 3.24).

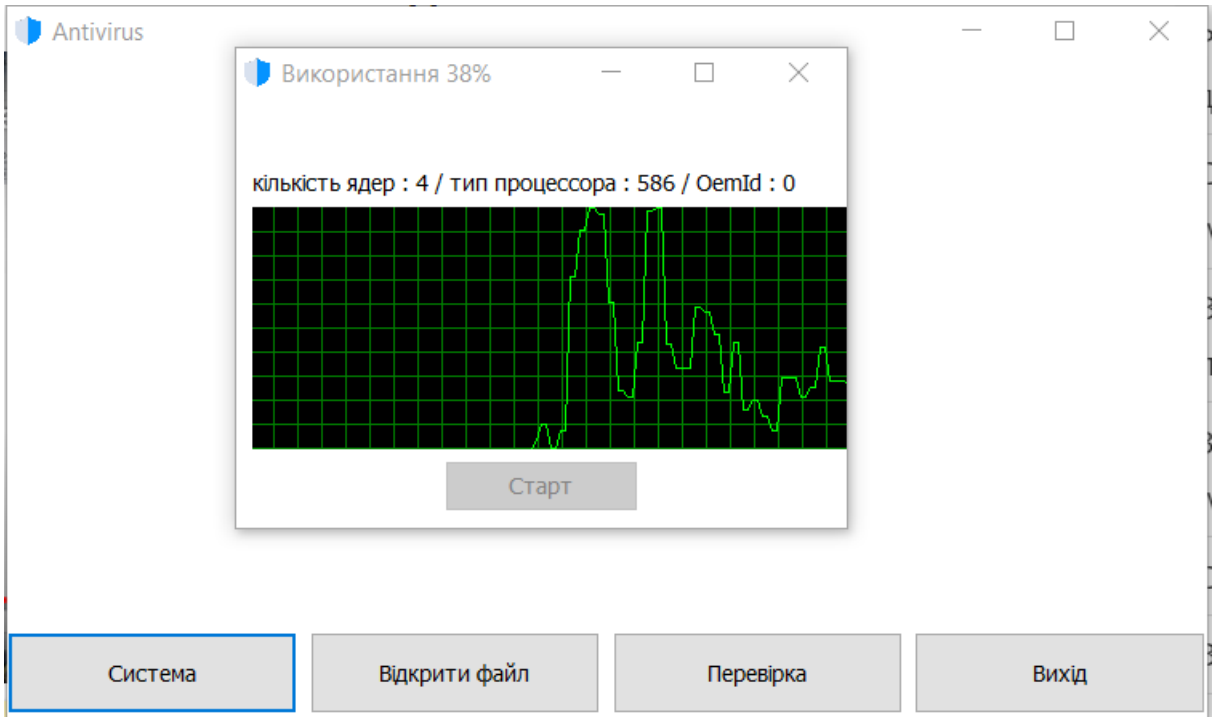

Рисунок 3.24 – Графік навантаженої системи

Наступним кроком є тестування файлового менеджера, для цього натискаємо на кнопку «відкрити файл», у спливаючому меню обираємо потрібний файл, для подальшої перевірки його на наявність вірусів, як показано на рисунку 3.25.

