Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм”

Виконав здобувач вищої освіти
II курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Краєвський М.О.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Краєвському Миколі Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Краєвський М.О. Дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи реалізації інструментальних засобів організації пакетів прикладних програм.

Метою розробки є дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

Об'єктом дослідження є процес реалізації інструментальних засобів організації пакетів прикладних програм.

Предметом дослідження є методи реалізації інструментальних засобів організації пакетів прикладних програм.

Методи дослідження базуються на методах інженерії програмного забезпечення, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.

Ключові слова: комп'ютерні науки, прикладні програми

ABSTRACT

Kraievskyi M.O. Research and software implementation of the implementation system of tools for organizing application program packages. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the implementation system of tools for organizing application program packages.

The goal of the development is the research and software implementation of the implementation system of tools for the organization of application program packages.

The object of the study is the process of implementing instrumental means of organizing application program packages.

The subject of the study is methods of implementing instrumental means of organizing application program packages.

Research methods are based on software engineering methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the implementation system of tools for the organization of application program packages.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10 environment.

Keywords: computer science, applied programs

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	програмне забезпечення
ППП	–	пакети прикладних програм
РОС	–	розподілені обчислювальні середовища
РППП	–	розподілені пакети прикладних програм
СРЗ	–	схема рішення завдання
СУРОС	–	система управління розподіленим обчислювальним середовищем

ВСТУП

Актуальність теми. Сучасні пакети прикладних програм (ППП) являють собою складні програмні комплекси, що включають наступні основні компоненти:

– функціональне (прикладне) наповнення;

– високорівневі язикові засоби опису досліджуваної предметної області;

– системне програмне забезпечення (ПЗ) для організації процесу рішення дослідницького завдання.

Сполучення в ППП різноманітних складних моделей, алгоритмів і методик їхнього дослідження базується, як правило, на використанні принципу модульної організації функціонального наповнення пакета. Модуль оформляється у вигляді програмної одиниці мовою програмування високого рівня (наприклад, у вигляді програми, що виконується, або підпрограми-функції), що забезпечує рішення окремого завдання.

Інтенсивний розвиток мережних технологій і апаратних засобів, спостережуване в останні роки, дозволило багаторазово підвищити продуктивність сучасних обчислювальних систем і забезпечило можливість організації ефективних паралельних обчислень. Природно виникла необхідність створення ППП, здатних максимально використовувати потенціал високопродуктивних обчислювальних систем.

Якісні вимоги до реалізації процесу паралельної обробки даних у прикладних програмах (наприклад, необхідність забезпечення ефективності, масштабованості, переносимості й т.д.) породили велике різноманіття систем для організації паралельних обчислень. Такі системи, як правило, жадають від фахівця-предметника досить високого рівня програмістської кваліфікації й навичок розробки паралельних програм. У процесі проектування й розробки паралельних програм фахівцеві-предметнику доводиться володіти тими або

іншими технологіями і моделями паралельного програмування, а також ураховувати архітектуру й особливості використовуваної обчислювальної системи.

Тому одним з найбільш перспективних і актуальних напрямків розвитку пакетної проблематики є створення інструментальних засобів, що забезпечують розробку й застосування ППП для паралельних і розподілених обчислювальних систем, у тому числі для розподілених обчислювальних середовищ (РОС). РОС розглядається в традиційному розумінні (див., наприклад, роботи В.Н. Коваленко, Д.А. Корягіна й ін.) як сукупність обчислювальних вузлів, об'єднаних комунікаційною мережею. Обчислювальний вузол являє собою програмно-апаратний ресурс, що включає:

– модуль оперативної пам'яті;

– один або кілька процесорів;

– жорсткий диск;

– системне й прикладне програмне забезпечення, що підтримує функціонування вузла у РОС.

Один з обчислювальних вузлів призначається головним вузлом і наділяється функціями управління завданнями користувачів і ресурсами РОС. Більшість відомих на сьогоднішній день інструментальних систем не вирішують всіх проблем, пов'язаних з розробкою й застосуванням таких прикладних програмних комплексів. Уведемо поняття успадкованого ПЗ. Під успадкованим ПЗ (legacy systems) розуміються програмні системи або комплекси, що не відповідають сучасним вимогам, але дотепер використовувані через значні фінансові, організаційні, технічні й інші утруднення, пов'язаних з їхньою заміною (див., наприклад, роботи Л.В. Массель і ін.).

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем реалізації інструментальних засобів організації пакетів прикладних програм.

– Дослідження системи реалізації інструментальних засобів організації пакетів прикладних програм.

– Програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

Об'єктом дослідження є процес реалізації інструментальних засобів організації пакетів прикладних програм.

Предметом дослідження є методи реалізації інструментальних засобів організації пакетів прикладних програм.

Методи дослідження базуються на методах інженерії програмного забезпечення, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод реалізації інструментальних засобів організації пакетів прикладних програм.

– Розроблено вітчизняний продукт реалізації інструментальних засобів організації пакетів прикладних програм, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі реалізації інструментальних засобів організації пакетів прикладних програм.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації програмного забезпечення інструментальних засобів організації пакетів прикладних програм (ППП). Класифікація і типові представники прикладного програмного забезпечення включають в себе:

– Проблемно-орієнтовані ППП.

– ППП автоматизованого проектування.

– ППП загального призначення.

– Методо-орієнтовні ППП.

– Офісні ППП.

– Настільні видавничі системи.

– Програмні засоби мультимедіа.

– Системи штучного інтелекту.

1.2 Область застосування

Областю застосування системи є пакети прикладних програм (ППП).

Проблемно-орієнтовані ППП

Це самий представницький клас програмних продуктів, усередині якого проводиться класифікація по різних ознаках:

– типам предметних областей;

– інформаційним системам;

– функціям і комплексам задач, реалізованих програмним способом, і ін.

Пакети прикладних програм по функціональному призначенню на ринку програмних продуктів:

– ППП автоматизованого бухгалтерського обліку;

– ППП фінансової діяльності;

– ППП керування персоналом (кадровий облік);

– ППП керування матеріальними запасами;

– ППП керування виробництвом;

– банківські інформаційні системи і т.п.

Основні тенденції в області розвитку проблемно-орієнтованих програмних засобів:

– створення програмних комплексів у виді автоматизованих робочих місць (АРМ) управлінського персоналу;

– створення інтегрованих систем керування предметною областю на базі обчислювальних мереж, що поєднують АРМи в єдиний програмний комплекс з архітектурою клієнт-сервер;

– організація даних великих інформаційних систем у виді розподіленої бази даних на мережі ЕОМ;

– наявність простих мовних засобів кінцевого користувача для запитів до бази даних;

– налаштування функцій обробки силами кінцевих користувачів (без участі програмістів);

– захист програм і даних від несанкціонованого доступу (парольний захист на рівні функцій, режимів роботи, даних).

Для подібного класу програм високі вимоги до оперативності обробки даних (наприклад, пропускна здатність для банківських систем повинна складати декілька сот транзакцій у секунду), великі обсяги збереженої інформації, що обумовлює підвищені вимоги до засобів адміністрування даних БД (актуалізації, копіювання, забезпечення продуктивності обробки даних). Найбільше важливо для даного класу програмних продуктів створення дружнього інтерфейсу для кінцевих користувачів. Даний клас програмних продуктів дуже динамічний як по

складу реалізованих ними функцій, так і по використовуваному для їхнього створення інструментарію розроблювача.

ППП автоматизованого проектування

Програми цього класу призначені для підтримки роботи конструкторів і технологів, зв'язаних з розробкою креслень, схем, діаграм, графічним моделюванням і конструюванням, створенням бібліотеки стандартних елементів (темплетів) креслень і їхнім багаторазовим використанням, створенням демонстраційних ілюстрацій і мультфільмів.

Відмінною рисою цього класу програмних продуктів є високі вимоги до технічної частини системи обробки даних, наявність бібліотек убудованих функцій, об'єктів, інтерфейсів із графічними системами і базами даних.

ППП загального призначення

Даний клас містить широкий перелік програмних продуктів, що підтримують переважно інформаційні технології кінцевих користувачів. Крім кінцевих користувачів цими програмними продуктами за рахунок убудованих засобів технології програмування можуть користатися і програмісти для створення ускладнених програм обробки даних.

Представники даного класу програмних продуктів.

1.Настільні системи керування базами даних (СУБД), що забезпечують організацію і збереження локальних баз даних на автономно працюючих комп'ютерах або централізоване збереження баз даних на файлі-сервері і мережний доступ до них. В даний час найбільш широке представлені реляційні СУБД для персональних комп'ютерів, що здійснюють:

– роботу з базою даних через екранні форми;

– організацію запитів на пошук даних за допомогою спеціальних мов запитів високого рівня;

– генерацію звітів різної структури даних з підведенням проміжних і остаточних підсумків;

– обчислювальну обробку шляхом виконання убудованих функцій, програм, написаних з використанням мов програмування і макрокоманд.

У сучасних СУБД (наприклад, у СУБД Access містяться елементи CASE-технології процесу проектування, зокрема:

– візуалізована схема баз даних;

– здійснено автоматичну підтримку цілісності баз даних при різних видах обробки (включення, чи видалення модифікація даних баз даних);

– надаються так називані майстри, що забезпечують підтримку процесу проектування (режим "конструктор") – майстер таблиць, майстер форм, майстер звітів, налаштовувач меню і т.п.;

– створені для широкого використання шаблони структур баз даних, форм, звітів і т.д.

2.Сервери баз даних – успішно розвивається вид програмного забезпечення, призначений для створення і використання при роботі в мережі інтегрованих баз даних в архітектурі клієнт-сервер.

Багатокористувальницькі СУБД (типу Paradox, Access, FoxPro і ін.) у мережному варіанті обробки даних зберігають інформацію на файлі-сервері – спеціально виділеному комп'ютері в централізованому виді, але сама обробка даних ведеться на робочих станціях, сервери баз даних, навпроти, всю обробку (збереження, пошук, витяг і передачу даних клієнту)виконують самостійно, одночасно забезпечуючи даними велике число користувачів мережі.

Загальним для різних видів серверів баз даних є використання реляційної мови SQL (Structured Query Language) для реалізації запитів до даних.

Найбільшою проблемою застосування серверів баз даних є забезпечення цілісності (несуперечності) баз даних, рішення питання, зв'язаного з дублюванням (тиражуванням) даних по вузлах мережі і їх синхронним відновленням.

3. Генератори (сервери) звітів – самостійний напрямок розвитку програмних засобів, що забезпечують реалізацію запитів і формування звітів у друкованому чи екранному виді в умовах мережі з архітектурою клієнт-сервер.

Сервери звітів включають:

– програми планування – облік часу для формування звітів за вимогою користувачів, складання розкладу видачі і поширення звітів по мережі;

– програми керування чергою запитів на формування звітів;

– програми ведення словника користувачів для розмежування доступу до сформованих звітів;

– програми ведення архіву звітів і ін.

Підготовлені звіти розсилаються клієнтам по електронній чи пошті за допомогою іншого транспортного агента. Сервери звітів звичайно підтримують різнорідні платформи, тим самим вони ефективно працюють у неоднорідних обчислювальних мережах.

4.Текстові процесори – автоматичне форматування документів, вставка мальованих об'єктів і графіки, складання змістів і покажчиків, перевірка орфографії, шрифтове оформлення, підготовка шаблонів документів.

Розвитком даного напрямку програмних продуктів є видавничі системи.

5.Табличний процесор – зручне середовище для обчислень силами кінцевого користувача; засобу ділової графіки, спеціалізований обробка (убудовані функції, робота з базами даних, статистична обробка даних і ін.).

6.Засоби презентаційної графіки – спеціалізовані програми, призначені для створення зображень і їхнього показу на екрані, підготовки слайдів-фільмів, мультфільмів, відеофільмів, їхнього редагування, визначення порядку проходження зображень.

Презентація може включати показ діаграм і графіків, усі програми презентаційної графіки умовно поділяються на програми для підготовки шоу, програми для підготовки мультимедіа-презентації. Для роботи цих програм необхідна також наявність спеціалізованого устаткування – рідкокристалічній

проекційної панелі, що просвічується проектором для висновку зображення на екран, відеотехніка.

Презентація вимагає попереднього складання плану показу. Для кожного слайда виконується проектування: визначаються зміст слайда, розмір, склад елементів, способи їхнього оформлення і т.п. Дані для використання в слайдах можна як готувати вручну, так і одержувати в результаті обміну з інших програмних систем.

7. Інтегровані пакети – набір декількох програмних продуктів, що функціонально доповнюють один одного, що підтримують єдині інформаційні технології, реалізовані на загальній обчислювальній і операційній платформі. Найбільш поширені інтегровані пакети, компонентами яких є:

– СУБД;

– текстовий редактор;

– табличний процесор;

– органайзер;

– засобу підтримки електронної пошти;

– програми створення презентацій;

– графічний редактор.

Компоненти інтегрованих пакетів можуть працювати ізольовано друг від друга, але основні достоїнства інтегрованих пакетів виявляються при їхньому розумному сполученні один з одним. Користувачі інтегрованих пакетів мають уніфікований для різних компонентів інтерфейс, тим самим забезпечується відносна легкість процесу їхнього освоєння.

Інтегровані пакети ефективні і при груповій роботі в мережі багатьох користувачів. Так, із прикладної програми, у якій знаходиться користувач, можна відправити документи і файли даних іншому користувачу, при цьому підтримуються стандарти передачі даних у виді об'єктів по чи мережі через електронну пошту.

Методоорієнтовні ППП

Даний клас включає програмні продукти, що забезпечують незалежно від предметної області і функцій інформаційних систем математичні, статистичні й інші методи рішення задач.

Найбільш поширені методи математичного програмування, рішення диференціальних рівнянь, імітаційного моделювання, дослідження операцій. Методи статистичної обробки й аналізу даних (описова статистика, регресійний аналіз, прогнозування значень техніко-економічних показників і т.п.) мають усезростаюче застосування.

Так, сучасні табличні процесори, значно розширили набір убудованих функцій, що реалізують статистичну обробку, пропонують інформаційні технології статистичного аналізу. Разом з тим необхідність у використанні спеціалізованих програмних засобів статистичної обробки, що забезпечують високу точність і різноманіття статистичних методів, також росте.

Офісні ППП

Даний клас програмних продуктів охоплює програми, що забезпечують організаційне керування діяльністю офісу:

1. Органайзери (планувальники) – програмне забезпечення для планування робочого часу, складання протоколів зустрічей, розкладів, ведення записної і телефонної книжки.

До складу програм органайзерів входять: калькулятор, записна книжка, годинник, календар і т.п. Найбільше часто подібне програмне забезпечення розробляється для ноутбуків, персональних комп'ютерів блокнотного типу.

2.Програми-перекладачі, засоби перевірки орфографії і розпізнавання тексту включають:

– програми-перекладачі, призначені для створення дослівника вихідного тексту зазначеною мовою;

– словники орфографії, використовувані при перевірці текстів;

– словники синонімів, використовувані для стильового виправлення текстів;

– програми для розпізнавання ліченої сканерами інформації і перетворення в текстове представлення.

3.Комунікаційні ППП – призначені для організації взаємодії користувача з вилученими чи абонентами інформаційними ресурсами мережі.

Електронна пошта також стає обов'язковим компонентом офісних ППП. Вона повинна забезпечувати шифрування переданої інформації, факсиміле підпису, перевірку орфографії на кожній з мов, керування повідомленнями по електронній пошті (оповіщення про нову пошту, організація поштових скриньок, пошук, цитування кореспонденції і т.д.).

4. Настільні видавничі системи

Даний клас програм включає програми, що забезпечують інформаційну технологію комп'ютерної видавничої діяльності:

– форматування і редагування текстів;

– автоматичну розбивку тексту на сторінки;

– створення заголовків;

– комп'ютерну верстку друкованої сторінки;

– монтування графіки;

– підготовку ілюстрації і т.п.

Програмні засоби мультимедіа

Цей клас програмних продуктів є відносно новим, він сформувався в зв'язку зі зміною середовища обробки даних, появою лазерних дисків високої щільності запису з гарними технічними параметрами за доступними цінами, розширенням складу периферійного устаткування, що підключається до персонального комп'ютера, розвитком мережної технології обробки, появою регіональних і глобальних інформаційних мереж, що розташовують могутніми інформаційними ресурсами.

Основне призначення програмних продуктів мультимедіа – створення і використання аудіо- і відеоінформації для розширення інформаційного простору користувача. Програмні продукти мультимедіа зайняли лідируюче положення на ринку в сфері бібліотечного інформаційного обслуговування, процесі навчання, організації дозвілля. Бази даних комп'ютерних зображень творів мистецтва, бібліотеки звукових записів і будуть складати основу для прикладних навчальних систем, комп'ютерних ігор, бібліотечних каталогів і фондів.

Системи штучного інтелекту

Даний клас програмних продуктів реалізує окремі функції інтелекту людини. Основними компонентами систем штучного інтелекту є база знань, інтелектуальний інтерфейс із користувачем і програма формування логічних висновків. Їхня розробка йде по наступним напрямках:

– програми-оболонки для створення експертних систем шляхом наповнення баз знань і правил логічного висновку;

– готові експертні системи для прийняття рішень у рамках визначених предметних областей;

– системи керування базами знань для підтримки семантичних моделей (процедуральної, семантичної мережі, фреймової, продукційної і ін.);

– системи аналізу і розпізнавання мови й ін.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Етапи розвитку ППП

Перші пакети прикладних програм являли собою прості тематичні добірки програм для рішення окремих завдань у тої або іншій предметній області, звертання до них виконувалося за допомогою засобів оболонки ОС або з інших програм. Сучасний пакет є складною програмною системою, що включає спеціалізовані системні і язикові засоби. У відносно короткій історії розвитку обчислювальних ППП можна виділити 4 основні покоління (класу) пакетів. Кожний із цих класів характеризується певними особливостями вхідний состав ППП компонентів – вхідних мов, предметного й системного забезпечення.

Перше покоління

Як вхідні мови ППП першого покоління використовувалися універсальні мови програмування (Фортран, Алгол-60 і т.п.) або мови керування завданнями відповідних операційних систем. Проблемна орієнтація вхідних мов досягалася за рахунок відповідної мнемоніки в ідентифікаторах. Складання завдань на такій мові практично не відрізнялося від написання програм алгоритмічною мовою. Предметне забезпечення перших ППП, як правило, було організовано у формі бібліотек програм, тобто у вигляді наборів (пакетів) незалежних програм на деякій базовій мові програмування (звідси вперше виник і сам термін «пакет»). Такі ППП іноді називають пакетами бібліотечного типу, або пакетами простої структури.

Як системне забезпечення пакетів першого покоління звичайно використовувалися штатні компоненти програмного забезпечення ЕОМ:

компілятори з алгоритмічних мов, редактори текстів, засобу організації бібліотек програм, архівні системи й т.д. Ці пакети не вимагали скільки-небудь розвитий системної підтримки, і для їхнього функціонування цілком вистачало зазначених системних засобів загального призначення. У більшості випадків розроблювачами таких пакетів були прикладні програмісти, які намагалися пристосувати універсальні мови програмування до своїх потреб.

Друге покоління

Розробка ППП другого покоління здійснювалася вже за участю системних програмістів. Це привело до появи спеціалізованих вхідних мов на базі універсальних мов програмування. Проблемна орієнтація таких мов досягалася не тільки за рахунок використання певної мнемоніки, але також застосуванням відповідних язикових конструкцій, які спрощували формулювання завдання й робили її більше наочної. Транслятор з такої мови являв собою препроцесор (найчастіше макропроцесор) до транслятора відповідної алгоритмічної мови. Як модулі в пакетах цього класу стали використовуватися не тільки програмні одиниці (тобто закінчені програми на тій або іншій мові програмування), але й такі об'єкти, як послідовність операторів мови програмування, сукупність даних, схема рахунку й ін.

Істотні зміни перетерпіли також принципи організації системного забезпечення ППП. У досить розвинених пакетах другого покоління вже можна виділити елементи системного забезпечення, характерні для сучасних пакетів: монітор, транслятори із вхідних мов, спеціалізовані банки даних, засобу опису моделі предметної області й планування обчислень і ін.

Третє покоління

Третій етап розвитку ППП характеризується появою самостійних вхідних мов, орієнтованих на користувачів-непрограмістів. Особлива увага в таких ППП приділяється системним компонентам які забезпечували простоту й зручність. Це досягається головним чином за рахунок спеціалізації вхідних мов і включення до складу пакета засобів автоматизованого планування обчислень.

Інтероперабельність ППП

Розширення сфери застосування обчислювальної техніки привело до появи різноманітних цифрових пристроїв (від настільних систем і нетбуків, до смартфонів і побутової техніки) на різних апаратно-програмних платформах. Ця ситуація, з одного боку, загострила питання, що виникло ще в розроблювачів обчислювальних систем першого-другого поколінь: як забезпечити роботу програм при зміні платформи? А з іншого боку – створила проблему для кінцевих користувачів: як використовувати необхідні їм ППП, створені для іншої системи й при цьому не міняти платформу? Тобто як, наприклад, змусити працювати додатка для андроід під керуванням iOS або як запустити MS Office під Linux'ом.