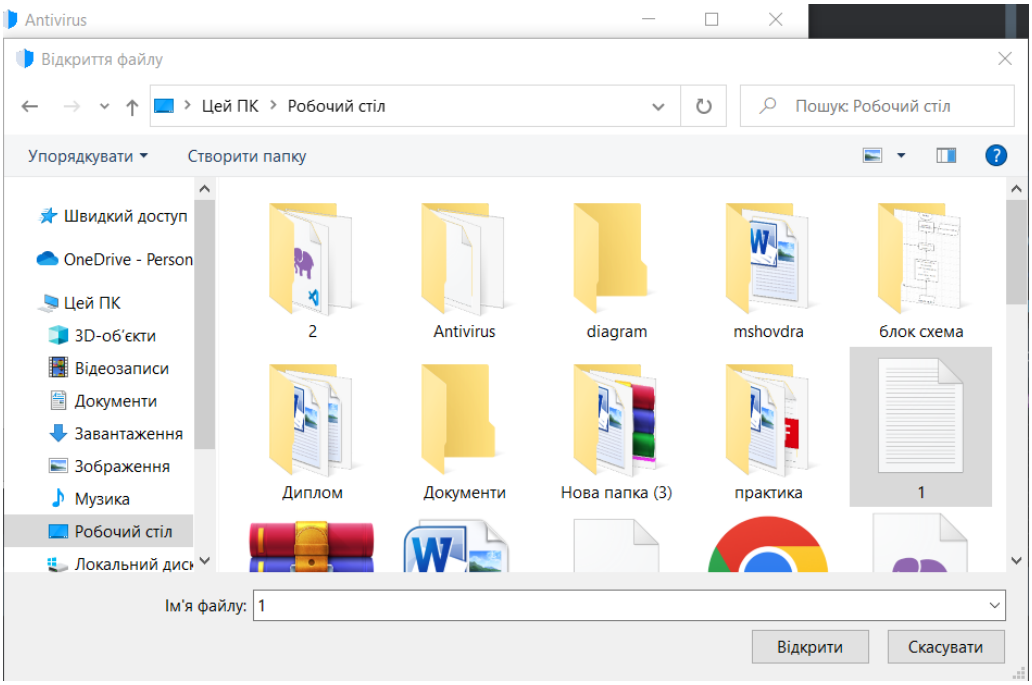

Рисунок 3.25 – Вибір документа для перевірки

Після вибору файлу, натискаємо кнопку «перевірка», яка відображає результат сканування. На рисунку 3.26 відображено результат перевірки незараженого шкідливим кодом файлу.

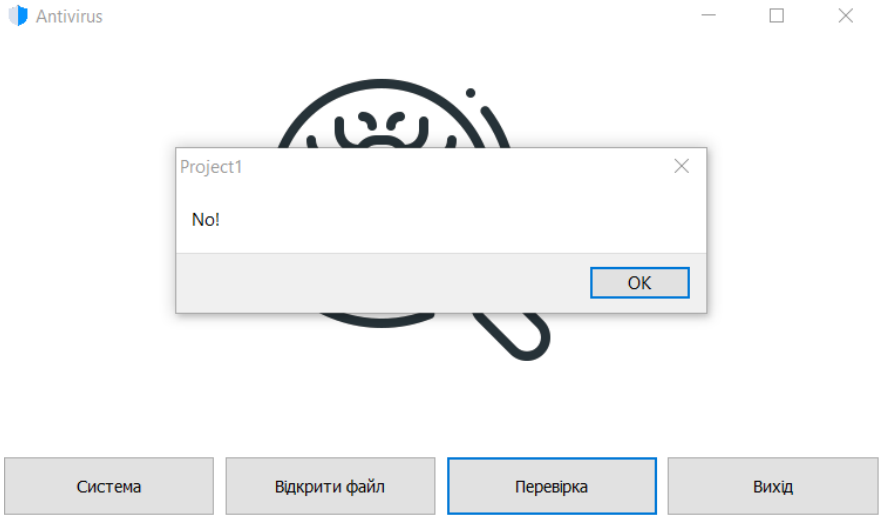

Рисунок 3.26 – Результат сканування незараженого файлу

Для повноцінного тестування програмного засобу потрібно просканувати файл, заражений вірусом. Саме такий результат показано на рисунку 3.27.