Рішенням проблеми забезпечення інтероперабельності (перенесення) ПЗ є концепція відкритих систем. Така система розробляється з використанням відкритих стандартів і специфікацій. Ця концепція всі частіше використовується розроблювачами прикладного програмного забезпечення й істотно полегшує життя користувачам. Прикладами такого ПЗ є офісний пакет LibreOffice, веб-браузер Firefox і всілякі веб-сервіси.

Четверте покоління

Четвертий етап характеризується створенням ППП, експлуатованих в інтерактивному режимі роботи. Основною перевагою діалогової взаємодії з ЕОМ є можливість активного зворотного зв'язка з користувачем у процесі постановки завдання, її рішення й аналізу отриманих результатів. Поява й інтенсивний розвиток різних форм діалогового спілкування обумовлено насамперед прогресом в області технічних засобів (графічна підсистема ЕОМ і засобу мультимедіа, мережні засоби).

Розвиток апаратного забезпечення спричинило створення різноманітних програмних засобів підтримки діалогового режиму роботи (діалогові операційні системи, діалогові пакети програм різного призначення й т.д.). Прикладна система складається з діалогового монітора – набору універсальних програм, що забезпечують ведення діалогу й обмін даними, і бази знань про предметну

область. Інформація про структуру, цілях і форма діалогу задає сценарій, відповідно до якого монітор управляє ходом діалогу. Носіями процедурних знань про предметну область є прикладні модулі, що реалізують функції властиво системи.

Таким чином, створення прикладної системи зводиться до налаштування діалогового монітора на конкретний діалог, шляхом заповнення бази знань. При цьому програмувати в традиційному змісті цього слова доводиться лише прикладні модулі, знання про діалог уводяться в систему за допомогою набору відповідних засобів – редактора сценаріїв. Логічно вимагати, щоб редактор сценаріїв також являв собою діалогову програму, що відповідала розглянутим вище вимогам. Завдяки готовому універсальному монітору програміст може зосередитися на рішенні чисто прикладних завдань, виділення ж знань про діалог у сценарій забезпечує в значній мірі необхідна гнучкість програмного продукту.

Велика увага в теперішній час приділяється проблемі створення «інтелектуальних ППП». Такий пакет дозволяє кінцевому користувачеві лише сформулювати своє завдання в змістовних термінах, не вказуючи алгоритму її рішення. Синтез рішення й складання цільової програми виробляються автоматично. При цьому деталі обчислень сховані від користувача, і комп'ютер стає інтелектуальним партнером людини, здатним розуміти його завдання. Предметне забезпечення подібного ППП являє собою деяку базу знань, що містить як процедурні, так і описові знання. Такий спосіб рішення іноді називають концептуальним програмуванням, характерними рисами якого є програмування в термінах предметної області використання ЕОМ уже на етапі постановки завдань, автоматичний синтез програм рішення завдання, нагромадження знань про розв'язувані завдання в базі знань.

Приклади сучасних прикладних пакетів

Для ілюстрації приведемо кілька прикладів сучасних пакетів прикладних програм з різних предметних областей. З огляду на те, що постійно з'являються нові версії програмних продуктів, тут будуть розглядатися не можливості

конкретних версій, а лише основні структурні компоненти, що входять до складу того або іншого пакета.

Autodesk AutoCAD

Основне призначення системи автоматизованого проектування Autodesk AutoCAD – створення креслень і проектної документації. Сучасні версії цього пакета представляють істотно більші можливості, серед яких побудова тривимірних твердотільних моделей, інженерно-технічні розрахунки й багато чого-багато чого іншого.

Перші версії системи AutoCAD, розроблювальною американською фірмою Autodesk, з'явилися ще на початку 80-х років двадцятого століття, і відразу ж залучили до себе увагу своїм оригінальним оформленням і зручністю для користувача. Постійний розвиток системи, облік зауважень, інтеграція з новими продуктами інших провідних фірм зробили AutoCAD світовим лідером на ринку програмного забезпечення для автоматизованого проектування.

Язикові засоби

В основі язикових засобів ППП AutoCAD – технологія Visual LISP, що базується мовою AutoLISP (підмножина мови LISP) і використовувана для створення додатків і керування в AutoCAD. Visual LISP представляє повне оточення, що включає:

– Інтегроване середовище розробки, що полегшує написання, налагодження й супровід додатків на AutoLISP.

– Доступ до об'єктів Active і оброблювачам подій.

– Захист вихідного коду.

– Доступ до файлових функцій операційної системи.

– Розширені функції мови LISP для обробки облікових структур даних.

Для розроблювачів сумісних додатків в AutoCAD включена підтримка ObjectARX. Це програмне оточення представляє об’єктно-орієнтований інтерфейс для додатків на мовах C++, C# і VB.NET і забезпечує прямий доступ до структур БД, графічній підсистемі й убудованим командам пакета.

Крім того, в AutoCAD є підтримка мови Visual Basic for Applications (VBA), що дозволяє використовувати цей пакет разом з іншими додатками, зокрема, із сімейства Microsoft Office.

Предметне забезпечення

До предметного забезпечення пакета в першу чергу відносяться функції побудови примітивів – різних елементів креслення. Прості примітиви це такі об'єкти як крапка, відрізок, коло (окружність) і т.д. До складних примітивів відносяться: полілінія, мультилінія, мультитекст, розмір, винесення, допуск, штрихування, входження блоку або зовнішнього посилання, атрибут, растрове зображення. Крім того, є просторові примітиви, видові екрани та ін.. Операції побудови більшої частини примітивів можуть бути виконані через користувальницький інтерфейс, усе – через команди мови.

Високорівневі засоби представлені розширеннями й додатками AutoCAD для конкретних предметних областей. Наприклад у машинобудуванні використовується Autodesk Mechanical Desktop – призначений для складного тривимірного моделювання, у тому числі валів і пружин. Для проектування деталей з листових матеріалів призначена система Copra Sheet Metal Bender Desktop (розроблювач – Data-M Software Gmb). Моделювання динаміки роботи механізмів може виконуватися в системі Dynamic Designer (Mechanical Dynamics). У числі відомих архітектурних і будівельних додатків можна відзначити системи АРКО (Апио-Центр), СПДС GraphiCS (Consistent Software), ArchiCAD. Для проектування промислових об'єктів може використовуватися система PLANT-4D (CEA Technology). Це лише деякі з областей використання AutoCAD.

Системне забезпечення

Серед системного забезпечення слід зазначити основний формат файлів AutoCAD .dwg, що став стандартом «де факто» для інших САПР.

До системного ж забезпечення забезпеченню відносяться типові й спеціалізовані бібліотеки деталей і шаблонів, використання яких дозволяє істотно

прискорити процес проектування. Тут же згадаємо вимоги галузевих і державних стандартів, яким повинні відповідати креслення й специфікації.

Конфігурація й налаштування різних режимів AutoCAD установлюються через т.зв. системні змінні. Змінюючи їхнього значення можна задавати шляхи до файлів, точність обчислень, формат виводу й багато чого іншого.

Adobe Flash

Adobe (раніше Macromedia) Flash – це технологія й інструментарій розробки інтерактивного змісту з більшими функціональними можливостями для цифрових, веб- і мобільних платформ. Вона дозволяє створювати компактні, масштабовані анімовані додатки (ролики), які можна використовувати як окремо, так і вбудовуючи в різне оточення (зокрема, у веб-сторінки). Ці можливості забезпечуються наступними компонентами технології: мовою Action Script, векторним форматом .swf і відеоформатом .flv, усілякими flash-плейерами для перегляду й редакторами для створення.

Розглянемо інтегроване середовище Adobe Flash як основний засіб створення flash-додатків. При цьому відзначимо, що язикові й системні засоби відносяться не тільки до цього пакета, а до технології в цілому. Якщо, наприклад, купити фотошоп cs3 у відповідній конфігурації, то ці засоби будуть доступні для всіх додатків із состава пакета.

Мова ActionScript

ActionScript – об’єктно-орієнтована мова програмування, що додає інтерактивність, обробку даних і багато чого іншого у вміст Flash-додатків. Синтаксис ActionScript заснований на специфікації ECMAScript (сюди ж відносяться мови JavaScript і JScript). Бібліотека класів ActionScript, написана на C++, представляє доступ до графічних примітивів, фільтрів, принтерів, геометричних функцій та ін..

ActionScript як мова з'явилася з виходом 5 версії Adobe (тоді ще Macromedia) Flash, що стала першої програмувальної на ActionScript середовищем. Перший реліз мови називався ActionScript 1.0. Flash 6 (MX). В 2004

році Macromedia представила нову версію ActionScript 2.0 разом з виходом Flash 7 (MX 2004), у якій було уведено строге визначення типів, засноване на класах програмування: спадкування, інтерфейси й т.д. Також Macromedia була випущена модифікація мови Flash Lite для програмування під мобільні телефони. ActionScript 2.0 є не більш ніж надбудовою над ActionScript 1.0, тобто на етапі компіляції ActionScript 2.0 здійснює якусь перевірку й перетворює класи, методи ActionScript 2.0 у колишні прототипи й функції ActionScript 1.0.

В 2005 році вийшов ActionScript 3.0 у середовищі програмування Adobe Flex, а пізніше в Adobe Flash 9.

ActionScript 3.0 представляє, у порівнянні з ActionScript 2.0 якісна зміна, вона використовує нову віртуальну машину AVM 2.0 і дає замість колишнього формального синтаксису класів справжнє класове (class-based) об'єктно-орієнтоване програмування. ActionScript 3.0 по швидкості наблизився до таких мов програмування, як Java і C++.

За допомогою ActionScript можна створювати інтерактивні мультимедіа-додатки, ігри, веб-сайти й багато чого іншого.

Системне забезпечення

ActionScript виконується віртуальною машиною (ActionScript Virtual Machine), що є складовою частиною Flash Player. ActionScript компілюється в байткод, що включається в SWF-файл.

SWF-файли виконуються Flash Player-ом. Flash Player існує у вигляді плагіна до веб-браузера, а також як самостійне додаток, що виконується. У другому випадку можливе створення exe-файлів, що виконуються коли swf-файл включається в Flash Player.

Для створення й перегляду відеофайлів у форматі .flv використовуються програмні кодеки, що підтримують цей формат.

Прикладне забезпечення

До прикладного забезпечення в рамках технології Flash відносяться засоби створення роликів у форматах .swf, .flv і .exe. Основним інструментом є

середовище середовище Adode Flash, що включає різні засоби для створення й редагування мультимедійного змісту, у т.ч. відео- і аудіофайлів, інтегроване середовище розробки на ActionScript і безліч додаткових функцій спрощення процесу створення роликів.

Пакет MatLab

MatLab (скорочення від англ. «Matrix Laboratory») – пакет прикладних програм для рішення завдань технічних обчислень, і мова програмування, використовуваний у цьому пакеті. За даними фірми-розроблювача, більше 1000000 інженерних і науковців використовують цей пакет, що працює на більшості сучасних операційних систем, включаючи GNU/Linux, Mac OS, Solaris і Microsoft Windows.

Мова MatLab

MATLAB як мова програмування була розроблена Кливом Моулером (англ. Cleve Moler) наприкінці 1970-х років. Метою розробки служило завдання використання програмних математичних бібліотек Linpack і EISPACK без необхідності вивчення мови Фортран. Акцент був зроблений на матричні алгоритми.

Програми, написані на MATLAB, бувають двох типів – функції й скріпти. Функції мають вхідні й вихідні аргументи, а також власний робочий простір для зберігання проміжних результатів обчислень і змінних. Скріпти же використовують загальний робочий простір. Як скріпти, так і функції не компілюються в машинний код, а зберігаються у вигляді текстових файлів. Існує також можливість зберігати так звані pre-parsed програми – функції й скріпти, наведені у вид, зручний для машинного виконання й, як наслідок, більше швидкі в порівнянні зі звичайними.

Системне забезпечення

Мова MATLAB є високорівневою интерпретуємою мовою програмування, що включає засновані на матрицях структури даних, широкий спектр функцій, інтегроване середовище розробки, об’єктно-орієнтовані можливості й інтерфейси

до програм, написаних на інших мовах програмування. Є інтерфейси для одержання доступу до зовнішнім даних, клієнтам і серверам, що спілкуються через технології Component Object Model (COM) або Dynamic Data Exchange (DDE), а також периферійним пристроям, які взаємодіють прямо з MATLAB. Багато хто із цих можливостей відомі за назвою MATLAB API.

Убудоване середовище розробки дозволяє створювати графічні інтерфейси користувача з різними елементами керування, такими як кнопки, поля уведення й іншими. За допомогою компонента MATLAB Compiler ці графічні інтерфейси можуть бути перетворені в самостійні додатки.

Для MATLAB є можливість створювати спеціальні набори інструментів (англ. toolbox), що розширюють його функціональність. Набори інструментів являють собою колекції функцій, написаних мовою MATLAB для рішення певного класу завдань.

Прикладне забезпечення

MATLAB надає зручні засоби для розробки алгоритмів, включаючи високорівневі з використанням концепцій об’єктно-орієнтованого програмування. У ньому є всі необхідні засоби інтегрованого середовища розробки, включаючи відладник і профайлер.

MATLAB надає користувачеві велика кількість (кілька сотень) функцій для аналізу даних, покриваючі практично всі області математики, зокрема:

– Матриці й лінійна алгебра – алгебра матриць, лінійні рівняння, власні значення й вектора, сінгулярності, факторизація матриць і інші.

– Багаточлени й інтерполяція – корінь багаточленів, операції над багаточленами і їхнім диференціюванням, інтерполяція й екстраполяція кривих і інші.

– Математична статистика й аналіз даних – статистичні функції, статистична регресія, цифрова фільтрація, швидке перетворення Фур'є й інші.

– Обробка даних – набір спеціальних функцій, включаючи побудову графіків, оптимізацію, пошук нулів, чисельне інтегрування (у квадратурах) і інші.

– Диференціальні рівняння – рішення диференціальних і диференційно-алгебраїчних рівнянь, диференціальних рівнянь із запізнюванням, рівнянь із обмеженнями, рівнянь у частинних похідних і інші.

– Розріджені матриці – спеціальний клас даних пакета MATLAB, що використовується в спеціалізованих додатках.

У складі пакета є велика кількість функцій для побудови графіків, у тому числі тривимірних, візуального аналізу даних і створення анімованих роликів, функції для створення алгоритмів для мікроконтролерів і інших додатків.

OpenDocument Format

OpenDocument Format (ODF, скорочене від OASIS Open Document Format for Office Application – відкритий формат документів для офісних додатків) – відкритий формат файлів документів для зберігання й обміну офісними документами, що редагуються: текстовими документами, електронними таблицями, малюнками, базами даних, презентаціями.

Стандарт був розроблений індустріальним співтовариством OASIS і заснований на XML-форматі. 1 травня 2006 року прийнятий як міжнародний стандарт ISO/IEC 26300, доступний для всіх і може бути використаний без обмежень. Цей формат підтримується в таких ППП як OpenOffice.org, IBM Lotus Symphony, Koffice, Scribus, Google Docs, AjaxWrite, Microsoft Office 2007 SP2.

Portable Network Graphics

PNG (англ. portable network graphics) – растровий формат зберігання графічної інформації зі стиском без втрат якості. PNG був створений спеціально для використання в Інтернет як альтернатива формату GIF. Цей формат був розроблений на початку 1995 р. по ідеї Т. Боутелла. У жовтні 1996 року специфікація PNG версії 1.0 була рекомендована консорціумом W3C у якості повноправного мережного формату й зараз широко використовується в Мережі.

Формат SVG

SVG (англ. Scalable Vector Graphics – масштабована векторна графіка) – мова розмітки масштабованої векторної графіки, створена консорціумом W3C і

вхідний у підмножину розширюваної мови розмітки XML. SVG призначений для опису двомірної векторної й змішаної векторно/растрової графіки. Формат підтримує як нерухливу, так анімовану й інтерактивну графіку. Оскільки SVG заснований на XML, те він представляє всі переваги розширюваної мови розмітки. Відзначимо основні:

– можливість роботи в різних середовищах;

– інтернаціоналізація;

– доступність для будь-яких додатків;

– легка модифікація через стандартні функції API;

– легке перетворення з інших форматів. Приведемо приклад: використовуючи XSL-трансформацію, можна, наприклад візуалізувати хімічні молекули, описаних мовою CML (Chemical Markup Language).

Формат стиску 7z

Алгоритм стиску 7z, що лежить в основі програми-архіватора 7-Zip – ще один приклад відкритого формату. Серед його достоїнств такі:

– відкритий код;

– високий ступінь стиску;

– висока швидкість розпакування;

– багатопоточний стиск;

– підтримка криптостійкого шифрування;

– підтримка 64-бітних систем.

2.2 Обгрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies.

Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи реалізації інструментальних засобів організації пакетів прикладних програм.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі.

Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розглянемо основні поняття, пов'язані з організацією розподілених пакетів прикладних програм (РППП), формалізований опис предметної області РППП, розроблені язикові засоби подання опису предметної області РППП у форматі XML і сформульовані вимоги до інструментального комплексу, призначеному для створення й застосування РППП у різнорідних розподілених обчислювальних середовищах (РОС).

РППП орієнтуються на клас завдань, що характеризуються наступними властивостями:

– рішення завдання вимагає проведення розрахунків на ЕОМ з використанням більших обсягів обчислювальних ресурсів (процесорного часу, оперативної пам'яті, дискового простору й ін.);

– можлива декомпозиція загального складного завдання на більше прості (з обчислювальної точки зору) підзадачи;

– процес рішення загального завдання має на увазі розподілене рішення набору її взаємозалежних підзадач;

– не передбачається інтенсивної взаємодії між паралельними процесами рішення підзадач;

– завдання допускає декомпозицію даних на блоки й незалежну паралельну обробку цих блоків;

– рішення завдання виконується, як правило, по одній слабко змінюваній синхронній схемі.

До такого класу відносяться завдання, що вимагають проведення різноманітних розрахунків, а також багато переборних завдань.

Нижче наведені характерні відмінності середовища функціонування, язикових засобів, функціонального наповнення й системної частини РППП від відповідних складових традиційних пакетів:

– Середовище функціонування. Середовищем функціонування РППП є РОС із різнорідними обчислювальними вузлами, організованими на основі різних програмно-апаратних платформ і керованими різними ОС. Найчастіше різнорідні РОС мають низький ступінь відказостійкості.

– Язикові засоби. Як вхідна мова пакета використовується розширювана мова розмітки XML. Потрібні засоби опису великоблочного розподіленого алгоритму рішення завдання у РОС.

– Функціональне наповнення РППП. Як модулі використовуються програми, що виконуються в пакетному режимі, які можуть бути написані на різних мовах програмування (наприклад, C#, Delphi XE5 і ін.), функціонувати під управлінням різних ОС (наприклад, MS Windows, Linux, Mac OS X і ін.) і бути платформо-залежні. Модулі розміщаються в різних вузлах РОС, в одному вузлі РОС може бути встановлено кілька модулів. Припустимо включення до складу функціонального наповнення успадкованого ПЗ, а також нетиражуємих програмних комплексів, жорстко прив'язаних до вузлів РОС. Обмін даними між модулями здійснюється через файли.

– Системна частина. Потрібна організація обчислень на основі віддаленого запуску модулів і розподіленого обміну даними. Необхідний моніторинг вузлів РОС.

Більшість інструментальних систем (наприклад, OLYMPUS, ПРИЗ, САФРА, СУПЕРЕЧКИ й ін.) розроблялися, як правило, для створення традиційних ППП. Застосування таких систем для побудови РППП є скрутним. Відомі системи організації розподілених обчислень (наприклад, кластерна система Condor, програмний комплекс BOINC, інструментарій X-COM і др.) дозволяють здійснити у РОС рішення не зв'язаних між собою завдань, що допускають розпаралелювання по даним, але не мають всі необхідні можливості

для реалізації перерахованих вище особливостей розподіленого пакета взаємозалежних прикладних програм. У цьому зв'язку виникає необхідність розробки інструментарію для організації спеціалізованого виду прикладних програмних комплексів – розподілених пакетів прикладних програм (РППП).

Властивості класу завдань, характерного для РППП (зокрема, рішення завдання по одній слабко змінюваній схемі), спричиняються вибір процедурного способу постановок завдань пакету. Ці ж властивості в сукупності з особливостями функціонального наповнення РППП (використання як модулі програм, що виконуються в пакетному режимі й розміщених у різних вузлах РОС) приводять до необхідності реалізації процесу рішення завдання по заданій синхронній схемі в режимі інтерпретації.

Формалізований опис предметної області РППП відноситься до класу обчислювальних моделей і являє собою сукупність значимих параметрів предметної області, а також модулів РППП, що реалізують обчислювальні операції із цими параметрами.