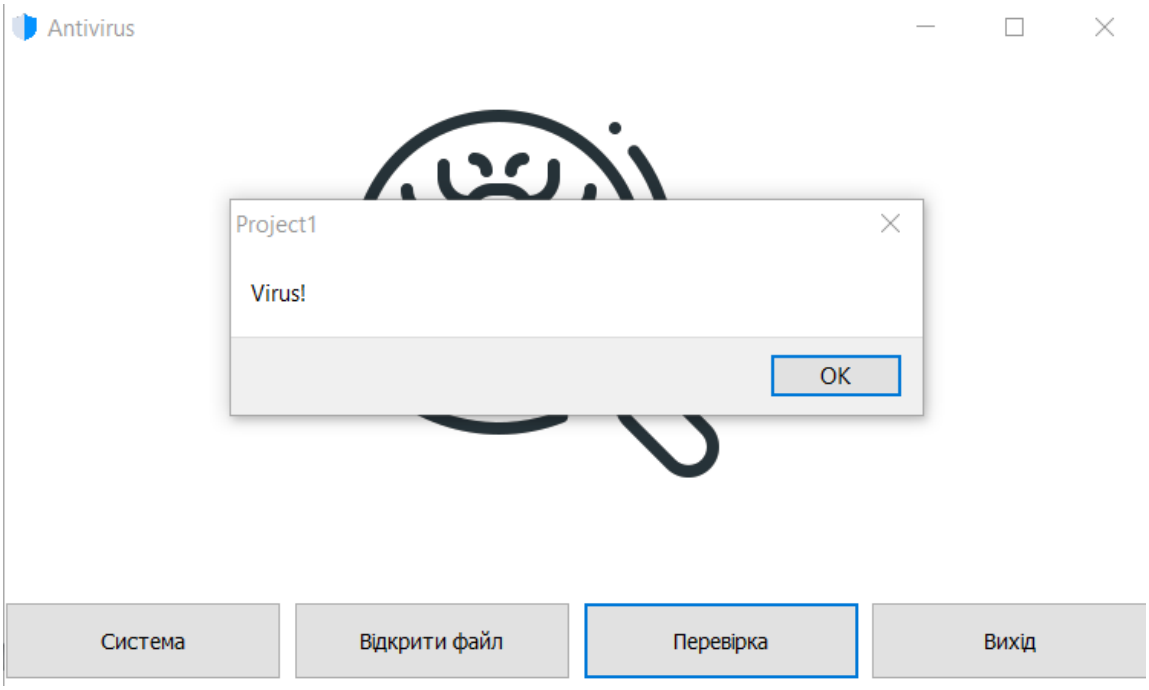

Рисунок 3.27 – Результат сканування зараженого файлу

Для завершення роботи із програмним засобом необхідно скористатися кнопкою «вихід».

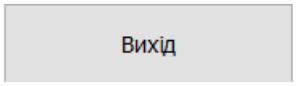

Рисунок 3.28 – Кнопка «вихід»

Для тестування програмного засобу було використано програмне забезпечення яке приведено на рисунку 3.29.

Складові ПК зображені у таблиці 3.1

Таблиця 3.1 – Конфігурація комп’ютера

Процесор	ОЗП	Тип системи	Відеокарта
Intel core i5-7300hq,2.50Ghz	8,00gb	x64	NVIDIA GeForce GTX 1050

В результаті тестування не було виявлено недоліків, які б могли завадити коректній роботі програмного засобу. Програмний засіб тестувався на персональному комп’ютері.

У таблиці 3.1 було подано перелік комплектуючих на яких відбувалося тестування програмного засобу.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку

Ринок антивірусного програмного забезпечення складається з продажу антивірусного програмного забезпечення юридичними особами (організаціями, приватними підприємцями та партнерствами), які використовуються для захисту комп'ютерів від вірусів шляхом їх сканування, виявлення та видалення.

Більшість антивірусних програм після завантаження працюють у фоновому режимі, забезпечуючи захист від вірусних атак у режимі реального часу. Поведінка всіх програм контролюється антивірусним програмним забезпеченням, яке позначає будь-яку сумнівну поведінку.