У найпростішому випадку опис предметної області РППП можна визначити у вигляді структури:

$$S =< Z, T, M, Y >,$$

де Z – множина параметрів, T – множина припустимих типів параметрів, M – множина модулів, Y – множина вузлів РОС, у яких розміщений той або інший модуль із M. Зв'язку між елементами множин Z, T, M і Y задані відносинами $ZT \subset Z \times T$, $IN \subset Z \times M$, $OUT \subset Z \times M$ і $MY \subset M \times Y$ (у загальному випадку типу «багато-к-багатьох»). Для кожного об'єкта структури S (параметра, типу параметра, модуля й вузла) визначений набір його атрибутів. Множина T включає наступні типи параметрів: тип *file,* використовуваний для опису параметрів невизначеної структури (блоків довільного тексту великого розміру); тип *filelist,* призначений для підтримки розпаралелювання по даним (паралельний список параметрів типу *file);* тип *fileconst,* уведений з метою скорочення обсягів передачі даних у РОС (значення параметра типу *fileconst* один раз передається вузлу РОС і

потім може неодноразово використовуватися при запуску модулів, розміщених у даному вузлі). Змістовно модуль $m_t \subset M$ реалізує можливість обчислення множини параметрів $OUT \subset Z$ по множини параметрів $IN^l \subset Z$, а множини IN^l і OUT^l називаються відповідно множинами вхідних і вихідних параметрів для модуля m_t. Заелементна обробка параметра z_j типу *filelist* модулем m_t виконується в такий спосіб: k-*й* елемент параметра z_{jk} обробляється k-*м* екземпляром модуля m_{ik}. Опис предметної області РППП будемо вважати цілісним, якщо виконуються наступні умови:

1. Умова наявності опису предметної області:

$$(Z \neq \varnothing) \wedge (M \neq \varnothing).$$

2. Умова наявності вхідних і вихідних параметрів модулів:

$$\forall m_i \in M \ (IN^i \neq \varnothing) \wedge (OUT^i \neq \varnothing).$$

3. Умова цілісності відносин *IN* і *OUT*:

$$\forall m_i \in M \ IN^i \bigcup OUT^i \subset Z.$$

4. Умова відсутності в модулях транзитних параметрів:

$$\forall m_i \in M \ IN^i \bigcap OUT^i = \varnothing.$$

5. Умова ненадмірності множини параметрів:

$$\forall z_j \in Z \ \exists m_i \in M : z_j \in IN^i \bigcup OUT^i.$$

Постановка завдання для структури *S* задається в процедурному виді й у загальному випадку формулюється в такий спосіб: «знаючи *S* виконати Q», де $Q \subset M$ являє собою частково впорядковану послідовність модулів з M, які необхідно виконати для рішення завдання. При встановленні часткового порядку множина *Q* розбивається на до непустих підмножин. Упорядкування підмножин здійснюється залежно від того, модулі якої підмножини повинні бути виконані раніше. У рамках кожної *к*-ї підмножини вхідні в нього модулі можуть виконуватися незалежно друг від друга в будь-якій послідовності або паралельно.

Схемою рішення завдання (СРЗ) будемо називати модель великоблочної програми, що відбиває інформаційно-логічну структуру обчислень у термінах

предметної області. СРЗ будується в паралельно-ярусній формі з елементів наступних множин:

– множини параметрів Z; множини модулів M;

– множини подій E, що виникають у процесі виконання СРЗ;

– множини функцій F, призначених для управління процесом виконання СРЗ;

– множини спеціальних операторів:

O = {START, STOP, READ <список параметрів>,
WRITE <список параметрів>,
CALL <ім'я модуля>,
FORK,
JOIN,
TERMINATE <список модулів>}.

Інформаційно-логічні зв'язки між параметрами й модулями СРЗ представляються у вигляді структури $H =< IN^h, \cup, OUT^h >$, де IN^h і OUT^h – множини вхідних і вихідних параметрів СРЗ, відповідно. Структура H зображується двочастковим орієнтованим інформаційним графом G_I, що включає дві непересічні множини вершин: вершин-параметрів і вершин-модулів. Вершини-параметри, як і вершини-модулі, є попарно несуміжними. Частково впорядковану послідовність модулів Q зручно задавати у вигляді булевої матриці розмірності $к \times n$, де $к$ – число ярусів СРЗ, а n – число модулів в M. Елемент матриці q = 1 означає, що модуль m розміщений на l-*о*му ярусі СРЗ.

Побудова СРЗ здійснюється у два етапи:

– розроблювач РППП формує яруси СРЗ, розміщає на цих ярусах модулі з M і визначає події з E, при виникненні яких необхідно виконати ті або інші керуючі функції з F;

– транслятор постановки завдання на основі інформаційно-логічних зв'язків між параметрами й модулями структури S визначає множини вхідних і

вихідних параметрів СРЗ і доповнює постановку завдання спеціальними операторами з *O*. СРЗ будемо називати припустимою, якщо вона задовольняє наступним умовам:

1. Умова процедурної постановки завдання:

$$(Q \neq \varnothing) \wedge (IN^h \bigcup OUT^h = \varnothing).$$

2. Умова допустимості включення модуль у СРЗ:

$$\forall m_i \in Q : q_{pi} = 1 \quad IN^i \setminus (IN^h \bigcup (\bigcup_{j=1}^{i-1} \bigcup_{\substack{i=1 \\ \forall q_{ji}=1}}^{n} OUT^i)) = \varnothing.$$

3. Умова інформаційної незалежності модулів, розташованих на тому самому ярусі СРЗ:

$$\forall m_i \in Q : q_{pi} = 1 \quad IN^i \bigcap (\bigcup_{\substack{j=1 \\ (\forall q_{ji}=1) \wedge (j \neq i)}}^{n} OUT^j) = \varnothing.$$

Керуючим графом СРЗ будемо називати впорядкований орієнтований граф O_u, вершинами якого є оператори виклику модулів з *M* (або їхніх екземплярів), а також спеціальні оператори.

Кожна дуга v_j графа O_u характеризується парою величин (/; c), де / є {0,1} – це логічний перемикач, що визначає можливість (/=1) або неможливість (/=0) переходу по дузі v_j, а *c* – обмежник числа переходів по дузі v_j. Запис (; c) для дуги $v_]$ означає безумовний перехід по даній дузі.

Виконання СРЗ у режимі інтерпретації являє собою процес послідовно-паралельного виконання її операторів відповідно до порядку їхнього входження в граф O_u. Передача даних (значень параметрів) між модулями виконується відповідно до графа О. Виконання СРЗ будемо вважати *коректним,* якщо воно завершується оператором *STOP* за кінцеве число кроків.

Опис предметної області РППП і постановок завдань представляються у форматі, розробленому на основі розширюваного метамови XML. Функціональні й системні вимоги до інструментального комплексу DISCOMP, призначеному для створення й застосування РППП у різнорідних РОС, сформульовані на основі аналізу відомих методів і засобів організації розподілених обчислень.

Основні тенденції в розвитку ППП

Як висновок відзначимо перспективні напрямки подальшого розвитку прикладного ПЗ. На сьогоднішній день як основних факторів, що впливають на функціональність ППП і складність їхньої розробки ПЗ, можна відзначити наступні:

– ріст продуктивності персональних комп'ютерів;
– розширення класів розв'язуваних завдань;
– збільшення загального числа користувачів;
– значна кількість раніше створеного (успадкованого) ПЗ;
– розвиток Інтернет і корпоративні мережі.

Розробка додатків з урахуванням цих факторів привела до появи прикладних пакетів і інтегрованих середовищ, які по своїх характеристиках виходять за рамки ППП четвертого покоління. Серед відмітних рис ПЗ нового покоління наступні:

– інтеграція компонентів прикладного пакета не тільки з додатками пакета, але й з оточенням;
– широке використання галузевих стандартів;
– використання інфраструктури Інтернет;
– платформонезалежність.

Особливу значимість на подальший сценарій розвитку ППП має вплив технологій Інтернет і, зокрема, Web. Можливості, що представляються глобальною мережею дозволяють обмінюватися будь-якою інформацією, яку можна представити в цифровому виді. Це вже зараз із успіхом використовується в провідних пакетах прикладних програм, у першу чергу для забезпечення спільної роботи користувачів. Практична реалізація загального доступу можлива, наприклад, з використанням проміжного ПЗ (middleware). Так, при використанні технології Active, у документ MS Word або таблицю MS Excel можна помістити будь-який документ, що підтримує Active. Впровадженим може бути документ,

розміщений в Інтернет, більше того, є потенційна можливість відредагувати його й зберегти зміни в Мережі.

Все більшою популярністю користується концепція «тонких клієнтів». Під «тонким клієнтом» мається на увазі Інтернет-браузер. Сучасні браузери дозволяють відображати не тільки гіпертекстові документи, але й зображення в растрових і векторних форматах, відео- і аудіодані. Крім цього, браузери представляють засоби інтерактивної взаємодії з веб-серверами у вигляді різних веб-форм (від форм авторизації або пошуку до форм завантаження файлів) і підтримують виконання програм-скриптів на своїй стороні. Це дозволяє створювати програми, що завантажуються з веб-сервера, але виконувані в браузері. Прикладом такого рішення є сервіси Google Docs (Google Документи). Користувачам цього сервісу представляється можливість створювати й редагувати текстові документи, електронні таблиці й презентації прямо у вікні браузера, зберігати їх в Інтернет і надавати в спільне використання.

Через повсюдне проникнення Інтернету, можна говорити про те, що прикладне програмне забезпечення буде переходити в розряд сервісу, тобто користувачі будуть працювати з необхідним програмним забезпеченням через Мережу, одержуючи на свої комп'ютери готові результати. Отже, необхідність у більших локальних потужностях частково відпаде, що буде сприяти росту попиту на недорогі комп'ютери з низьким енергоспоживанням.

В основі технологій, що забезпечують подібні можливості, ряд спільних наробітків провідних виробників ПЗ і організацій по стандартизації. До них відносяться сервісно-орієнтована архітектура корпоративних додатків (веб-сервіси) і стандартизовані формати документів.

Веб-сервіси

Веб-сервіс – програмна система, доступна через локальну або глобальну мережу по заданій адресі, чиї загальнодоступні інтерфейси визначені мовою XML. Ця програмна система доступна іншим програмними системами, які можуть взаємодіяти з нею за допомогою XML-повідомлень. переданих за

допомогою інтернет-протоколів. Веб-служба є одиницею модульності при використанні сервісно-орієнтованої архітектури додатка. Іншими словами, веб-сервіс – це іменований компонент розподіленої прикладної системи, доступний по гіпертекстових протоколах.

Сервісно-орієнтовані додатки побудовані на наступних індустріальних стандартах:

– XML: Розширювана мова розмітки, призначена для зберігання й передачі структурованих даних;

– SOAP: Протокол обміну повідомленнями на базі XML;

– WSDL: Мова опису зовнішніх інтерфейсів веб-служби на базі XML;

– UDDI: Універсальний інтерфейс розпізнавання, описи й інтеграції (Universal Discovery, Description, and Integration). Каталог веб-служб і відомостей про компанії, що надають веб-служби в загальне користування або конкретні компанії.

Основними достоїнствами веб-сервісів є:

– інтероперабельність;

– відкритість архітектури;

– взаємодія програмних систем через засоби захисту інформації (проксі-сервери, міжмережні екрани).

Основним недоліком є менша продуктивність додатків і більший обсяг мережного трафіку в порівнянні з іншими технологіями розподілених обчислень (RMI, CORBA, DCOM/Active). Ще одним недоліком є підвищена вимогливість до апаратних ресурсам на стороні сервера додатків (постачальника веб-сервісів).

Уніфікація форматів

Перспективним напрямком у розвитку ППП є використання уніфікованих форматів документів на основі відкритих стандартів. Відкритий стандарт – загальнодоступна специфікація, звичайно розроблювальна некомерційною організацією по стандартизації, вільна від ліцензійних обмежень при

використанні. Відкриті формати є підмножиною відкритих стандартів і визначають специфікації зберігання й подання цифрових даних.

Використання відкритих форматів у ППП дозволяє гарантувати можливість доступу до даних з будь-якого сумісного додатка без оглядки на ліцензійні права й технічні специфікації. Актуальність концепції відкритих форматів підтверджується практикою – урядові організації багатьох країн використовують їх як основний засіб.

На сьогоднішній день розроблені й застосовуються відкриті формати практично для всіх класів завдань, розв'язуваних ППП, починаючи від офісних додатків до мультимедійних даних і 3D-графіки.

Додатки за запитом

Поширення веб-сервісів на основі відкритих стандартів веде до ситуації, коли замість запуску певних програм корпоративні користувачі зможуть одержати доступ до будь-яких прикладних засобів, необхідним у цей момент, просто підключившись до Інтернет. У такому контексті додатка можуть бути представлені як вільно й безкоштовно, так і платно, по підписці, залежно від обсягу споживання.

Сполучення широкополосного інтернету із платформонезалежними додатками (написаними, приміром, мовою Java) у деяких областях уже зробили модель комунальних послуг в області IT реальністю. Наприклад, Salesforce.com за помірну місячну плату пропонує в Інтернеті додатка для керування відносинами із клієнтами (CRM). Користувачам, кількість яких уже становить близько 100 тис., не потрібно встановлювати або підтримувати в себе складні пакети CRM. Їм досить тільки запустити браузер і підключитися до серверу і послуг. У свою чергу вільний доступ до офісних додатків представляє раніше згаданий сервіс Google Docs, число користувачів якого постійно росте.

3.2 Розробка структурної схеми

Розглянемо структурну схему системи, зображену на рисунку 3.1. В архітектурі можна виділити наступні складові:

– систему управління РОС (СУРОС);

– набір обчислювальних клієнтів РОС;

– систему зберігання даних;

– засоби доступу користувачів до РППП.

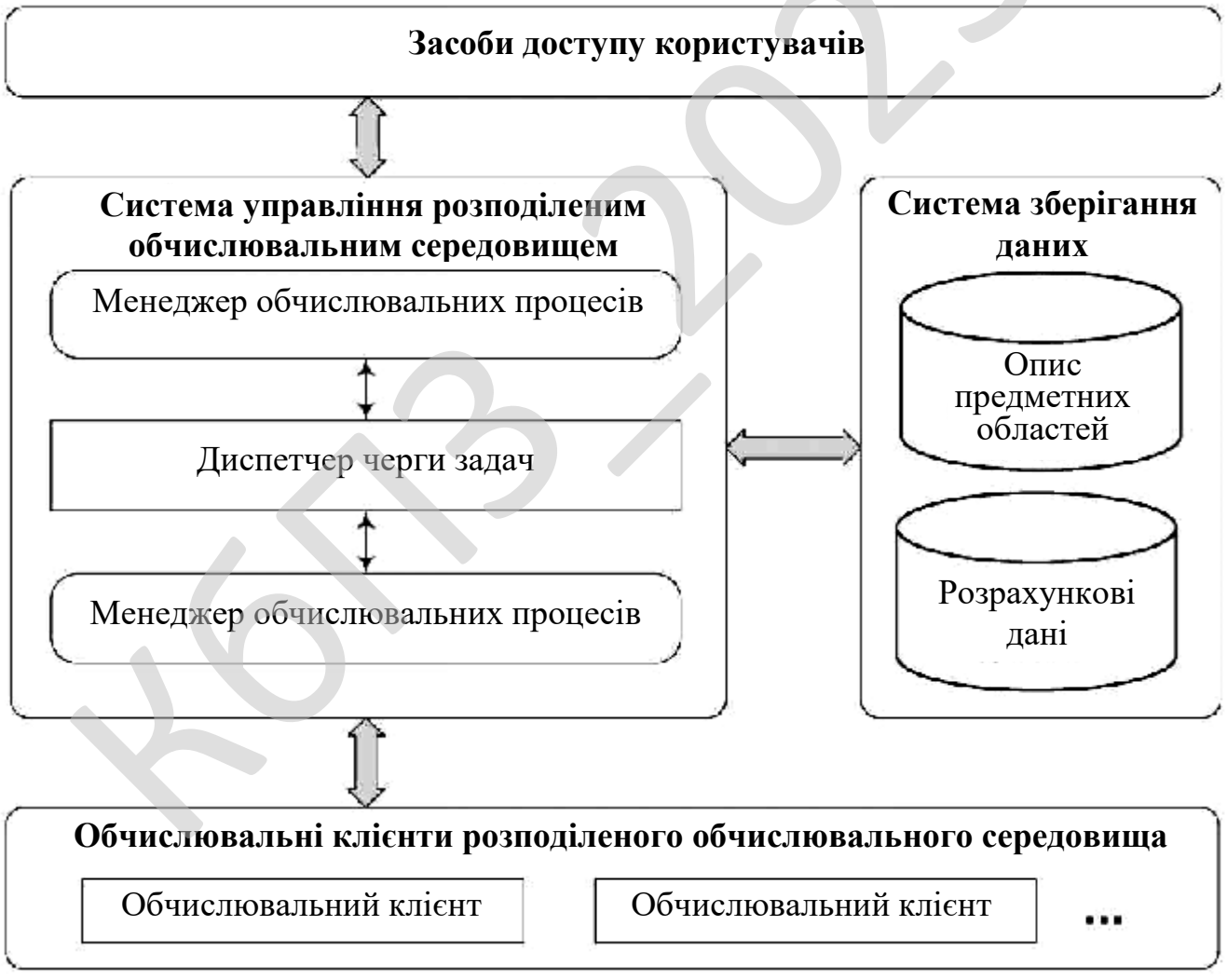

Рисунок 3.1 – Структурна схема системи

Процес виконання (інтерпретації) СРЗ у РОС являє собою обчислювальний процес, що включає трансляцію цієї схеми у внутрішнє подання і її покрокову інтерпретацію. Інтерпретація операторів виклику модулів вимагає виконання відповідних модулів у вузлах РОС.

3.3 Розробка функціональної схеми

Розглянемо функціональну схему системи, зображену на рисунку 3.2, що представляє розширений опис функцій структурної схеми. СУРОС являє собою комплекс взаємозалежних підсистем, призначених для централізованого управління РОС, виконання обчислювальних процесів, диспетчеризації черги завдань і виконання ряду інших функцій. СУРОС включає наступні основні компоненти:

– системне ядро;

– менеджер обчислювальних процесів;

– менеджер обчислювальних ресурсів;

– диспетчер черги завдань;

– підсистему журналювання;

– виконавчу підсистему.

Обчислювальний клієнт призначений для організації процесу виконання модуля у вузлі РОС. Обчислювальний клієнт виконує наступні функції:

– створення відповідного середовища виконання модуля (створення тимчасових директорій, файлів вхідних і вихідних параметрів модулів, завдання значень змінні оточення, перенапрямок уведення/виводу й т.д.);

– одержання значень вхідних параметрів модуля з керуючого вузла;

– запуск модуля;

– контроль процесу його виконання;

– відсилання значень вихідних параметрів у керуючий вузол після завершення роботи модуля.

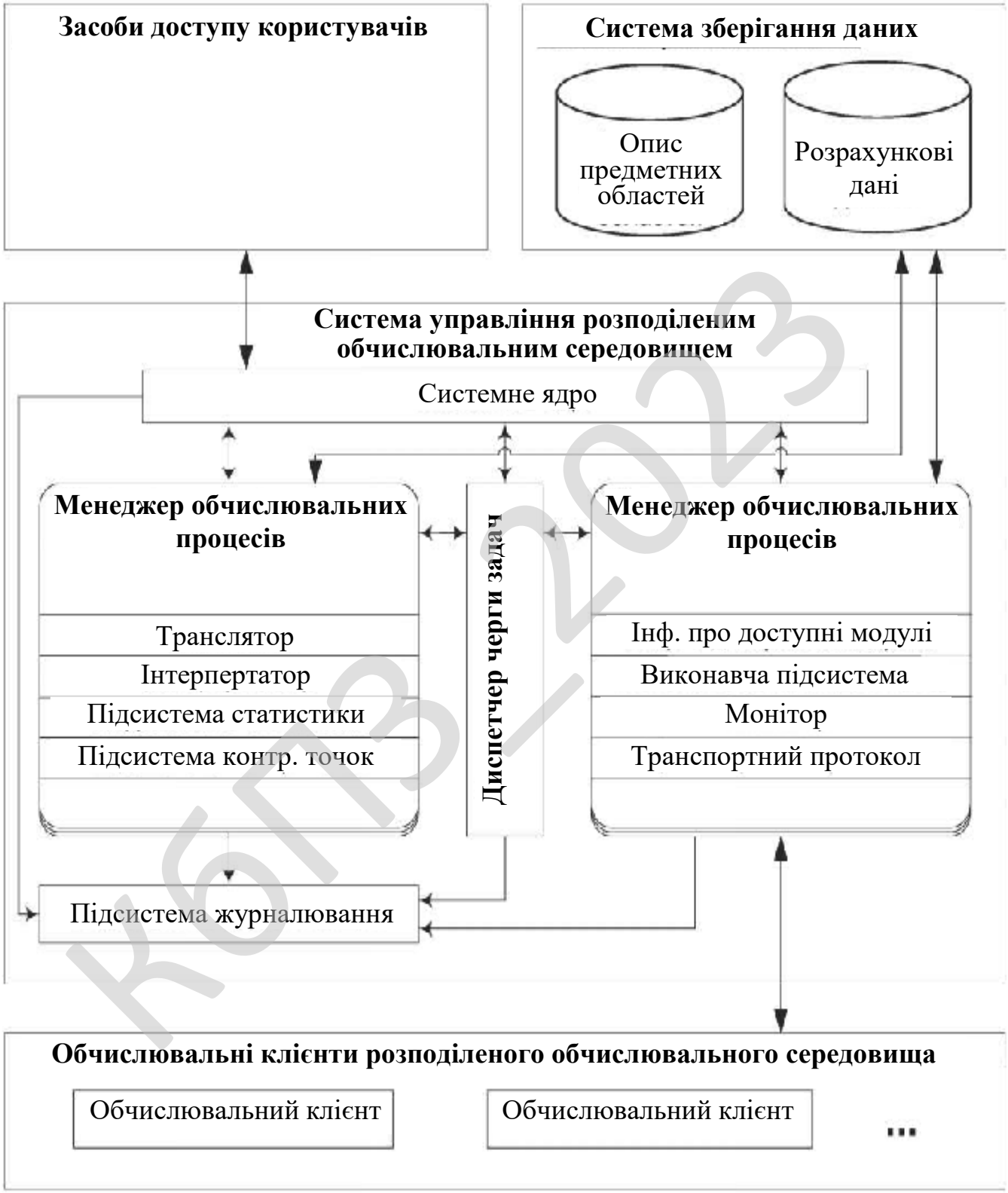

Рисунок 3.2 – Функціональна схема системи

Система зберігання даних призначена для структурованого зберігання описів предметної області й постановок завдань у форматі XML, а також розміщення розрахункових даних у вигляді файлів і надання доступу до них з різних підсистем. Синхронізація процесів читання/запису даних здійснюється за допомогою файлових блокувань.