Основними типами антивірусного програмного забезпечення є комп'ютери, планшети, смартфони та інші. Комп'ютерне антивірусне програмне забезпечення використовується в комп'ютерах для запобігання, сканування та виявлення вірусів і шкідливих програм, які завдають шкоди комп'ютеру.

Різні операційні системи включають Windows, MAC, Android або IOS або Linux і використовуються різними вертикалями, такими як корпоративні, особисті, державні.

Очікується, що глобальний ринок антивірусного програмного забезпечення зросте з 3,92 мільярда доларів у 2021 році до 4,06 мільярда доларів у 2022 році при сукупному річному темпі зростання (CAGR) 3,6%. Зміна тенденції зростання ринку антивірусного програмного забезпечення в основному пов'язана зі стабілізацією виробництва компаній після задоволення попиту, який експоненціально зріс під час пандемії COVID-19 у 2021 році.

Очікується, що ринок досягне 4,75 мільярда доларів у 2026 році при середньому темпі зростання 4,0 %.

Основними гравцями на ринку антивірусного програмного забезпечення є Symantec, McAfee, ESET, Trend Micro, F-Secure, BitDefender, G Data CyberDefense, Fortinet, Microsoft Corporation, Cheetah Mobile, AVG Technologies, Qihoo 360, Quick Heal, Tencent, Comodo Cybersecurity, Kaspersky , AhnLab Inc, Ad-Aware, Panda Security та Lavasoft.

Північна Америка була найбільшим регіоном на ринку антивірусного програмного забезпечення в 2021 році. Європа була другим за величиною регіоном на ринку антивірусного програмного забезпечення. Регіони, охоплені у звіті про ринок антивірусів: Азіатсько-Тихоокеанський регіон, Західна Європа, Східна Європа, Північна Америка, Південна Америка, Близький Схід та Африка.

Країни, охоплені у звіті про ринок антивірусного програмного забезпечення: Австралія, Бразилія, Китай, Франція, Німеччина, Індія, Індонезія, Японія, Росія, Південна Корея, Великобританія, США.

4.2 Розрахунок витрат на проєктування

Для того,щоб почати підготовку для розробки проєкту,потрібно провести основні критерії оцінки ринку.

Важливою частиною розробки програмного засобу як стартап-проєкту є проведення економічних розрахунків і визначення аналізу на попит і пропозицію продукту за такими показниками: обсягу, рівень прибутковості ,коефіцієнту конкурентноспроможності,платоспроможності споживачів.

Розрахунок витрат на заробітню плату розробникам проєкту,наведено у таблиці В1 додатку В, сукупні витрати для створення проєкту відображаются у відповідному кошторисі – кошторис витрат по таблиці В2 додатку В.

4.3 Обґунтування необхідності розробки

На сьогоднішній день існує багато вірусних програм, які перешкоджають роботі цифрових пристроїв. В більшості випадків під час завантаження програм з невідомих ресурсів, після їх запуску пристрої

виходять з ладу, цим самим це призводить до великих витрат на різні комплектуючі. Найвигіднішим варіантом було б придбати антивірусну програму, яка зменшить затрати на закупівлю різних ресурсів. Купивши програмний засіб це збереже час і гроші. Тому , найголовнішим чинником є забезпечення регулярної та більш зосередженої роботи працівників.

ВИСНОВКИ

Проживаючи у сучасному світі цифрових технологій, на кожному пристрої повинно бути встановлене антивірусне програмне забезпечення. Його відсутність стане проблемою для всіх людей. Саме тому розробка програмного засобу для виявлення шкідливого програмного коду є затребуваним проєктом на часі.

У ході проєктування було вирішено використовувати таку мову програмування, як C++, відповідно програмний код розроблявся в інтегрованому програмному середовищі Embarcadero RAD Studio, це дає змогу максимально швидко створити доступний графічний інтерфейс користувача.

Для виконання мети проєкту програмного засобу користувачеві стає доступний такий функціонал:

– надання інформації про систему;
– перевірка файлу на наявність вірусу;
– сканування всієї системи.

Таким чином в результаті розробки програмного засобу є готовий продукт, який виконує спроєктовані в процесі розробки функції з можливістю майбутньої модернізації та розширення функціоналу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. С++ в сучасному світі. *Habr*: веб сайт. URL:https://habr.com/ru/ company /pvs-studio/blog/259777/ (дата звернення: 23.05.2022);
2. Бібліотека хешів. *Portable C++ Hashing Library*: веб сайт URL: https://create.stephan-brumme.com/hash-library/ (дата звернення: 23.05.2022);
3. Графічний інтерфейс користувача. *C++ Builder*: веб сайт URL: https://uk.wikipedia.org/wiki/C%2B%2B_Builder (дата звернення: 23.05.2022);
4. Інтегроване середовище розробки. *Embarcadero RAD Studio*: веб сайт URL: https://uk.wikipedia.org/wiki/ Embarcadero_RAD_Studio (дата звернення: 23.05.2022);
5. Інтерфейс програмування.*Windows API*: веб сайт URL: https://ru.wikipedia.org/wiki/Windows_API(дата звернення: 23.05.2022).
6. Контрольна сума. *Вікіпедія*. URL: https://en.wikipedia.org/wiki/Checksum (дата звернення: 23.05.2022);
7. Технології виявлення вірусів. *Studfile*. URL: https://studfile.net/preview/5368373/page:3/ (дата звернення: 23.05.2022);
8. Антивірусні програми – захист локальної мережі. *Sites.google*. URL: https://www.sites.google.com/site/zahistlokalnoiemerezi/zahist/antivirus (дата звернення: 23.05.2022).

ДОДАТКИ

Додаток А

Програмний код основної форми

Лістинг А1 – Код для підключення бібліотек

```
#include <vcl.h>
#pragma hdrstop
#include <IdHashMessageDigest.hpp>
#include "Antivirus.h"
#include "Unit1.h"
#include <fstream>
#include "sha256.cpp"
#pragma package(smart_init)
#pragma resource "*.dfm"
```

Лістинг А2 – Кнопка для переходу на іншу форму

```
void __fastcall TForm2::infoButtonClick(TObject *Sender)
{
Form1->ShowModal();
}
```

Лістинг А3 – Код для кнопки OpenDialog

```
void __fastcall TForm2::OpenButtonClick(TObject *Sender)
{
      if    (OpenDialog1->Execute()) {
    Edit1->Text=OpenDialog1->FileName.c_str();
  }
}
```

Лістинг А4 – Код для кнопки «перевірка файлу»

```
void __fastcall TForm2::CheckButtonClick(TObject *Sender)
{
      std::ifstream file;
      std::istream* input = NULL;
      SHA256 digestSha2;
      file.open(OpenDialog1->FileName.c_str(),      std::ios::in      |
std::ios::binary);
      input = &file;
    const size_t BufferSize = 144*7*1024;
    char* buffer = new char[BufferSize];
```

```
  // process file
  while (*input)
  {
            input->read(buffer, BufferSize);
            std::size_t numBytesRead = size_t(input->gcount());
            digestSha2.add(buffer, numBytesRead);
  }
      // clean up
      // AnsiString s=

  Edit1->Text=(digestSha2.getHash().c_str());

  file.close();

  int i;
      TStringList *List = new TStringList;
      List->LoadFromFile("C:\\desktop\\1.txt");
      if (List->IndexOf(Edit1->Text) == -1)
      {
            ShowMessage("No!");
      }
      else
      {
         ShowMessage("Virus!");
      }


   delete[] buffer;
}
```

Лістинг А5 – Код для кнопки Вихід

```
void __fastcall TForm2::ExitButtonClick(TObject *Sender)
{
      Form2->Close();
}
```

Додаток Б

Програмний код другорядної форми

Лістинг Б1 – Код для підключення бібліотек

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma link "perfgrap"
#pragma link "perfgrap"
#pragma link "perfgrap"
#pragma resource "*.dfm"
#define SystemProcessorTimes 8
#define MAX_PROCESSORS 32
```

Лістинг Б2 – Код для створення форми