Засоби доступу користувачів до розподілених пакетів прикладних програм (РППП) призначені для взаємодії користувача з пакетом, розміщеним у РОС, управління обчислювальними процесами, уведення даних, одержання результатів обчислень та ін. надає два способи взаємодії користувача із РППП: за допомогою набору утиліт командного рядка й за допомогою інтерактивного користувальницького web-інтерфейсу.

Зв'язок компонентів між собою реалізована на основі спеціально розробленого *протоколу мережної взаємодії.* Механізм мережної взаємодії вузлів з архітектурою «клієнт-сервер» реалізований з використанням засобів, надаваних крос-платформною бібліотекою класів Qt. Обмін даними відбувається по TCP-каналі, що забезпечує доставку повідомлень зі збереженням порядку пакетів і управління потоком даних. Серверна частина мережного додатка реалізована з підтримкою принципу багатопоточності, що дозволяє обробляти підключення на центральній машині паралельно, без запропонованого порядку в часі.

Розроблені в даному розділі магістерської роботи структура й принципи функціонування, а також способи й засоби її реалізації забезпечують широкий спектр функціональних можливостей даного комплексу для організації розподілених обчислень. Компоненти розробленого програмного забезпечення є крос-платформними й дозволяють використовувати весь потенціал доступних різнорідних вузлів РОС.

Ефективність рішення завдань у РППП досягається шляхом використання багатозадачного режиму функціонування, виділення ресурсів завданням на основі їхніх пріоритетів, динамічного управління процесами рішення завдань, а також за допомогою забезпечення гарної продуктивності взаємодії розподілених

підсистем між собою (така продуктивність досягається встановленням безперервного з'єднання між клієнтом і сервером).

Розроблене програмного забезпечення забезпечує необхідний рівень безпеки як для РППП, так і для РОС, що є середовищем функціонування РППП. Відповідні компоненти програмного забезпечення реалізують автентифікацію й авторизацію користувачів, обмеження доступу користувачів до сервера й віддалених вузлів РОС, розмежування прав користувачів, захист їхніх програм і даних від несанкціонованого використання, належний рівень відказостійкості на основі обчислювальної надмірності вузлів РОС і інші елементи безпеки всієї обчислювальної системи.

Дистрибутив програмного забезпечення включає всі необхідні засоби для установки програмного забезпечення у вузлах з типовою конфігурацією. Така комплектність дистрибутива виключає необхідність установки якого-небудь додаткового ПЗ і забезпечує швидку установку й налаштування програмного забезпечення (у тому числі й користувачами, що не володіють навичками системного адміністрування).

Працездатність протестована у РОС із вузлами, що функціонують під управлінням наступних ОС: MS Windows, Gentoo Linux, ASP Linux, Fedora Core Linux, Mac OS X.

Розробка архітектури й програмна реалізація виконані особисто автором.

Розглянемо технологію організації РППП, у загальному випадку яка включає наступні основні етапи:

1. Проектування: проведення структурного аналізу досліджуваної предметної області з метою виявлення її основних об'єктів і їхніх взаємозв'язків; визначення класу завдань, розв'язуваних у рамках предметної області; проведення декомпозиції складних завдань на більше прості в обчислювальному змісті підзадачи.

2. Розробка функціонального наповнення: створення або вибір з успадкованого ПЗ модулів, що реалізують процеси рішення завдань або окремих підзадач.

3. Створення системної частини: установка компонентів програмного забезпечення у вузлах РОС; конфігурування РОС, визначення політик безпеки й правил доступу, реєстрація користувачів і т.д.

4. Опис предметної області: опис параметрів і модулів предметної області; формування процедурних постановок завдань.

5. Розміщення модулів: розміщення модулів і їхньої специфікацій у вузлах РОС (виконується в ручному або автоматичному режимах).

6. Налагодження й тестування: виконання серії тестів на різних наборах даних, виявлення й виправлення помилок.

7. Застосування:

– вибір потрібної постановки завдання;

– підготовка вихідних даних (значень вхідних параметрів завдання);

– завдання вимог до ресурсів РОС, необхідних для рішення завдання;

– запуск завдання на розрахунок;

– контроль над виконанням обчислювального процесу (аналіз проміжних результатів, файлів журналу, статистики по використанню ресурсів та ін.);

– аналіз результатів обчислень.

8. Супровід: включення до складу РППП нових модулів; формування нових постановок завдань; консультування користувачів і т.д.

Розроблена технологія апробована при рішенні ряду завдань із різних предметних областей.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після

початку роботи розробленого ПЗ ми потрапляємо до інтерфейсу ПЗ звідки можемо потрапити до довідкової інформації, налаштування ПЗ та системи зберігання даних з подальшим переходом до розрахункових скриптів та предметної області.

Крім цього з інтерфейсу ПЗ можна потрапити до система управління розподіленим обчисленим середовищем з подальшим переходом до менеджеру обчислювальних процесів доступу користувачів, диспетчеру черги задач, менеджеру обчислювальних процесів РОС, зв'язку з обчислювальним клієнтом.

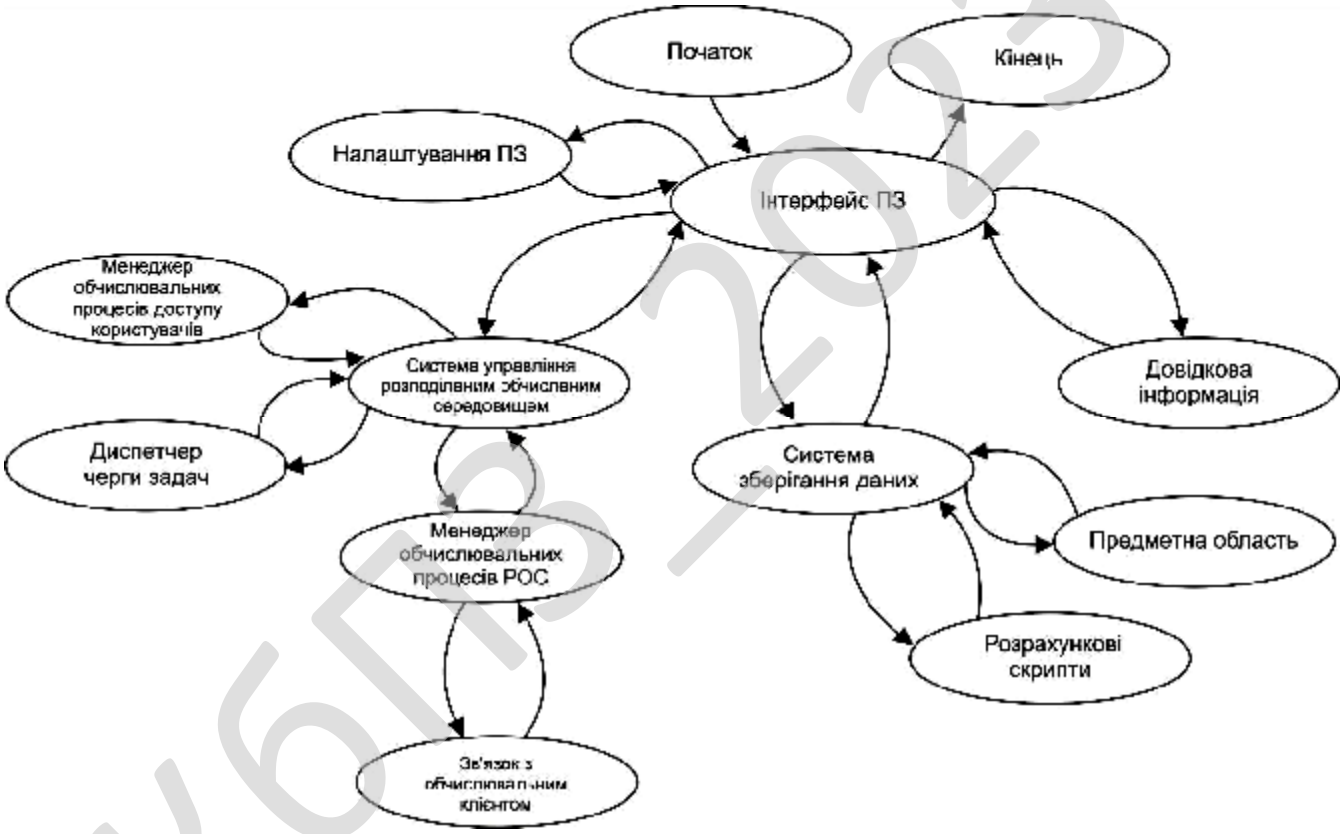

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми здійснюється наступне:

– Виділення пам'яті ПЗ.

– Ініціалізація ресурсів ПЗ.

– Читання налаштувань ПЗ.

– Спроба підключення до системи зберігання даних, перевірка цілісності.

– Перевірка пройдена (запит).

– Перевірка прав доступу до ОС.

– Перевірка доступу пройдена? (запит).

– Ініціалізація менеджеру обчислювальних процесів РОС.

– Ініціалізація менеджеру обчислювальних процесів доступу користувачів.

– Запит розрахункових скриптів та запуск диспетчера черги задач.

– ДЧЗ запущено (запит).

– Запуск системи журналювання.

– Запит предметної області.

– Очікування дії користувача.

– Запит оновлення розрахункових даних.

– Підпрограма оновлення розрахункових даних.

– Запит запуску обчислювальних клієнтів?

– Підпрограма обчислювальні клієнти.

– Запит запуску виконавчої підсистеми?

– Підпрограма виконавчої підсистеми.

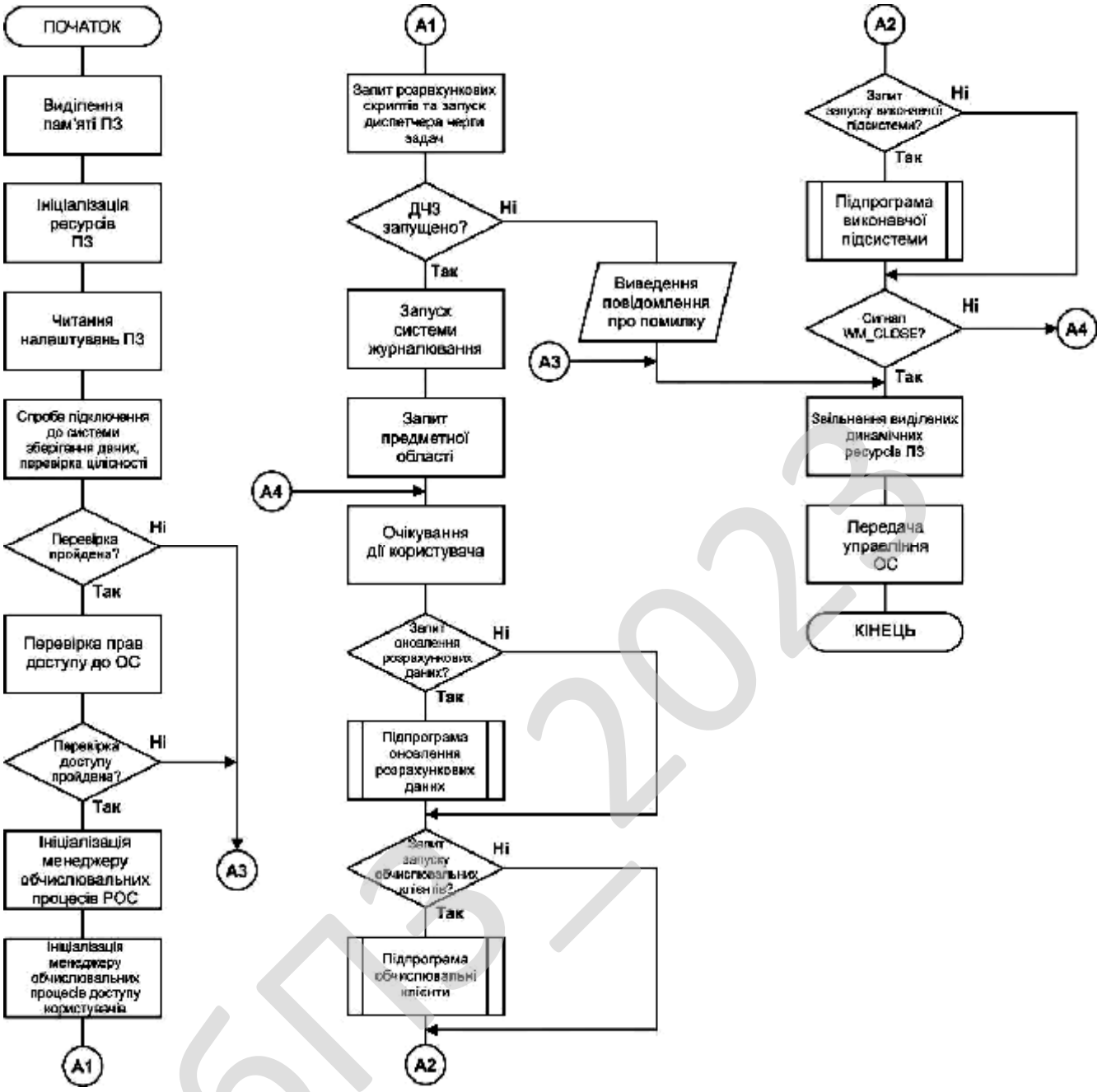

Рисунок 4.1 – Блок-схема основної програми

– Сигнал WM_CLOSE?

– Звільнення виділених динамічних ресурсів ПЗ.

– Передача управління ОС.

На рисунку 4.2 зображено роботи підпрограми оновлення розрахункових даних, де відбуваються наступні дії:

– Читання файлу розрахункових даних системи.

– Локальне оновлення даних системи (запит).

– Пошук файлу оновлення розрахункових даних системи.

– Файл знайдено (запит).

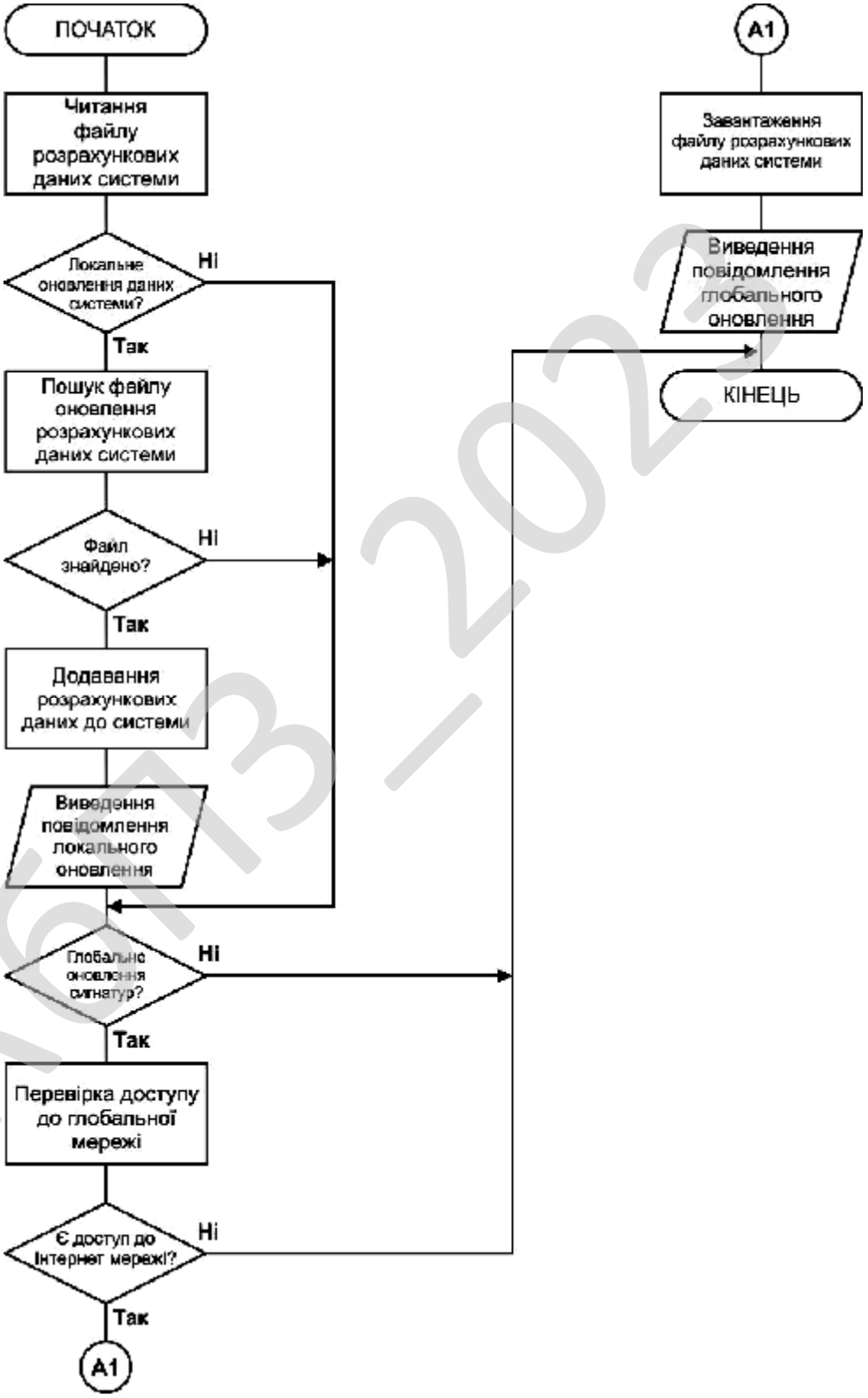

Рисунок 4.2 – Блок-схема підпрограми оновлення розрахункових даних

– Додавання розрахункових даних до системи.

– Виведення повідомлення локального оновлення.

– Глобальне оновлення сигнатур (запит).

– Перевірка доступу до глобальної мережі.

– Є доступ до Інтернет мережі (запит).

– Завантаження файлу розрахункових даних системи.

– Виведення повідомлення глобального оновлення.

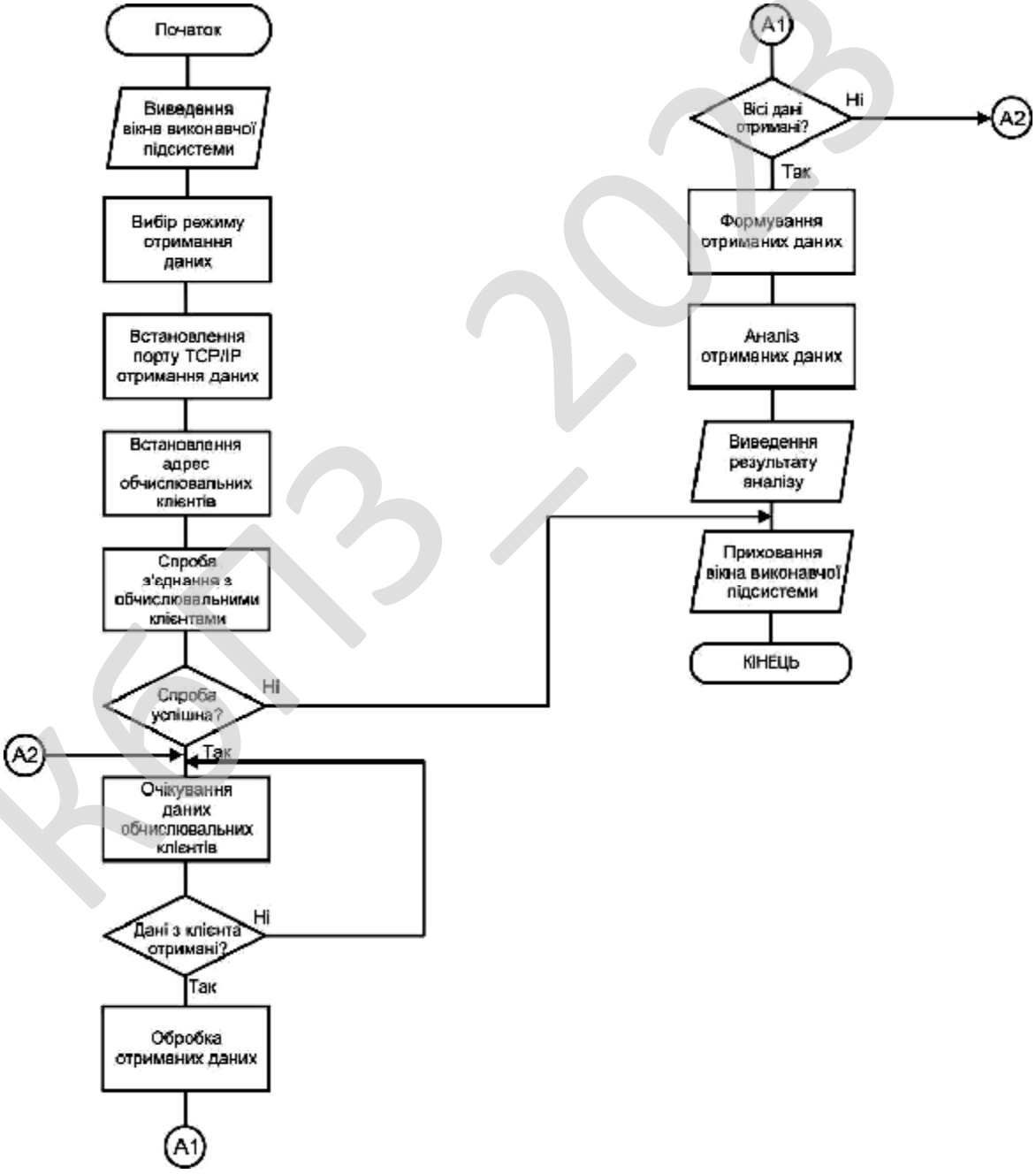

Рисунок 4.3 – Блок-схема підпрограми виконавчої підсистеми

На рисунку 4.3 зображено роботу підпрограми виконавчої підсистеми, де відбуваються наступні дії:

– Виведення вікна виконавчої підсистеми.

– Вибір режиму отримання даних.

– Встановлення порту TCP/IP отримання даних.

– Встановлення адрес обчислювальних клієнтів.

– Спроба з'єднання з обчислювальними клієнтами.

– Спроба успішна (запит).

– Очікування даних обчислювальних клієнтів.

– Дані з клієнта отримані (запит).

– Обробка отриманих даних.

– Вісі дані отримані (запит).

– Формування отриманих даних.

– Аналіз отриманих даних.

– Виведення результату аналізу.

– Приховання вікна виконавчої підсистеми.

Опис алгоритмів функціонування системи

Розглянемо реалізацію одержання списку поточних сесій. Для визначення користувачів підключених до комп'ютера була розроблена функція Net_session. Оголошення функції для Windows 10/11:

```
Net_session: function(Servername, Uncclientname, Username: Pwchar; Level: DWORD; bufptr: Pointer; prefmaxlen: DWORD;entriesread, totalentries, resume_handle: LPDWORD):DWORD;
```

Параметри:

-Servername повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо дивимося в себе то даному параметрові потрібно привласнити NIL.

– UnClientName містить покажчик на рядок утримуючу ім'я сесії.

– Username містить покажчик на рядок утримуючу ім'я користувача.

– Level повинен містити ідентифікатор структури.

– Bufptr повинен містити адреса покажчика на масив структур.

– Prefmaxlen повинен містити максимальну довжину повернутих даних у байтах

– Entriesread повинен містити покажчик на змінну в яку запишеться кількість загальних ресурсів доступних на даний момент.

Для чіткого представлення часу роботи в магістерської програмі була розроблена функція, завдання якої перетворювати кількість секунд у більш звичну форму відображення.

```
function Tmainform.Cardinaltotimestr(Value:Cardinal):String;
       var d,h,m,s: Real;
       begin
           d:=0;     h:=0;    m:=0;    s:=Value;
           if s > 59 then begin
               m:=int(s / 60);
               s:=s - (m*60);
           end;
           if m > 59 then begin
               h:=int(m/60);
               m:=m - (h*60);
           end;
           if h > 23 then begin
               d:=int(h/24);
               h:=h - (d*24);
           end;
           Result:='';
    if (d>0) then Result:=Result+floattostr(d)+' d. ';
    if (h<9) then Result:=Result+'0'+floattostr(h)+':' else
                  Result:=Result+floattostr(h)+':';
    if (m<9) then Result:=Result+'0'+floattostr(m)+':' else
                  Result:=Result+floattostr(m)+':';
    if (s<9) then Result:=Result+'0'+floattostr(s)     else
                  Result:=Result+floattostr(s);
       end;
```

Розробка механізму завершення сесій. Для завершення відкритих сесій розроблена функція NetSessionDel.

Оголошення функції:

```
Var NetSessionDel: function(Servername,
                            Uncclientname, Username:Pwchar):DWORD;
```

Параметри:

– Servername повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція;

– uncclientname повинен містити ім'я клієнта чия сесія завершується;

– username повинен містити ім'я користувача чия сесія завершується, якщо параметр NIL, завершаться всі сесії указані в параметрі uncclientname.

Розглянемо вихідний код виклику функції:

```
procedure Tmainform.btnclosesessionclick(Sender: Tobject);
var
OS: Boolean; Flibhandle : Thandle; Cnament: Pwidechar; Cname9x: Pansichar;
  Key:Smallint; i: Integer;
begin
  if not OS(OS) then Close;
//З'ясовуємо тип системи
  if not Assigned(lvsessions.Selected) then Exit;
  i:= lvsessions.Selected.Index;
//Визначаємо номер обраної сесії
  if OS then begin
    Flibhandle := Loadlibrary('MY_1.DLL');
    if Flibhandle = 0 then Exit;
@Netsessiondelnt:=Getprocaddress(Flibhandle, 'Netsessiondel');
    if not Assigned(Netsessiondelnt) then
    begin
      Freelibrary(Flibhandle); Exit;
    end;
    //Перетворимо дані в необхідний вигляд
    Cnament := Pwchar(Widestring('\\'+lvsessions.Items.Item[i].Caption));
    Netsessiondelnt(nil,Cnament,nil);
  end else begin
    Flibhandle := Loadlibrary('MY_2.DLL');
    if Flibhandle = 0 then Exit;
@Netsessiondel:=Getprocaddress(Flibhandle, 'Netsessiondel');
    if not Assigned(Netsessiondel) then
    begin Freelibrary(Flibhandle); Exit;   end;
    //Перетворимо дані в необхідний вигляд
    Cname9x := Pansichar(lvsessions.Items.Item[i].Caption);
```

```
            key := Sessionclosekey[i];
        // Беремо ключ із масиву
            Netsessiondel(nil,Cname9x,Key);
          end;
          Freelibrary(Flibhandle);
        end;
```

Розглянемо використаний компонент TNMHTTP (NetMasters HTTP):

```
{ПОЧАТОК йде заголовок файлу і визначення форми (TForm1) і її екземпляра Form1. У формі є кнопка TButton і одне поле TEdit. При натисненні на кнопку викликається обробник події OnClick – Button1Click. }
        procedure Button1Click(Sender: TObject);
        begin
        {Намагаємося отримати заголовок}
        NMHTTP1.Head(Edit1.Text);
        end;
        …
        //Викачування вказаної URL в папку розробленого ПЗ
        {ПОЧАТОК Йде заголовок файлу і визначення форми
        TForm1 і її екземпляра Form1. У формі є кнопка TButton і три
        поля TEdit. При натисненні на кнопку викликається обробник
        події OnClick – Button1Click. Перед цим в перший TEdit введено
        адресу URL, в другій – ім'я файлу для заголовка, а у третій –
        ім'я файлу для тіла html}

        procedure Button1Click(Sender: TObject);
        begin
        {Намагаємося отримати http-документ}
            {Результат треба записати у файли}
        NMHTTP1.InputFileMode := True;
        { вказуємо в які саме файли}
        NMHTTP1.Header := Edit2.Text;
        NMHTTP1.Body := Edit3.Text;
        NMHTTP1.Get(Edit1.Text);
        end;
```

Розгляд викачування відразу декількох URL одночасно необхідно розглянути більш детально. У DELPHI дуже легко створювати окремі, підлеглі процеси за допомогою базового класу TThread.

Розглянемо приклад реалізований у ПЗ – одночасне викачування вказаних URL в заданий каталог.

```
….
{ПОЧАТОК Йде заголовок файлу і визначення
форми TForm1 і її екземпляра Form1,
опис класу окремого процесу}
type
THTTPThread = class(TThread)
Private
{Для кожного процесу – створюємо свій компонент TNMHTTP}
FHTTP: TNMHTTP;
Protected
{Execute викликається при запуску процесу;
override – замінюємо існуючу
процедуру базового класу TThread}
procedure Execute; override;
{DoWork – створена функція, виконання
якої синхронізується в Execute}
procedure DoWork;
public
{Вказує процесу, який URL йому потрібно викачати}
URL: string;
end;
{У форму потрібно помістити три кнопки TButton,
одне поле TEdit і один список TListBox. При
натисненні на кнопку Button1 викликається
обробник події OnClick – Button1Click. Перед
цим в TEdit потрібно ввести шлях до каталогу,
в якому зберігатимуться викачані файли, а ListBox1
потрібно заповнити списком URL-лів для викачування
(за допомогою кнопок Add (Button2) і Delete (Button3))
}
procedure TForm1.Button3Click(Sender: TObject);
begin
{Видалення виділеного URL із списку}
if ListBox1.ItemIndex >= 0 then
ListBox1.Items.Delete(ListBox1.ItemIndex);
end;

procedure TForm1.Button2Click(Sender: TObject);
var s: string;
begin
{Додавання URL в список}
s := InputBox('Input','Введення URL:','');
```

```
if s <> '' then ListBox1.Items.Add(s);
end;

procedure TForm1.Button1Click(Sender: TObject);
var i: Integer;
begin
{Перевірка на існування каталогу}
if Length(Edit1.Text)> 0 then
if not DirectoryExists(Edit1.Text) then
MkDir(Edit1.Text);
{Далі йде створення для кожного URL в списку свого процесу}
for i := 0 to ListBox1.Items.Count-1 do
  begin
     with THTTPThread.Create(True) do begin
{Створюємо припинену задачу, указуємо їй її URL і запускаємо її}
     URL := ListBox1.Items[i];
     Resume;
end;
end;
end;

{Оператори процесу THTTPThread}
procedure THTTPThread.Execute;
begin
{Робимо так, щоб кожний процес виконувався
одночасно з іншими (синхронізація)}
Synchronize(DoWork);
end;

procedure THTTPThread.DoWork;
    var
i: Integer;
begin
{Створюємо компонент TNMHTTP}
FHTTP := TNMHTTP.Create(Form1);

{Результат записуємо у файли}
FHTTP.InputFileMode := True;
{Підбираємо імена для файлів}
i:= 1;
while FileExists(Form1.Edit1.Text+'\page'+IntToStr(i)+'.htm') do
Inc(i);
```

```
        {Указуємо, в які саме файли класти результат}
        FHTTP.Body := Form1.Edit1.Text+'\body'+IntToStr(i)+'.htm';
        FHTTP.Header := Form1.Edit1.Text+'\header'+IntToStr(i)+'.txt';
        {Намагаємося послати запит}
        FHTTP.Get(URL);
        FHTTP.Free;
end;
```

Оглянемо реалізацію програмного опитування користувачів.

При роботі розробленого програмного продукту в мережі в деяких випадках необхідно знати поточний стан як локального, так і видалених хостов (чи має локальний хост в даний момент можливість виходу в мережу Інтеренет, чи доступний якийсь видалений хост і т.д.)

Загальновідомо, що для вказаної мети використовується утиліта ping. Принцип роботи ping-а заснований на використовуванні протоколу ICMP – Internet Control Message Protocol (протокол керівників, або контрольних, повідомлень).

За допомогою ICMP хост в мережі обмінюються різною службовою інформацією (інформацією про зміну маршруту, зменшення швидкості передачі, неприступність якої-небудь адреси і т.д.)

В основі протоколу ICMP лежить поняття повідомлень. Повідомлення ICMP протоколу, як правило, оповіщають про помилки, що виникають при обробці датаграмм. ICMP використовує основні властивості протоколу IP, неначебто він був протоколом більш високого рівня. На самій же справі ICMP є складовою частиною IP.

Одним з типів повідомлень протоколу є ехо-запит. Отримавши ехо-запит хост зобов'язаний відповісти тому, що послав "ехо-відповіддю".

По суті, "ехо-запит" та "ехо-відповідь" відрізняються лише адресами відправника і одержувача і кодом типу повідомлення (тип 8 – "лунає-запит, тип 0 – "лунає-відповідь).

Реалізації утиліти ping на різних платформах істотно відрізняються. Так, в Ос UNIX використовуються RAW sockets (необроблені, "сирі" сокети), а в Ос Windows всіх версій – т.з. ICMP API.

На практиці, у всіх версіях Windows є і використовується бібліотека icmp.dll. Отже, на даний момент, можна її використовувати що було зроблено.

Визначення типів і прототипи функцій для Microsoft icmp.dll:

```
type
// Інформація заголовка IP (Наповнення)
ip_option_information = packed record
Ttl : byte;
// Час життя (використовується traceroute-ом)
Tos : byte;
// Тип обслуговування, звичайно 0
Flags : byte;
// Прапори заголовка IP, звичайно 0
// Розмір даних в заголовку, звичайно 0, максимум 40
OptionsSize : byte;
OptionsData : Pointer;
// Покажчик на дані
end;
icmp_echo_reply = packed record
        Address : u_long;
// Адреса відповідаючого
        Status : u_long;
// IP_STATUS
// Час між луна-запитом і луна-відповіддю
        RTTime : u_long;
// в мілісекундах
        DataSize : u_short;
// Розмір повернених даних
        Reserved : u_short;
// Зарезервовано
        Data : Pointer;
// Покажчик на повернені дані
// Інформація із заголовка IP
        Options : ip_option_information;
    end;
    PIPINFO = ^ip_option_information;
    PVOID = Pointer;
function IcmpCreateFile():THandle;stdcall;external 'ICMP.DLL';
function IcmpCloseHandle(IcmpHandle : THandle): BOOL; stdcall;
         external 'ICMP.DLL'  name 'IcmpCloseHandle';
function IcmpSendEcho(IcmpHandle : THandle;
// Адреса одержувача (в мережному порядку)
```

```
DestAddress : u_long;
RequestData : PVOID;
// Покажчик на посилані дані
RequestSize : Word;
// Розмір посиланих даних
RequestOptns : PIPINFO;
// Покажчик на посилану структуру
//ip_option_information (може бути nil)
//Покажчик на буфер, що містить відповіді.
ReplyBuffer:PVOID;
ReplySize : DWORD;
//Розмір буфера відповідей
Timeout : DWORD
//Час очікування відповіді в мілісекундах
): DWORD; stdcall; external 'ICMP.DLL' name 'IcmpSendEcho';
```

Функція IcmpSendEcho() посилає ICMP ехо-запит за заданою IP адресою і поміщає всі відповіді, отримані за час заданого таймауту, в буфер відповідей (ReplyBuffer).

Перед використовуванням функцій Icmp.dll необхідно викликати функцію WSAStartup() з Winsock.

Якщо цього не зробити, то перший же виклик функції IcmpSendEcho() приведе до помилки 10091 (WSASYSNOTREADY).

Слід помітити, що відповіді, поміщені в буфер, необов'язково будуть повідомленнями ехо-відповідь. Можливо, що у відповідь на ехо-запит прийдуть повідомлення ICMP про виниклі помилки.

Природно, ці повідомлення так само будуть поміщені в ReplyBuffer. Розмір буфера відповідей повинен бути достатнім для прийому хоча б однієї відповіді, будь то ехо-відповідь або повідомлення про помилку.

Звідси, вказаний розмір складається з розміру самої структури icmp_echo_reply плюс Max(RequestSize, 8), оскільки повідомлення ICMP про помилку складає 8 байт.

Функція IcmpSendEcho() повертає кількість відповідей (або структур icmp_echo_reply) в масиві ReplyBuffer. Якщо функція повернула 0, то для більш

детальної діагностики виниклої помилки використовується функція GetLastError().

Нижче приведений базовий фрагмент коду, необхідний для однократного посилання ехо-запиту:

```
procedure TForm1.Button1Click(Sender: TObject);
var
    hIP : THandle;
    pingBuffer : array [0..31] Char;
    pIpe : ^icmp_echo_reply;
    pHostEn : PHostEnt;
    wVersionRequested : WORD;
    lwsaData : WSAData;
    error : DWORD;
    destAddress : In_Addr;
begin
// Створюємо handle
    hIP := IcmpCreateFile();
GetMem( pIpe, sizeof(icmp_echo_reply)+ sizeof(pingBuffer));
    pIpe.Data := @pingBuffer;
    pIpe.DataSize := sizeof(pingBuffer);
    wVersionRequested := MakeWord(1,1);
    error := WSAStartup(wVersionRequested,lwsaData);
    if (error <> 0) then
    begin
Memo1.SetTextBuf('Error in call to '+'WSAStartup().');
Memo1.Lines.Add('Error code: '+IntToStr(error));
    Exit;
    end;
        pHostEn := gethostbyname;
    error := GetLastError();
    if (error <> 0) then
    begin
Memo1.SetTextBuf('Error in call to'+'gethostbyname().');
Memo1.Lines.Add('Error code: '+IntToStr(error));
    Exit;
    end;
destAddress := PInAddr(pHostEn^.h_addr_list^)^;
// Посилаємо ping-пакет
Memo1.Lines.Add('Pinging'+pHostEn^.h_name+' ['+
                inet_ntoa(destAddress)+'] '+ ' with '+
```

```
IntToStr(sizeof(pingBuffer))+ 'bytes data:');
IcmpSendEcho(hIPdestAddress.S_addr, pingBuffer
             sizeof(pingBuffer), Nil, pIpe sizeof(icmp_echo_reply)+
             sizeof(pingBuffer), 5000);
error := GetLastError();
    if (error <> 0) then
    begin
          Memo1.SetTextBuf('Error in call to '+'IcmpSendEcho()');
          Memo1.Lines.Add('Error code: '+IntToStr(error));
    Exit;
    end;
// Дивимося деякі з даних, що повернулися
Memo1.Lines.Add('Reply from '+ IntToStr(LoByte(LoWord(pIpe^.Address)))+'.'+
IntToStr(HiByte(LoWord(pIpe^.Address)))+'.'+
IntToStr(LoByte(HiWord(pIpe^.Address)))+'.'+
IntToStr(HiByte(HiWord(pIpe^.Address))));
Memo1.Lines.Add('Reply time:'+IntToStr(pIpe.RTTime)+' ms');
    IcmpCloseHandle(hIP);
    WSACleanup();
    FreeMem(pIpe);
end;
```

Для роботи цього фрагмента коду необхідно:

– створити форму TForm1;

– включити в розділ Uses юніт WinSock;

– помістити на форму компонент Memo1:TMemo і кнопку Button1:TButton;

– в розділі Implementation помістити описи типів і функцій Icmp.dll вище);

– зіставити події OnClick кнопки приведену функцію.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму Md5. Він отримує на вході повідомлення довільної довжини і створює на виході

дайджест повідомлення довжиною 128 біт. Алгоритм складається з наступних кроків:

1. Додавання недостаючих біт. Повідомлення доповнюється так, щоб його довжина стала рівна 448 по модулю 512 (довжина 448 mod 512). Це означає, що довжина доданого повідомлення на 64 біта менше, ніж число, кратне 512. Додавання проводиться завжди, навіть якщо повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 біт, воно доповнюється 512 бітами до 960 біт. Таким чином, число біт, що додаються, знаходиться в діапазоні від 1 до 512.

Додавання складається з одиниці, за якою слідує необхідна кількість нулів.

2. Додавання довжини. 64-бітове представлення довжини початкового (до додавання) повідомлення в бітах приєднується до результату першого кроку. Якщо первинна довжина більша, ніж 264, то використовуються тільки останні 64 біта. Таким чином, поле містить довжину початкового повідомлення по модулю 264.

В результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам. Це розширене повідомлення представляється як послідовність 512-бітових блоків Y_0, Y_1,..., Y_{l-1}, при цьому загальна довжина розширеного повідомлення рівна L*512 бітам. Таким чином, довжина отриманого розширеного повідомлення кратна шістнадцяти 32-бітовим словам.

3. Ініціалізація MD-буфера. У алгоритмі Md5 використовується 128-бітовий буфер для зберігання проміжних і остаточних результатів хеш-функції. Буфер може бути представлений як чотири 32-бітові регістри (*A*, *B*, *C*, *D*). Ці регістри ініціалізувалися наступними шістнадцятковими числами:

A = 01234567

B = 89abcdef

C = Fedcba98

D = 76543210

4. Обробка послідовності 512-бітових (16-словних) блоків. Основою алгоритму Md5 є модуль, що складається з чотирьох циклічних обробок, позначений як Hmd5. Чотири цикли мають схожу структуру, але кожен цикл використовує свою елементарну логічну функцію, *ff*, що позначається, *fg*, *fh* і *fi* відповідно.