```
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
```

Лістинг Б3 – Код для оголошення змінних

```
typedef struct _SYSTEM_BASIC_INFORMATION
{
    ULONG Unknown,
    MaximumIncrement,
    PhysicalPageSize,
    NumberOfPhysicalPages,
    LowestPhysicalPage,
    HighestPhysicalPage,
    AllocationGranularity,
    LowestUserAddress,
    HighestUserAddress,
    ActiveProcessors;
    char NumberProcessors;
}SYSTEM_BASIC_INFORMATION, *PSYSTEM_BASIC_INFORMATION;

typedef struct _SYSTEM_PROCESSOR_TIMES
{
    __int64 IdleTime,
    KernelTime,
```

```
            UserTime,
            DpcTime,
            InterruptTime;
            ULONG InterruptCount;
      }    SYSTEM_PROCESSORS_TIMES[MAX_PROCESSORS],    SYSTEM_PROCESSOR_TIMES,
*PSYSTEM_PROCESSOR_TIMES;

      typedef                              UINT                              __stdcall
(*ZwQuerySystemInformation)(DWORD,void*,DWORD,DWORD*);

      /*function*/
      SYSTEM_INFO SYSINFO;

      int GetProcessorsCount(void)
      {
        SYSTEM_INFO SYSINFO;
        GetSystemInfo(&SYSINFO);
        return SYSINFO.dwNumberOfProcessors;
      }

      int GetProcessorType(SYSTEM_INFO& SYSINFO)
      {
      GetSystemInfo(&SYSINFO);
      return SYSINFO.dwProcessorType;
      }

      int GetProcessorOemId(SYSTEM_INFO& SYSINFO)
      {
      GetSystemInfo(&SYSINFO);
      return SYSINFO.dwOemId;
      }


      int * cpu_usage;
      __int64 temp = 0;
      unsigned int cycle_count = 0,
                   count_mas = 0,
                         interval = 1000; //період запросу (мс)

      HMODULE lib = NULL;
      DWORD nowtime, oldtime, pertime, rez, curtime = 0;
      SYSTEM_PROCESSORS_TIMES CurrentSysProcTimes, PreviousSysProcTimes;
```

```
ZwQuerySystemInformation func;

String msg = "";
char * windowname = new char[20];

bool init = false;
```

Лістинг Б4 – Код для кнопки «старт»

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{

   strcpy(windowname,"CPU usage");

    ZeroMemory(&CurrentSysProcTimes[0],sizeof(CurrentSysProcTimes));
    ZeroMemory(&PreviousSysProcTimes[0],sizeof(PreviousSysProcTimes));

      lib = LoadLibrary(L"Ntdll.dll");      //динамічне завантаження
бібліотеки

     if (!lib)
     {
            msg = "Error #1: не вдалося загрузити бібліотеку!";
         Application->MessageBox(msg.c_str(), L"Warning", MB_OK |
MB_ICONERROR);
         return;
     }

      func                                                    =
(ZwQuerySystemInformation)GetProcAddress(lib,"ZwQuerySystemInformation");
//запрос адреси функції

     if (!func)
     {
            msg = "Error #2: Не вдалося получити адресу функції!";
         Application->MessageBox(msg.c_str(), L"Warning", MB_OK |
MB_ICONERROR);
         FreeLibrary(lib);
         return;
     }

      Label1->Caption = "кількість ядер : " +
IntToStr(GetProcessorsCount()) + " / тип процессора : " +
```

```
IntToStr(GetProcessorType(SYSINFO))    +    "    /    OemId    :    "    +
IntToStr(GetProcessorOemId(SYSINFO));
          init = true;
            msg = "Ініціалізація успішна!";
          Application->MessageBox(msg.c_str(),   (String(windowname)).c_str(),
MB_OK | MB_ICONINFORMATION);
          Button1->Enabled = false;


          cpu_usage = new int[100000];
            count_mas = 0;
            Timer1->Enabled = true;
      }
```