Кожен цикл приймає на вхід поточний 512-бітовий блок Y_q, що обробляється в даний момент, і 128-бітове значення буфера ABCD, яке є проміжним значенням дайджесту, і змінює вміст цього буфера. Кожен цикл також використовує четвертую частина 64-елементної таблиці *T*[1 . 64], побудованої на основі функції sin. *i*-ий елемент *T*, *T*[i], що позначається, має значення, рівне цілій частині від 232 * abs (sin (*i*)), і задане в радіанах. Оскільки abs (sin (*i*)) є числом між 0 і 1, кожен елемент *T* є цілим, яке може бути представлене 32 бітами. Таблиця забезпечує "випадковий" набір 32-бітових значень, які повинні ліквідовувати будь-яку регулярність у вхідних даних.

Для отримання *Mdq*+1 вихід чотирьох циклів складається по модулю 232 з *Mdq*. Складання виконується незалежно для кожного з чотирьох слів в буфері.

5) Вихід Md5. Після обробки всіх *L* 512-бітових блоків виходом *L*-ої стадії є 128-бітовий дайджест повідомлення.

Детальніше логіку кожного з чотирьох циклів виконання одного 512-бітового блоку розглянуто нижче. Кожен цикл складається з 16 кроків, що оперують з буфером *ABCD*.

$$A \quad B + \text{Clss}\,(A + f\,(B, C, D) + X\,[k] + T\,[i]),$$

A, *B*, *C*, *D* – чотири слова буфера; після виконання кожного окремого кроку відбувається циклічне зрушення вліво на одне слово.

f – одна з елементарних функцій *ff*, *fg*, *fh*, *fi*.

CLSs – циклічне зрушення вліво на s біт 32-бітового аргументу.

$X\ [k] - M\ [q * 16 + k]$ – кодує 32-бітове слово в *q*-ому 512 блоці повідомлення.

T [*i*] – *i*-е 32-бітове слово в матриці T.

+ – складання по модулю 232.

На кожному з чотирьох циклів алгоритму використовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-бітові слова на вході і на виході створює одне 32-бітове слово. Кожна функція є безліччю побітових логічних операцій, тобто *n*-ий біт виходу є функцією від n-ого біта трьох входів. Елементарні функції наступні:

$$ff = (B \ \& \ C) \quad (\text{not } B \ \& \ D)$$

$$fg = (B \ \& \ D) \ \text{V} \ (C \ \& \ \text{not } D)$$

$$fh = B \quad C \quad D$$

$$fi = C \quad (B \ \& \ \text{not } D)$$

Масив з 32-бітових слів *X* [0..15] містить значення поточного 512-бітового вхідного блоку, який обробляється зараз. Кожен цикл виконується 16 разів, а оскільки кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожен блок вхідного повідомлення обробляється по схемі 64 рази. Якщо представити вхідний 512-бітовий блок у вигляді шістнадцяти 32-бітових слів, то кожне вхідне 32-бітове слово використовується чотири рази, по одному разу в кожному циклі, і кожен елемент таблиці T, що складається з 64 32-бітових слів, використовується тільки один раз. Після кожного кроку циклу відбувається циклічне зрушення вліво чотирьох слів *A*, *B*, *C* і *D*. На кожному кроці змінюється тільки одне з чотирьох слів буфера *ABCD*. Отже, кожне слово буфера змінюється 16 разів, і потім 17-й раз в кінці для отримання остаточного виходу даного блоку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Файл; Дані; Додатки; Налаштування; Довідка.

– Підвікно обчислювальні клієнти: Назва; IP адреса; Порт TCP/IP; Список знайдених ОК; Перегляд результатів роботи обчислювальних клієнтів.

– Функції: Запуск обчислювальної системи; Сканування доступу до обчислювальних клієнтів; Встановлення розрахункового скрипту; Перегляд розрахункових скриптів; Налаштування ПЗ; Налаштування доступу до обчислювальних клієнтів.

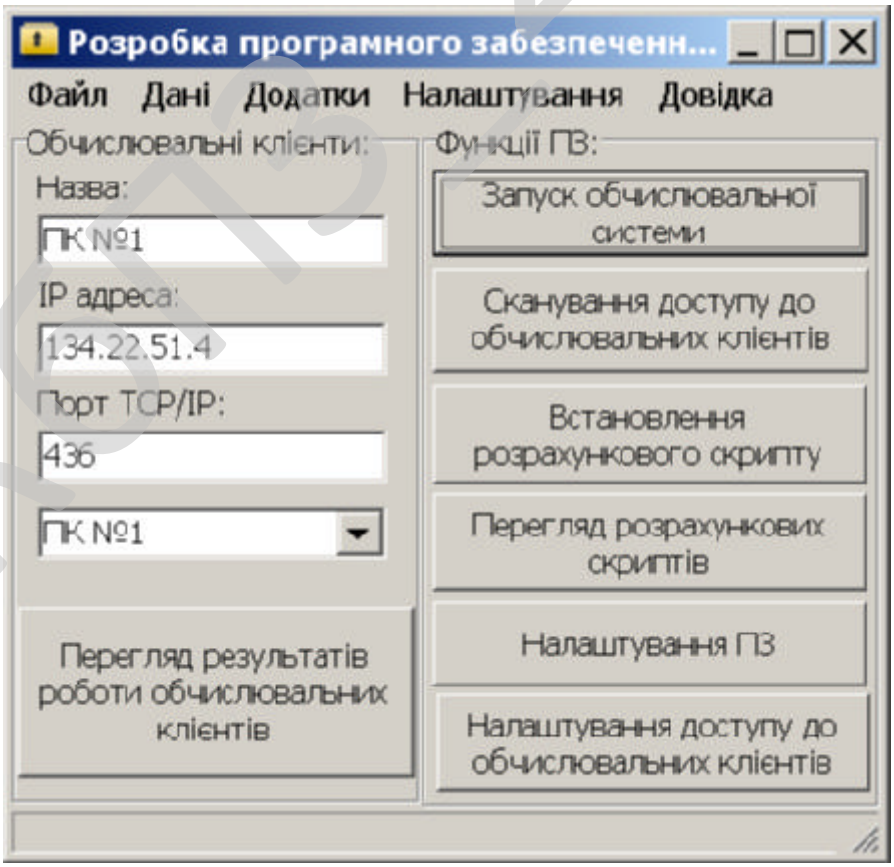

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права. Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (дискету чи CD-ROM).

Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

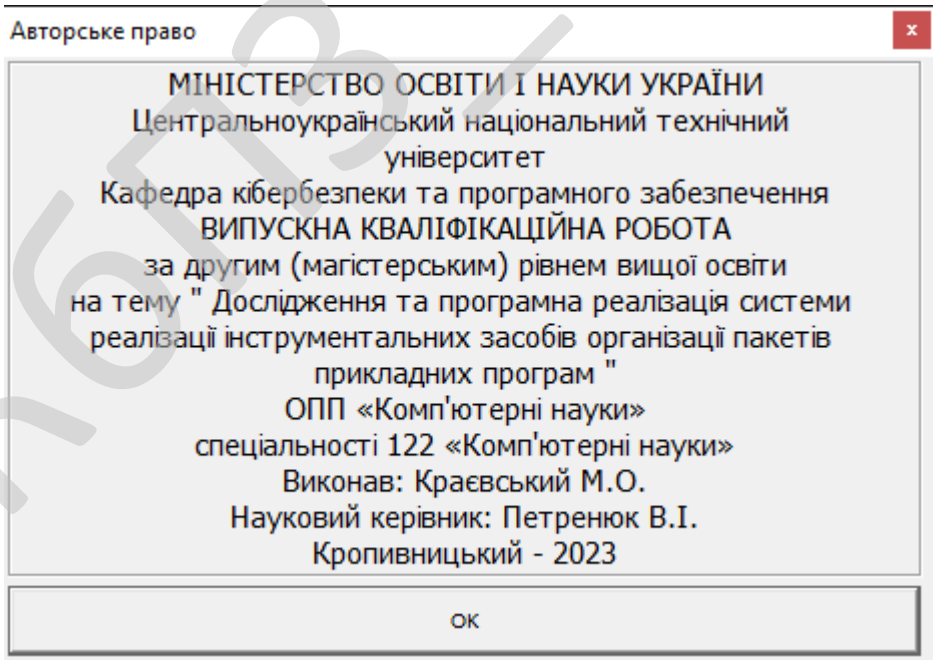

Рисунок 5.2 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи реалізації інструментальних засобів організації пакетів прикладних програм.

Метою розробки є дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

Об'єктом дослідження є процес реалізації інструментальних засобів організації пакетів прикладних програм.

Предметом дослідження є методи реалізації інструментальних засобів організації пакетів прикладних програм.

Методи дослідження базуються на методах інженерії програмного забезпечення, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод реалізації інструментальних засобів організації пакетів прикладних програм.

– Розроблено вітчизняний продукт реалізації інструментальних засобів організації пакетів прикладних програм, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові данні

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	40
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B , \qquad (7.1)$$

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot75=126 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	126	Ф 7.1-7.4
Впровадження	13	Д13
Всього	167	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{167 \cdot 1}{60\text{-}5} = 3 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	9	810	13,5
Монітор	60	9	540	9
Клавіатура	30	9	270	4,5
Маніпулятор «мишка»	30	9	270	4,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор– маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м. п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	48,49

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{49 \cdot 3}{1,2} = 122,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=122,5/(60·8)=0,25 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів – електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів д мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	0,25	10000	7500
Продакт-менеджер	0,25	10000	7500
Інженер-програміст	3	10000	90000
Інженер-електронщик	0,25	10000	7500
Інженер-системотехнік	0,25	10000	7500
Адміністратор мережі	0,5	10000	15000
Системний програміст	0,25	10000	7500
Дизайнер WEB	0,25	10000	7500
Інженер-верстальник	0,25	10000	7500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$=5,75	-	$Ф_{роб}$=172500

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{172500}{5,75 \cdot 60} = 500 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. $S_у$ – питома площа на одне робоче місце, $м^2$; $Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 28.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок	Fujitsu Esprimo P757 E90+ Tower	7347
Процесор	Intel Core i5-6500 (4 ядра по 3.2 – 3.6 GHz), 6 MB Smart Cache	-
Системна плата	Intel Q250 Chipset	-
Відеокарта	nVidia Quadro 2000, 1 GB GDDR5 128-bit	-
Жорсткий диск	240 GB SSD	-
Оперативна пам’ять	16 GB DDR4 (два модулі по 8 GB)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	Fujitsu Esprimo D757 E85+ SFF, 4x USB 2.0, 6x USB 3.0, 2x VGA, 1x DVI, 2x DisplayPort, 2x PS/2, 4x Audio, 1x LAN (RJ-45), 280W	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
Клавіатура, мишка	Wireless Keyboard and Mouse X1800S	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG 24MB37PM / 24" (1920x1080) IPS / DVI, VGA, Audio / 2x 1W / VESA 100x100 / Pivot	3600
Принтер лазерний	Samsung ML-3750ND /1200x1200 dpi / A4 / 35 стр./мин / USB 2.0 /	2700
Принтер струменевий	EPSON XP 345 / кольоровий струменевий друк / A4 / 5760x1440 / до 33 стр./хв. / фотодрук / сканер	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість ,шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 500 \cdot 167/40 = 2090 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 506 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м \, n_{міс}. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1 \cdot 0{,}5 = 105 \text{ грн}.$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д \,, \tag{7.17}$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 49,2 грн./шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \tag{7.18}$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Samsung ML-3750ND – 574 грн.; картридж для EPSON XP 345 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_М = (105 + 492 + 1702)/40=57 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом (Нп = 15%) від основної зарплати виконавців

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_п = 2090 \cdot 15 \cdot 0{,}01=314 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м . \quad (7.21)$$

$$C_п = 2090+209+506+314+57+314+876 = 4366 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot P_п \cdot C_п , \quad (7.22)$$

де $P_п$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 4366 = 2401 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	506
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4366
9. Плановий прибуток	$П_р$	2401
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	6767
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1353,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9168

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	9168
Всього капітальних витрат	–	9168

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	120780	26840
2. Витрати на електроенергію	$З_{ел}$	2099	1049
3. Витрати на амортизацію	$З_{ам}$	0	4584
Всього витрат за рік	$І$	122879	32473

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*; $З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 900 годин на рік до 200 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 900 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 120780\ грн.$$

до

$$З_{р\ нов} = 200 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 26840\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел}=П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз=·0,475·2455·1,8 = 2099 грн

Зел нов=·0,475·1227·1,8 = 1049 грн

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	9168	–	4584
Всього відрахувань	-	–	9168	–	4584

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (6767 - 4366) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 61005 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{188075}{(6767 - 4366) \cdot 40 \cdot 12 / 3} = 0,5 \; років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (122879 - 32473) - 0,5 \cdot 9168 = 85822 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{9168}{122879 - 32473} = 0{,}1 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	4366
3. Ціна розробленої програми	Грн.	6767
4. Плановий прибуток від реалізації розробленої програми	Грн.	2401
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	96040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	61005
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9168
11.Величина економічного ефекту у користувача програмної продукції	Грн.	85822
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Наявний в даний час в нашій країні комплекс розроблених організаційних заходів та технічних засобів захисту, накопичений передовий досвід роботи ряду обчислювальних центрів показує, що є можливість домогтися значно більших успіхів у справі усунення впливу на працюючих небезпечних і шкідливих виробничих факторів. Проте стан умов праці та його безпеки в ряді обчислювальних центрів (ОЦ) та підприємств ще не задовольняють сучасним вимогам. Оператори ЕОМ, оператори підготовці даних, програмісти та інші працівники ОЦ та підприємств ще стикаються з впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика і інші.

Багато працівників ОЦ та підприємств пов'язані з впливом таких психофізичних факторів, як розумова перенапруга, перенапруження зорових і слухових аналізаторів, монотонність праці, емоційні перевантаження. Вплив зазначених несприятливих факторів призводить до зниження працездатності, викликане розвиваються втомою. Поява і розвиток втоми пов'язане зі змінами, які виникають під час роботи в центральній нервовій системі, з гальмівними процесами в корі головного мозку. Наприклад сильний шум викликає труднощі з розпізнаванням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5 – 12% продуктивність праці. Тривала дія шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30 – 60%.

Медичні обстеження працівників ОЦ та підприємств показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення

слуху. Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Аналіз травматизму серед працівників ВЦ показує, що в основному нещасні випадки відбуваються від впливу фізично небезпечних виробничих факторів при заправці носія інформації на обертовий барабан при знятому кожусі, під час співробітниками невластивих їм робіт. На другому місці випадки, пов'язані з дією електричного струму.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному

і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці

може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [2]

8.3 Пропозиції щодо підвищення працездатності ІТ – фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців it-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців it-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців it-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Для більшого розуміння, пропозиції щодо підвищення працездатності it-фахівців, розіб'ємо на декілька категорій:

1. Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням it-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють it-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання,

переохолодження it-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці it-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

2. Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність it-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві it-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна

пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються ветикальні електроди: метелевий куток *63·63·6 мм*., (згідно з ДСТУ 2251-93 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною *L=2 м.,* та горизонтальний електрод – метелева полоса з перетином *60·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: $H=0,5$ *м*. Відстань між вертикальними заземлювачами (електродами) $A=2$ *м*. Глибина закладення горизонтального контура заземлення $t=0,6$ *м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+2/2=1,6 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [12];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [12].

Еквівалентний діаметр ветикальньного електрода (кутка) [12]:

$$Dв=0,95 \cdot K = 0,95 \cdot 63 = 59,85 \text{ мм.} = 0,0598 \text{ м.}$$

де $K = 63$ *мм.* – розмір метелевого кутка (задан).

Відношення $A/L=2/2=1$

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [12]:

$$R_O=0,366(\rho/L)[\lg(2L/Dв)+(1/2)\lg((4T+L)/(4T-L))] =$$

$$=0,366(54,5/2)[(\lg(2 \cdot 2/0,0598)+(1/2)\lg((4 \cdot 1,6+2)/(4 \cdot 1,6-2))] =$$

$$=19,6 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів $Кев=0,62$ при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [12].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без вхарування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_O /(Кев\ R_{ЗН})= 19,6 / (0,62\cdot4)= 7,8 \approx 8\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2\cdot8=16,6\approx 16\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [12]:

$$R_П =0,366(\rho\cdot K_П/L_П)lg(2\cdot L_П^2 /(B\cdot t))=$$

$$=0,366(40\cdot5/16)\cdot lg((2\cdot16^2)/(0,06\cdot0,6))=20,2\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [12]:

$B = 60\ мм. = 0,06\ м.$- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [12]:

$$R =(R_O\cdot R_П)/(R_O\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$

$$=(19,6\cdot 20,2)/(19,6\cdot 0,6+ 8\cdot 20,2\cdot 0,62)=3,53\ Ом.$$

де $\eta_П = 0,6$ – табличне значення коефіцієнта екранування з'єднуючої полоси [12].

Умова $R \le R_{ЗН}$ виконується ($3,53 \le 4$).

Остаточно отримали: кількість вертикальних електродів дорівнює *8* при $R = 3,53\ Ом$.

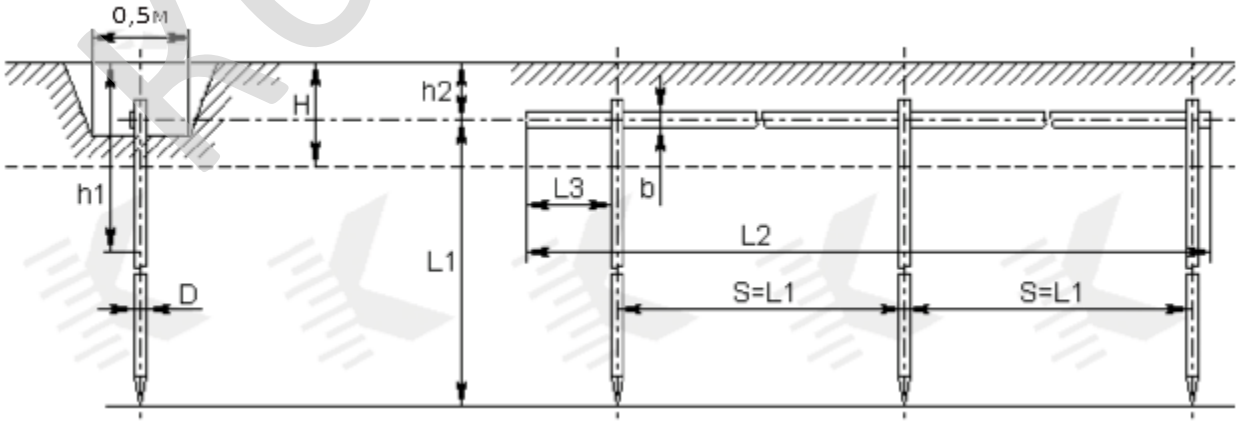

Рисунок 8.1 – Схема штучного заземлення

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи реалізації інструментальних засобів організації пакетів прикладних програм.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів реалізації інструментальних засобів організації пакетів прикладних програм.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем реалізації інструментальних засобів організації пакетів прикладних програм.

– Досліджена система реалізації інструментальних засобів організації пакетів прикладних програм.

– На основі отриманих результатів досліджень створена програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання реалізації інструментальних засобів організації пакетів прикладних програм.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Md5.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 85822 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 рік.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Краєвський М.О. Дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Fernando Doglio. Skills of a Successful Software Engineer. Manning. 2022. 182 c.
3. M. Holmes He. Creating Apps with React Native. Apress Media. 2022. 445 p.
4. Maurício Aniche. Effective Software Testing. Manning Publications. 2021. 372 p
5. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.
6. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.
7. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.
8. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.
9. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.
10. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.
11. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.
12. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

13. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

14. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

15. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

16. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

17. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

18. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

19. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

20. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

21. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

22. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

23. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

24. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

25. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

26. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

27. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

28. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

29. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

30. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

31. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

32. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

33. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

34. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

35. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду

абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

36. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

37. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

38. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

39. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

40. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

41. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

42. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

43. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

44. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

45. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

46. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

47. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

48. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA – 2017.

49. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп'ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

51. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

52. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). – Харків: ХУПС. – 2016. – С.96-100.

53. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". – Випуск 8 (145). – Х.: ХУПС – 2016. – С. 77-80.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи реалізації інструментальних засобів організації пакетів прикладних програм.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 33-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи реалізації інструментальних засобів організації пакетів прикладних програм;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці ІТ-фахівця.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 113 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 11.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Петренюк В.І.

Дослідження та програмна реалізація системи реалізації інструментальних засобів організації пакетів прикладних програм

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 43

Літера: РП

Кропивницький – 2023 року

Файл проекту ПЗ PZ.dpr

```
program PZ;

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Лістинг ПЗ до магістерської роботи на тему:
Дослідження та програмна реалізація системи
реалізації інструментальних засобів організації
пакетів прикладних програм
Виконав: студент  Краєвський Микола Олегович
2023 рік
}
Uses
// Підключення бібліотек
  Forms,
  Unit1 in 'Mr1.pas' {Form1},
// Файл вікна ПЗ Form1
  Unit2 in 'Mr2.pas' {Form2},
// Файл вікна ПЗ Form2
  Unit3 in 'Mr3.pas' {Form3};
// Файл вікна ПЗ Form3
  Unit4 in 'Mr4.pas' {Form4};
// Файл вікна ПЗ Form4
  Unit5 in 'Mr5.pas' {Form5};
// Файл вікна ПЗ Form5
{$R *.res}
// Підключення ресурсів

begin
  Application.Initialize;
//Ініціалізіція ПЗ
  Application.CreateForm(TForm1, Form1);
// Підключення вікна 1
  Application.CreateForm(TForm2, Form2);
// Підключення вікна 2
  Application.CreateForm(TForm3, Form3);
// Підключення вікна 3
  Application.CreateForm(TForm4, Form4);
// Підключення вікна 4
  Application.CreateForm(TForm5, Form5);
// Підключення вікна 5
  Application.Run;
// Виведення головного вікна ПЗ (Form1)
end.
// Кінець файлу
```

Файл першого вікна ПЗ (Mr1.pas)

```
unit Mr1.pas; // Початок файлу

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Лістинг ПЗ до магістерської роботи на тему:
Дослідження та програмна реалізація системи
реалізації інструментальних засобів організації
пакетів прикладних програм
Виконав: студент  Краєвський Микола Олегович
2023 рік
}

interface // Секція інтерфейсної частини файлу

Uses // Підключення бібліотек
  Classes, SysUtils, Windows, ShellApi;

Type // Визначення типів
  IDiskHugeNumber = interface
    ['{C540F255-F373-4A20-AF1E-2A3C03BFDFDE}']
    function AsNumber: Int64;
    function Formatted: String;
  end;
  TDiskHugeNumber = class(TInterfacedObject, IDiskHugeNumber)
  private
    fValue: Int64;
  public
    constructor Create(Value: Int64);
    destructor Destroy; override;

    function AsNumber: Int64;
    function Formatted: String;
  end;

  IDiskBoolean = interface
    ['{D6669ECF-6FB5-4854-AD86-04CEDA6B6692}']
    function AsBoolean: Boolean;
    function FormatYesNo: String;
    function FormatOnOff: String;
  end;
  TDiskBoolean = class(TInterfacedObject, IDiskBoolean)
  // Визначення класу від TInterfacedObject, IDiskBoolean
  private
    fValue: Boolean;
  public
    constructor Create(Value: Boolean);
    destructor Destroy; override;

    function AsBoolean: Boolean;
    function FormatYesNo: String;
    function FormatOnOff: String;
  end;
//--------------------------------
Type // Визначення типів
  TSystemRVSTypeEnum = (dtUnknown, dtNoRoot, dtRemovable, dtFixed, dtRemote,
dtCdRom, dtRam);

  ISystemRVSType = interface
    ['{7279EE2B-5B97-41FF-B793-4DD5A0C39783}']
    function SystemRVSType: TSystemRVSTypeEnum;
    function Name: String;
```

```
  end;
  TSystemRVSType = class(TInterfacedObject, ISystemRVSType)
  // Визначення класу від TInterfacedObject, ISystemRVSType
  private
    fSystemRVS: Integer;
    function SystemRVSPath: String;
  public
    constructor Create(SystemRVS: Integer);
    destructor Destroy; override;
    function SystemRVSType: TSystemRVSTypeEnum;
    function Name: String;
  end;

const //введення констант
cSystemRVSTypeMap: array[dtUnknown..dtRam] of String = (RsDisk_Type_Unknown,
RsDisk_Type_NoRoot,
RsDisk_Type_Removable, RsDisk_Type_Fixed,
RsDisk_Type_Remote, RsDisk_Type_CdRom,RsDisk_Type_Ram);

//----------------------------------------
// Реалізація пошуку
//----------------------------------------

Type // Визначення типів
  ISystemRVSFeatures = interface
    ['{E94E88F0-74D0-4DB2-8737-28F76D32A2B6}']
    function NamedStreams: IDiskBoolean;
    function ReadOnly: IDiskBoolean;
    function ObjectIds: IDiskBoolean;
    function ReparsePoints: IDiskBoolean;
    function SparseFiles: IDiskBoolean;
    function DiskQuotas: IDiskBoolean;
    function CasePreserved: IDiskBoolean;
    function CaseSensitive: IDiskBoolean;
    function FileCompression: IDiskBoolean;
    function FileEncryption: IDiskBoolean;
    function PersistentAcl: IDiskBoolean;
    function UnicodeOnDisk: IDiskBoolean;
    function Compressed: IDiskBoolean;
  end;
  TSystemRVSFeatures = class(TInterfacedObject, ISystemRVSFeatures)
  // Визначення класу від TInterfacedObject, ISystemRVSFeatures
  private
    fFeatures: DWord;
  public
    constructor Create(Features: DWord);
    destructor Destroy; override;
    function FileCompression: IDiskBoolean;
    function FileEncryption: IDiskBoolean;
    function PersistentAcl: IDiskBoolean;
  end;

Type // Визначення типів
  ISystemRVSSpace = interface
    ['{4FC33F5D-8C76-4F5F-AAD6-EAE1C51E8D29}']
    function BytesAvailable: IDiskHugeNumber;
    function BytesTotal: IDiskHugeNumber;
    function BytesFree: IDiskHugeNumber;
    function BytesUsed: IDiskHugeNumber;
  end;
  TSystemRVSSpace = class(TInterfacedObject, ISystemRVSSpace)
  // Визначення класу від TInterfacedObject, ISystemRVSSpace
  private
    fSystemRVS: Integer;
```

```
    fBytesAvailable: Int64;
    fTotalBytes: Int64;
    fTotalFreeBytes: Int64;

    procedure GetSystemRVSSpaceInfo;
  public
    constructor Create(SystemRVS: Integer);
    destructor Destroy; override;

    function BytesAvailable: IDiskHugeNumber;
    function BytesTotal: IDiskHugeNumber;
    function BytesFree: IDiskHugeNumber;
    function BytesUsed: IDiskHugeNumber;
  end;

  ISystemRVSShellInfo = interface
    ['{CF04F655-B982-4B06-8216-2719E29D9DDB}']
    function Icon: HIcon;
    function Image: Integer;
    function DisplayName: String;
    function TypeName: String;
  end;

  TSystemRVSShellInfo = class(TInterfacedObject, ISystemRVSShellInfo)
  // Визначення класу від TInterfacedObject, ISystemRVSShellInfo
  private
    fSystemRVS: Integer;
    fShellSystemRVSInfo: TSHFileInfo;

    procedure GetSystemRVSShellInfo;
  public
    constructor Create(SystemRVS: Integer);
    destructor Destroy; override;

    function Icon: HIcon;
    function Image: Integer;
    function DisplayName: String;
    function TypeName: String;
  end;

Type // Визначення типів
  TVolumeInformation = record
    Index: Integer;
    RootPathName: String;
    DisplayName: String;
    VolumeNameBuffer: String;
    VolumeSerialNumber: DWord;
    MaximumComponentLength: DWord;
    FileSystemFlags: DWord;
    FileSystemName: String;
  end;

  ISystemRVSInfo = interface
    ['{4300B119-01C0-4140-95F5-521ED5F7BC9B}']
    function Available: IDiskBoolean;
    function SystemRVSType: ISystemRVSType;
    function Index: Integer;
    function Letter: String;
    function VolumeLabel: String;
    function Features: ISystemRVSFeatures;
    function Space: ISystemRVSSpace;
    function Serial: String;
    function Shell: ISystemRVSShellInfo;
    function FileSystem: String;
    procedure ShowShellDialog;
```

```
  end;
  TSystemRVSInfo = class(TInterfacedObject, ISystemRVSInfo)
  // Визначення класу від TInterfacedObject, ISystemRVSInfo
  private
    fSystemRVS: Integer;
    fSystemRVSInfo: TVolumeInformation;

    procedure GetSystemRVSInfo;
  public
    constructor Create(SystemRVS: Integer);
    destructor Destroy; override;

    function Available: IDiskBoolean;
    function SystemRVSType: ISystemRVSType;
    function Index: Integer;
    function Letter: String;
    function VolumeLabel: String;
    function Features: ISystemRVSFeatures;
    function Space: ISystemRVSSpace;
    function Serial: String;
    function Shell: ISystemRVSShellInfo;
    function FileSystem: String;
    procedure ShowShellDialog;
  end;


//----------------------------------------
// Обєкт пошуку
//----------------------------------------

Const //введення констант
cDisk_SystemRVSPath_Fmt = '%s:\';
// Букви дисків
cSystemRVSLetters: array[0..25] of char =
('A','B','C','D','E','F','G','H','I','J','K','L','M',
'N','O','P','Q','R','S','T','U','V','W','X','Y','Z');

Type // Визначення типів
  TcxSystemRVSInfo = class  // Визначення класу
  private
    function GetSystemRVSInfoByIndex(Index: Integer): ISystemRVSInfo;
  public
    function Version: String;

    function ByLetter(Letter: String): ISystemRVSInfo;
    property SystemRVSs[Index: Integer]: ISystemRVSInfo read
GetSystemRVSInfoByIndex; default;
  end;

var
  cxSystemRVS: TcxSystemRVSInfo;

implementation // Секція реалізації

{ Реалізація класу -  TMRDiskHugeNumber }

function TDiskHugeNumber.AsNumber: Int64;
begin
  Result:= fValue;
end;

constructor TDiskHugeNumber.Create(Value: Int64);
begin
  inherited Create;
  fValue:= Value;
```

```
end;

destructor TDiskHugeNumber.Destroy;
begin
  inherited;
end;

function TDiskHugeNumber.Formatted: String;
const //введення констант
  Kbyte = 1024;
  MByte = 1048576;
  GByte = 1073741824;
begin
  if fValue < KByte then
    Result:= Format(RsDisk_Format_Bytes, [fValue])
  else
    if fValue < MByte then
      Result:= Format(RsDisk_Format_KBytes, [fValue / KByte])
    else
      if fValue < GByte then
        Result:= Format(RsDisk_Format_MBytes, [fValue / MByte])
      else
        Result:= Format(RsDisk_Format_GBytes, [fValue / GByte]);
end;

{ Реалізація класу -  TMRDiskBoolean }

function TDiskBoolean.AsBoolean: Boolean;
begin
  Result:= fValue;
end;

constructor TDiskBoolean.Create(Value: Boolean);
begin
  inherited Create;
  fValue:= Value;
end;

destructor TDiskBoolean.Destroy;
begin
  inherited;
end;

function TDiskBoolean.FormatOnOff: String;
const //введення констант
cBoolean_Results: array[False..True] of String = (RsDisk_Format_Off,
RsDisk_Format_On);
begin
  Result:= cBoolean_Results[fValue];
end;

function TDiskBoolean.FormatYesNo: String;
const //введення констант
  cBoolean_Results: array[False..True] of String = (RsDisk_Format_No,
RsDisk_Format_Yes);
begin
  Result:= cBoolean_Results[fValue];
end;

{ Реалізація класу -  TMRSystemRVSType }

constructor TSystemRVSType.Create(SystemRVS: Integer);
begin
  inherited Create;
```

```
  fSystemRVS:= SystemRVS;
end;

destructor TSystemRVSType.Destroy;
begin
  inherited;
end;

function TSystemRVSType.SystemRVSPath: String;
begin
  Result:= Format(cDisk_SystemRVSPath_Fmt, [cSystemRVSLetters[fSystemRVS]]);
end;

function TSystemRVSType.SystemRVSType: TSystemRVSTypeEnum;
var
  Path: String;
begin
  Path:= SystemRVSPath;
  Result:= TSystemRVSTypeEnum(GetSystemRVSType(PChar(Path)));
end;

function TSystemRVSType.Name: String;
begin
  Result:= cSystemRVSTypeMap[SystemRVSType];
end;

{ Реалізація класу - TMRSystemRVSFeatures }

function TSystemRVSFeatures.CasePreserved: IDiskBoolean;
begin
  Result:= TDiskBoolean.Create((fFeatures and FS_CASE_IS_PRESERVED) <> 0);
end;

function TSystemRVSFeatures.CaseSensitive: IDiskBoolean;
begin
  Result:= TDiskBoolean.Create((fFeatures and FS_CASE_SENSITIVE) <> 0);
end;

function TSystemRVSFeatures.Compressed: IDiskBoolean;
begin
  Result:= TDiskBoolean.Create((fFeatures and
                          FS_VOL_IS_COMPRESSED) <> 0);
end;

constructor TSystemRVSFeatures.Create(Features: DWord);
begin
  inherited Create;

  fFeatures:= Features;
end;

destructor TSystemRVSFeatures.Destroy;
begin
  inherited;
end;

function TSystemRVSFeatures.DiskQuotas: IDiskBoolean;
begin
Result:=TDiskBoolean.Create((fFeatures and
                            FILE_VOLUME_QUOTAS)<>0);
end;

function TSystemRVSFeatures.FileCompression: IDiskBoolean;
begin
```

```
  Result:= TDiskBoolean.Create((fFeatures and FS_FILE_COMPRESSION)
           <> 0);
end;

function TSystemRVSFeatures.FileEncryption: IDiskBoolean;
begin
  Result:= TDiskBoolean.Create((fFeatures and FS_FILE_ENCRYPTION)
            <> 0);
end;

function TSystemRVSFeatures.NamedStreams: IDiskBoolean;
begin
  Result:=TDiskBoolean.Create((fFeatures and FILE_NAMED_STREAMS)
          <> 0);
end;

function TSystemRVSFeatures.ObjectIds: IDiskBoolean;
begin
  Result:= TDiskBoolean.Create((fFeatures and FILE_SUPPORTS_OBJECT_IDS) <> 0);
end;

function TSystemRVSFeatures.PersistentAcl: IDiskBoolean;
begin
  Result:=TDiskBoolean.Create((fFeatures and FS_PERSISTENT_ACLS)
          <> 0);
end;

function TSystemRVSFeatures.ReadOnly: IDiskBoolean;
begin
  Result:= TDiskBoolean.Create((fFeatures and
           FILE_READ_ONLY_VOLUME) <> 0);
end;

function TSystemRVSFeatures.ReparsePoints: IDiskBoolean;
begin
  Result:=TDiskBoolean.Create((fFeatures and
          FILE_SUPPORTS_REPARSE_POINTS) <> 0);
end;

function TSystemRVSFeatures.SparseFiles: IDiskBoolean;
begin
  Result:=TDiskBoolean.Create((fFeatures and
          FILE_SUPPORTS_SPARSE_FILES) <> 0);
end;

function TSystemRVSFeatures.UnicodeOnDisk: IDiskBoolean;
begin
  Result:=TDiskBoolean.Create((fFeatures and
          FS_UNICODE_STORED_ON_DISK) <> 0);
end;

{ Реалізація класу - TMRSystemRVSSpace }

function TSystemRVSSpace.BytesAvailable: IDiskHugeNumber;
begin
  Result:= TDiskHugeNumber.Create(fBytesAvailable);
end;

function TSystemRVSSpace.BytesFree: IDiskHugeNumber;
begin
  Result:= TDiskHugeNumber.Create(fTotalFreeBytes);
end;

function TSystemRVSSpace.BytesTotal: IDiskHugeNumber;
begin
```

```
  Result:= TDiskHugeNumber.Create(fTotalBytes);
end;

function TSystemRVSSpace.BytesUsed: IDiskHugeNumber;
begin
  Result:= TDiskHugeNumber.Create(fTotalBytes - fTotalFreeBytes);
end;

constructor TSystemRVSSpace.Create(SystemRVS: Integer);
begin
  inherited Create;

  fSystemRVS:= SystemRVS;
  GetSystemRVSSpaceInfo;
end;

destructor TSystemRVSSpace.Destroy;
begin
  inherited;
end;

procedure TSystemRVSSpace.GetSystemRVSSpaceInfo;
var
  Path: String;
begin
  Path:= Format(cDisk_SystemRVSPath_Fmt, [cSystemRVSLetters[fSystemRVS]]);
  //Використання WinApi функції SysUtils
  SysUtils.GetDiskFreeSpaceEx(PChar(Path), fBytesAvailable,
  fTotalBytes, @fTotalFreeBytes)
end;

{ Реалізація класу -  TMRSystemRVSShellInfo }

constructor TSystemRVSShellInfo.Create(SystemRVS: Integer);
begin
  inherited Create;

  fSystemRVS:= SystemRVS;
  GetSystemRVSShellInfo;
end;

destructor TSystemRVSShellInfo.Destroy;
begin
  inherited;
end;

function TSystemRVSShellInfo.DisplayName: String;
begin
  Result:= Trim(String(fShellSystemRVSInfo.szDisplayName));
end;

procedure TSystemRVSShellInfo.GetSystemRVSShellInfo;
var
 Path: String;
begin
  Path:= Format(cDisk_SystemRVSPath_Fmt , [cSystemRVSLetters[fSystemRVS]]);
  ShGetFileInfo(PChar(Path), 0, fShellSystemRVSInfo, SizeOf
  (TSHFileInfo), SHGFI_TYPENAME or SHGFI_DISPLAYNAME or
  SHGFI_SYSICONINDEX);
end;

function TSystemRVSShellInfo.Icon: HIcon;
begin
  Result:= fShellSystemRVSInfo.hIcon;
end;
```

```
function TSystemRVSShellInfo.Image: Integer;
begin
  Result:= fShellSystemRVSInfo.iIcon;
end;

function TSystemRVSShellInfo.TypeName: String;
begin
  Result:= Trim(String(fShellSystemRVSInfo.szDisplayName));
end;

{ Реалізація класу -  TMRSystemRVSInfo }

function TSystemRVSInfo.Available: IDiskBoolean;
begin
Result:= TDiskBoolean.Create(cxSystemRVS[fSystemRVS].SystemRVSType.SystemRVSType
<>dtNoRoot);
end;

constructor TSystemRVSInfo.Create(SystemRVS: Integer);
begin
  inherited Create;

  fSystemRVS:= SystemRVS;
  GetSystemRVSInfo;
end;

destructor TSystemRVSInfo.Destroy;
begin
  inherited;
end;

function TSystemRVSInfo.SystemRVSType: ISystemRVSType;
begin
  Result:= TSystemRVSType.Create(fSystemRVS);
end;

function TSystemRVSInfo.Features: ISystemRVSFeatures;
begin
  Result:= TSystemRVSFeatures.Create(fSystemRVSInfo.FileSystemFlags);
end;

function TSystemRVSInfo.FileSystem: String;
begin
  Result:= Trim(fSystemRVSInfo.FileSystemName);
end;

procedure TSystemRVSInfo.GetSystemRVSInfo;
var
  Path: String;
  VolumeName: array[0..MAX_PATH - 1] of Char;
  VolumeSerial: DWord;
  MaxComponentLength: Cardinal;
  FeatureFlags: DWord;
  FileSystem: array[0..MAX_PATH - 1] of Char;

  ErrorMode: Cardinal;
begin
  Path:= Format(cDisk_SystemRVSPath_Fmt , [cSystemRVSLetters[fSystemRVS]]);

  ErrorMode:= SetErrorMode(SEM_FAILCRITICALERRORS);
  try
if GetVolumeInformation(PChar(Path), VolumeName, MAX_PATH,
    @VolumeSerial, MaxComponentLength, FeatureFlags,
    @FileSystem, MAX_PATH) then
```

```
    begin
      fSystemRVSInfo.Index:= fSystemRVS;
      fSystemRVSInfo.RootPathName:= Path;
      fSystemRVSInfo.VolumeNameBuffer:= Trim(String(VolumeName));
      fSystemRVSInfo.VolumeSerialNumber:= VolumeSerial;
      fSystemRVSInfo.MaximumComponentLength:= MaxComponentLength;
      fSystemRVSInfo.FileSystemFlags:= FeatureFlags;
      fSystemRVSInfo.FileSystemName:= Trim(String(FileSystem));
    end
    else
    begin
      fSystemRVSInfo.Index:= fSystemRVS;
      fSystemRVSInfo.RootPathName:= Path;
      fSystemRVSInfo.VolumeNameBuffer:= '';
      fSystemRVSInfo.VolumeSerialNumber:= 0;
      fSystemRVSInfo.MaximumComponentLength:= 0;
      fSystemRVSInfo.FileSystemFlags:= 0;
      fSystemRVSInfo.FileSystemName:= '';
    end;
  finally
    SetErrorMode(ErrorMode);
  end;
end;

function TSystemRVSInfo.Index: Integer;
begin
  Result:= fSystemRVS;
end;

function TSystemRVSInfo.Letter: String;
begin
  Result:= cSystemRVSLetters[fSystemRVS];
end;

function TSystemRVSInfo.Serial: String;
begin
  Result:= IntToStr(fSystemRVSInfo.VolumeSerialNumber);
end;

function TSystemRVSInfo.Shell: ISystemRVSShellInfo;
begin
  Result:= TSystemRVSShellInfo.Create(fSystemRVS);
end;

procedure TSystemRVSInfo.ShowShellDialog;
Var
  Path: String;
  ExecuteInfo: TShellExecuteinfo;
Begin
  Path:= Format(cDisk_SystemRVSPath_Fmt, [cSystemRVSLetters[fSystemRVS]]);
  FillChar(ExecuteInfo, SizeOf(ExecuteInfo), 0);
  ExecuteInfo.cbSize:= SizeOf(ExecuteInfo);
  ExecuteInfo.lpFile:= PChar(Path);
  ExecuteInfo.fMask:= SEE_MASK_INVOKEIDLIST;
  ShellExecuteEx(@ExecuteInfo);
end;

function TSystemRVSInfo.Space: ISystemRVSSpace;
begin
  Result:= TSystemRVSSpace.Create(fSystemRVS);
end;

function TSystemRVSInfo.VolumeLabel: String;
begin
  Result:= Trim(fSystemRVSInfo.VolumeNameBuffer)
```

```
end;

{ Реалізація класу -  TMRcxSystemRVSInfo }

function TcxSystemRVSInfo.ByLetter(Letter: String): ISystemRVSInfo;
var
  i: Integer;
begin
  for i:= 0 to 25 do
    if cSystemRVSLetters[i] = UpperCase(Letter) then
    begin
      Result:= TSystemRVSInfo.Create(i);
      Exit;
    end;
end;

function TcxSystemRVSInfo.GetSystemRVSInfoByIndex(Index: Integer):
ISystemRVSInfo;
begin
  Result:= TSystemRVSInfo.Create(Index);
end;

function TcxSystemRVSInfo.Version: String;
begin
Result:=Format(RsDisk_Format_Version,[(cToolkit_Version div 1000),
((cToolkit_Version - ((cToolkit_Version div 1000)*1000)) div 100),
(cToolkit_Version mod 100)]+' Copyriht 2023');
end;
initialization
  cxSystemRVS:= TcxSystemRVSInfo.Create;
finalization
  cxSystemRVS.Free;
end. // Кінець файлу
```