Лістинг Б5 – Код для таймера

```
      void __fastcall TForm1::Timer1Timer(TObject *Sender)
      {
                  Timer1->Enabled = true;
                  nowtime = GetTickCount();
                  pertime = nowtime - oldtime;
               curtime += pertime;
               oldtime = nowtime;


               func(SystemProcessorTimes,
                     &CurrentSysProcTimes[0],
                     sizeof(CurrentSysProcTimes),
                     &rez);

               temp = 0;
               for(int j=0;j<MAX_PROCESSORS;j++)
               {
                   temp       +=       CurrentSysProcTimes[j].IdleTime       -
PreviousSysProcTimes[j].IdleTime;
               }
                   temp /= 10000;  //перевод з наносекунд в мілісекунди
                   temp /= (int) GetProcessorsCount(); //ділимо на ксть ядер

               temp = (int) pertime - temp;


               if (cycle_count >= 1)
               {
```

```
            cpu_usage[count_mas] = temp / (float) pertime*100;


            msg = "";
                msg += "Використання " + IntToStr(cpu_usage[count_mas]) +
"%";
                Caption = msg;
            PerformanceGraph1->Scale += 1;


            count_mas++;
            }

                cycle_count++;

            memcpy(&PreviousSysProcTimes[0],
                    &CurrentSysProcTimes[0],
                            sizeof(PreviousSysProcTimes));

    }
```

Лістинг Б6 – Код для графіка

```
void __fastcall TForm1::PerformanceGraph1ScaleChange(TObject *Sender)
{
      PerformanceGraph1->DataPoint(clLime, cpu_usage[count_mas]);
      PerformanceGraph1->Update();
}
```

Додаток В

Таблиці до економічного розділу

Таблиця В1 – Розрахунок витрат на заробітню плату

№ п/п	Посада виконавця	Оклад, грн/міс	Відрахування, грн/міс.	Кількість чол.	Кількість місяців	Сума, з/п, грн.
1	Інженер І категорії	8000	1600	1	5	6400
2	Інженер	8000	1600	1	5	6400
3	Технік	8000	1600	1	5	6400

Таблиця В2 – Сукупні витрати для створення проєкту

Найменування статей витрат	Сумма,грн	Обгрунтування
1.Зарплата проєктувальників	24000	Оптимальний час виконання проєкту сягає 5 місяців при злагодженій роботі трьох спеціалістів. Зм. Арк. № докум. Підпис. Дата 70 Арк. ДП.КН 20.417.01.000 ПЗ Посадові оклади узгодженні із тарифним коефіцієнтом кожної посади
2. Відрахування на соціальні потреби	4800	Сума відрахувань від зарплати на соціальні потреби визначається по встановленому нормативу у відсотках від суми заробітної плати (4723 грн/особа) = 22%.
3.Контрагентські роботи і послуги	-	-

4. Відрядження	-	-
5. Інші прямі витрати	1500	Витратні матеріали та канцелярські товари.
6. Усього прямих витрат	30300	Підсумкова вартість витрат пункту 1-5.
7. Планові накопичення	4500	Оплата за опалення, електроенергію та інші супутні послуги протягом 5 місяців роботи над проєктом.
8. Планові накопичення	6060	20% від суми прямих і накладних витрат, яка спрямована на преміювання виконавчих осіб проєкту.
9.Усього, кошторисна вартість проєкту	40860	Загальна вартість прямих і накладних витрат та планових накопичень.
10. Податок на додану вартість	8172	Сума, яка сягає 20% від кошторисної вартості проєкту (визначається по діючому нормативу).
11. Загалом, договірна ціна розробки ЗП.	49032	Вказує на суму кошторисної вартості роботи та податку на додану вартість і визначає ті витрати на проєктування, що несе підприємство-замовник.