Файл другого вікна ПЗ (Mr2.pas)

```
unit Mr2; // Початок файлу

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Лістинг ПЗ до магістерської роботи на тему:
Дослідження та програмна реалізація системи
реалізації інструментальних засобів організації
пакетів прикладних програм
Виконав: студент  Краєвський Микола Олегович
2023 рік
}


interface // Секція інтерфейсної частини файлу

Uses // Підключення бібліотек
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, stdCtrls, LZRW1KH, FileCtrl;

Type // Визначення типів
  TAlloc = function (AppData: Pointer; Items, Size: Integer): Pointer;
  TFree = procedure (AppData, Block: Pointer);
  TMRAStreamRec = packed record
    next_in: PChar;
    avail_in: Integer;
    total_in: Integer;
    next_out: PChar;
    avail_out: Integer;
    total_out: Integer;
    msg: PChar;
    internal: Pointer;
    zalloc: TAlloc;
    zfree: TFree;
    AppData: Pointer;
    data_type: Integer;
    reserved: Integer;
  end;

Type // Визначення типів
  TSystemReservMode       = (bmAll, bmIncremental);
  TRestoreMode      = (rmAll, rmNoOverwrite, rmNewer, rmExisting,
                       rmExistingNewer);
  TCompressionLevel = (clFastest, clNone, clDefault, clMax);
  TProgressEvent    = procedure(Sender : TObject; Filename:
                      String; Percent: TPercentage; var Continue:
                      Boolean) of object;
  TNeedDiskEvent    = procedure(Sender : TObject; DiskID: word;
                                var Continue: Boolean) of object;
  TRestoreFileEvent = procedure(Sender : TObject; var Filename:
        String; FA: Integer; var DoRestore: Boolean) of object;
  TFileRestoredEvent = procedure(Sender : TObject; var Filename:
                       String; var Stream: TFilestream) of object;
  TSystemReservErrorEvent = procedure(Sender : TObject; const Error:
                      integer; ErrString: string) of object;

  TSystemReservSeachFilePZ = class(TComponent)
  // Визначення класу від TComponent
  private
    FSystemReservTitle: string;
    fMaxSize: Integer;
    fSystemReservMode: TSystemReservmode;
```

```
    fRestoreMode: TRestoremode;
    fCompressionLevel: TCompressionLevel;
    fSetArchiveFlag: Boolean;
    fFilesTotal, fFilesProcessed, fSizeTotal, fProgressSize, fTotal: Integer;
    CurrentFile: string;
    IsBusy, Continue: boolean;
    fLastErr: integer;

    fInputStream, fOutputStream: TStream;
    fRestoreFullPath : Boolean;
    fSaveFileID     : Boolean;

    fOnProgress: TProgressEvent;
    fOnNeedDisk: TNeedDiskEvent;
    fOnRestoreFile: TRestoreFileEvent;
    fOnFileRestored: TFileRestoredEvent;
    fOnError: TSystemReservErrorEvent;
    function  GetVersion: string;
    procedure SetVersion(dummy: string);
    procedure SetSystemReservMode(value: TSystemReservmode);
    procedure DeCompress(InStream, OutStream: TStream; DoWrite: Boolean);
    procedure DeCompressOldMethod(InStream, OutStream: TStream; DoWrite:
Boolean);
    procedure MessageError(err: integer);
  protected
  public
    property  FilesTotal: Integer read fFilesTotal;
    property  SizeTotal: Integer read fSizeTotal;
    property  FilesProcessed: Integer read fFilesProcessed;
    function  SystemReservToStream(const Filelist: TStrings; Target:
                          TStream): boolean;
    function  SystemReserv(const Filelist: TStrings; Target: string):
                  boolean;
    function  RestoreFromStream(Source: TStream; TargetPath:
                           string): boolean;
    function  Restore(Source: String; TargetPath: string):
                  boolean;
    function  GetArchiveTitle(const Source: String; var Filelist:
                         TStringlist): string;
    function  GetArchiveTitleFromStream(Source: TStream; var
                                  Filelist: TStringlist):
                                  string;
    function  CompressionRate: integer;
    function  Busy: boolean;
    procedure Stop;
  published
    property SystemReservTitle: string read fSystemReservTitle write
fSystemReservTitle;
    property SystemReservMode: TSystemReservmode read fSystemReservMode write
SetSystemReservMode;
    property CompressionLevel: TCompressionLevel read fCompressionLevel write
fCompressionLevel;
    property RestoreMode: TRestoremode read fRestoreMode write fRestoreMode;
    property MaxSize: Integer read fMaxSize write fMaxSize;
    property SetArchiveFlag: Boolean read fSetArchiveFlag write fSetArchiveFlag;
    property OnProgress: TProgressEvent read fOnProgress write fOnProgress;
    property OnNeedDisk: TNeedDiskEvent read fOnNeedDisk write fOnNeedDisk;
    property OnRestoreFile: TRestoreFileEvent read fOnRestoreFile
             write fOnRestoreFile;
    property OnFileRestored: TFileRestoredEvent read |
             fOnFileRestored write fOnFileRestored;
    property OnError: TSystemReservErrorEvent read fOnError write
             fOnError;
    property RestoreFullPath : Boolean  read fRestoreFullPath
             write fRestoreFullPath;
```

```
    property SaveFileID:Boolean  read fSaveFileID write
             fSaveFileID;
  end;

implementation // Секція реалізації
procedure _tr_init; external;
procedure _tr_tally; external;
procedure _tr_flush_block; external;
procedure _tr_align; external;
procedure _tr_stored_block; external;
procedure adler32; external;
procedure inflate_blocks_new; external;
procedure inflate_blocks; external;
procedure inflate_blocks_reset; external;
procedure inflate_blocks_free; external;
procedure inflate_set_dictionary; external;
procedure inflate_trees_bits; external;
procedure inflate_trees_dynamic; external;
procedure inflate_trees_fixed; external;
procedure inflate_trees_free; external;
procedure inflate_codes_new; external;
procedure inflate_codes; external;
procedure inflate_codes_free; external;
procedure _inflate_mask; external;
procedure inflate_flush; external;
procedure inflate_fast; external;

procedure _memset(P: Pointer; B: Byte; count: Integer);cdecl;
begin
  FillChar(P^, count, B);
end;

procedure _memcpy(dest, source: Pointer; count: Integer);cdecl;
begin
  Move(source^, dest^, count);
end;

function deflateInit_(var strm: TMRAStreamRec; level: Integer; version: PChar;
recsize: Integer): Integer; external;
function deflate(var strm: TMRAStreamRec; flush: Integer): Integer; external;
function deflateEnd(var strm: TMRAStreamRec): Integer; external;

function inflateInit_(var strm: TMRAStreamRec; version: PChar; recsize:
Integer): Integer; external;
function inflate(var strm: TMRAStreamRec; flush: Integer): Integer; external;
function inflateEnd(var strm: TMRAStreamRec): Integer; external;
function inflateReset(var strm: TMRAStreamRec): Integer; external;


function zlibAllocMem(AppData: Pointer; Items, Size: Integer): Pointer;
begin
  GetMem(Result, Items*Size);
end;

procedure zlibFreeMem(AppData, Block: Pointer);
begin
  FreeMem(Block);
end;

{ Реалізація класу - TSystemReservSeachFilePZ}

function TSystemReservSeachFilePZ.GetVersion: string;
begin
     result:=cVersion;
end;
```

```
procedure TSystemReservSeachFilePZ.SetVersion(dummy: string);
begin
end;

procedure TSystemReservSeachFilePZ.SetSystemReservMode(value:
TSystemReservmode);
begin
     if value <> fSystemReservMode then
     begin
          if value = bmIncremental then SetArchiveFlag:=true;
          fSystemReservMode:=value;
     end;
end;

function TSystemReservSeachFilePZ.SystemReservToStream(const Filelist: TStrings;
Target: TStream): boolean;
begin
     if Target <> nil then
     try
        fOutputStream:=Target;
        fMaxSize:=0;
        Result:=SystemReserv(Filelist, ': STREAM');
     except
        Result:=false;
     end;
end;

function TSystemReservSeachFilePZ.SystemReserv(const Filelist: TStrings; Target:
string): boolean;
const //введення констант
  Levels: array [TCompressionLevel] of ShortInt =
 (Z_BEST_SPEED, Z_NO_COMPRESSION, Z_DEFAULT_COMPRESSION,
  Z_BEST_COMPRESSION);

var
   ext: string;
   I,L: Integer;
   FA: integer;
   S: String;
   Files  : TStringlist;
   Size: Longint;
   SStr: TFilestream;
   TStr: TStream;
   ArchiveNumber: Word;

   UseStream: boolean;

   FID: TStringlist;
   FIDTemp: string;
   TempFileP: array[0..255] of Char;

   procedure SystemReservFile;
   var
      InBuffer:  array[0..BufferSize-1] of Byte;
      OutBuffer: array[0..BufferSize-1] of Byte;
      Res, Size, BytesRead: Integer;
      FZRec: TMRAStreamRec;
      Is_: boolean;
   begin
      try
TStr.WriteBuffer(FSignature, SizeOf(FSignature));
       inc(fTotal, sizeof(FSignature));
while (Size > 0) and Continue and ((TStr.position < (MaxSize-100-BufferSize)) or
(MaxSize = 0)) do
```

```
          begin
               if  (DiskFree(0)-10*1024) < size then break;
               BytesRead:=SStr.Read(InBuffer, BufferSize);
               dec(Size, BytesRead);
               inc(fProgressSize, BytesRead);
               FillChar(FZRec, sizeof(FZRec), 0);
               FZRec.zalloc:=zlibAllocMem;
               FZRec.zfree:=zlibFreeMem;
               FZRec.next_out:=@OutBuffer;
               FZRec.avail_out:=sizeof(OutBuffer);
if deflateInit_(FZRec, Levels[fCompressionLevel], zlib_version,
                 sizeof(FZRec)) < 0 then
               begin
                    fLastErr:=idCompression;
                    Continue:=false;
               end;

               FZRec.next_in:=@InBuffer;
               FZRec.avail_in:=BytesRead;

               Res:=deflate(FZRec, Z_FINISH);
               case Res of
               Z_OK: is_:=false;
                  Z_STREAM_END: is_:=true;   // Буфер
                  else begin
                    fLastErr:=idCompression;
                    continue:=false;
                  end;
               end;
               if Continue then
               begin
                    if is_ then
                    begin
                         Size:=FZRec.total_out;
                         TStr.WriteBuffer(Size, SizeOf(Size));
                         TStr.Write(OutBuffer, Size);
                    end else
                    begin
                         Size:=BytesRead * (-1);
                         TStr.WriteBuffer(Size, SizeOf(Size));
                         TStr.Write(InBuffer, BytesRead);
                    end;
               end;

               deflateEnd(FZRec);

               inc(fTotal, abs(Size));
               inc(fTotal, sizeof(Size));

               Application.processmessages;
if assigned(fOnProgress) then fOnProgress(self, CurrentFile, (fProgressSize*100)
                                          div fSizeTotal, Continue);
          end;

          Size:=0;  //кінець файлу
          TStr.WriteBuffer(Size, SizeOf(Size));

          if (Size > 0) and Continue then
          begin
                 inc(ArchiveNumber);
                 ext:=('00'+inttostr(ArchiveNumber));
                 ext:=copy(ext, length(ext)-2, 3);

                 S:='NEXT:DISK'+Ext;
                 L:=length(s);
```

```
                TStr.writeBuffer(L, sizeof(L));
                TStr.writeBuffer(PChar(s)^,L);
                TStr.free;
                TStr:=nil;

if assigned(fOnNeedDisk) then fOnNeedDisk(self, ArchiveNumber, Continue)
                else Continue:=MessageDlg(Format(cInsertDisk,
[inttostr(ArchiveNumber)]), mtInformation, mbOKCancel, 0) = mrOK;

                if Continue then
                begin
                  Target:=ChangeFileExt(target, '.'+ext);
                  TStr:=TFilestream.create(Target, fmCreate);
                  TStr.seek(0, 0);
                end;
         end;

      except
         Continue:=false;
         fLastErr:=idCantWriteArchive;
      end;
   end;

   procedure FindFiles(pattern: string);
   var
      IncludeSubs: boolean;
      SR: TSearchRec;
      FindResult: Integer;
   begin
        pattern:=lowercase(pattern);
        IncludeSubs:=pos('/s',pattern) > 0;

        if IncludeSubs then
        begin
pattern:=trim(copy(pattern, 1, pos('/s',pattern)-1));
FindResult:=FindFirst(ExtractFilePath(pattern)+'*.*', faDirectory, SR);
            while FindResult = 0 do
            begin
if (SR.Name <> '.') and (SR.Name <> '..') and (sr.Attr AND faDirectory > 0) then
            begin

S:=ExpandFilename(ExtractFilepath(pattern)+SR.name+'\'+extractfilename(pattern))
+' /s';
                FindFiles(lowercase(S));
                end;
                FindResult:=FindNext(SR);
            end;
            FindClose(SR);
        end;
        FindResult:=FindFirst(pattern, faAnyFile-faDirectory, SR);
        while FindResult = 0 do
        begin
            S:=lowercase(ExpandFilename(ExtractFilepath(pattern)+SR.name));
            if (files.indexof(S) = -1) and (S <> lowercase(Target)) then
            begin
if SystemReservMode = bmAll then Files.add(S)
else if (SR.Attr AND faArchive > 0) then Files.add(S);
            end;
            FindResult:=FindNext(SR);
        end;
        FindClose(SR);
   end;
begin
     UseStream:=Target = ':STREAM';
     fLastErr:=0;
```

```
     IsBusy:=true;
     Files:=TStringlist.create;
     if fSaveFileID then FID:=TStringlist.create;
     files.beginupdate;
     for I:=0 to Filelist.count-1 do FindFiles(Filelist[i]);
     files.endupdate;
     result:=false;
     Continue:=true;
if assigned(fOnProgress) then fOnProgress(self, '', 0, Continue);
     fSizeTotal:=0;
     fFilesProcessed:=0;
     for I:=0 to files.count-1 do
         if Fileexists(files[i]) then
            try
SStr:=TFilestream.create(files[i], fmOpenRead or fmShareDenyNone);
fSizeTotal:=fSizeTotal + SStr.Size;

if fSaveFileID then FID.add(files[i] + #9 + nttostr(Fileage(Files[i])) + '=' +
inttostr(SStr.size));
            finally
               SStr.free;
            end;

     if fSaveFileID then
     try
        GetTempPath(SizeOf(TempFileP),@TempFileP);
        GetTempFileName(TempFileP, 'FID', 0, TempFileP);
        FIDTemp:=StrPas(TempFileP);
        FID.savetofile(FIDTemp);
        Files.insert(0, FIDTemp);
        inc(fSizeTotal, length(FID.text));
        FID.free;
     except;
        Continue:=false;
        fLastErr:=idCantCreateFileID;
     end;
     try
     ArchiveNumber:=1;
     if UseStream then TStr:=fOutputStream
     else begin
           TStr:=TFilestream.create(Target, fmCreate);
           TStr.seek(0, 0);
     end;
     fFilesTotal:=files.count;
     if fSaveFileID then fFilesTotal:=files.count-1;
     L:=length(fSystemReservTitle);
     TStr.writeBuffer(L, sizeof(L));    //розмір буфера
     TStr.writeBuffer(PChar(fSystemReservTitle)^,L);
     TStr.writeBuffer(fSizeTotal, sizeof(fSizeTotal));
     //розмір файлу SystemReserv
     TStr.writeBuffer(fFilesTotal, sizeof(fFilesTotal));
     //Кількість файлів SystemReserv
     fProgressSize:=0;
     fTotal:=0;
     except
       fLastErr:=idCantwriteArchive;
       Continue:=false;
     end;
     for I:=0 to files.count-1 do
     begin
        if Fileexists(files[i]) and Continue then
        begin
             try
 if fSaveFileID and (I = 0) then CurrentFile:='FILE:LIST'
               else CurrentFile:=trim(files[i]);
```

```
                l:=length(CurrentFile);
                //розмір поточного файлу
                TStr.writeBuffer(L, sizeof(L));
                TStr.writeBuffer(PChar(CurrentFile)^,L);
                //Ім'я файлу
                CurrentFile:=trim(files[i]);
                try
SStr:=TFilestream.create(Currentfile, fmOpenRead or
                         fmShareDenyNone);
                  FA:=FileGetDate(SStr.handle);
                  TStr.writeBuffer(FA, sizeof(FA));
                  Size:=SStr.Size;
                  while (Size > 0) and Continue do SystemReservFile;
                  if Continue then
                  repeat
                        SystemReservFile;
                  until (Size <= 0) or (not Continue);

if (CurrentFile <> FIDTemp) or (not fSaveFileID) then
                inc(fFilesProcessed);
                except
                  continue:=false;
                  fLastErr:=idCantReadfile;
                end;
                if not continue then break;
             finally
               SStr.free;
if fSetArchiveFlag then FileSetAttr(files[i],
    FileGetAttr(files[i])- faArchive);
             end;
        end;
     end;

     if Continue then
     try
       L:=0;
       TStr.writeBuffer(L, sizeof(L)); // SystemReserv файл
       result:=true;
     except
       fLastErr:=idCantwriteArchive;
       Continue:=false;
     end;
     try
        Files.free;
        if not UseStream then TStr.free;
     except
     end;
     if fSaveFileID and (FIDTemp <> '') and Fileexists(FIDTemp)
     then Deletefile(FIDTemp);
     if assigned(fOnProgress) then fOnProgress(self, '', 100,
                                               Continue);
     IsBusy:=false;
     if (fLastErr <> 0) then MessageError(fLastErr);
end;

procedure TSystemReservSeachFilePZ.MessageError(err: integer);
var
   S: string;
begin
     case err of
       idCantreadFile:     S:=errCantreadFile;
       idCantwriteFile:    S:=errCantwriteFile;
       idCantreadArchive:  S:=errCantreadArchive;
       idCantwriteArchive: S:=errCantwriteArchive;
       idInvalidfiletype:  S:=errInvalidfiletype;
```

```
       idCompression:      S:=errCompression;
       idCantCreateFileID: S:=errCantCreateFileID;
     end;
     if assigned(fOnError) then fOnError(self, err, S)
     else MessageDlg(S, mtError, [mbOK], 0);
end;

function TSystemReservSeachFilePZ.CompressionRate: integer;
begin
   try
     result:=100 - ((fTotal * 100) div fSizeTotal);
   except
     result:=0;
   end;
end;

function TSystemReservSeachFilePZ.RestoreFromStream(Source: TStream; TargetPath:
string): boolean;
begin
     if Source <> nil then
     try
       fInputStream:=Source;
       Result:=Restore(':STREAM', TargetPath);
     except
       result:=false;
     end;
end;

function TSystemReservSeachFilePZ.Restore(Source: String; TargetPath: string):
boolean;
var
   L: Integer;
   FA, FAT: integer;
   S, Disk: String;
   SStr: TStream;
   TStr: TFilestream;
   DoRestore: Boolean;
   UseStream: Boolean;
begin
     if (TargetPath <> '') and (TargetPath[length(TargetPath)] <> '\') then
TargetPath:=TargetPath + '\';
     UseStream:=Source = ':STREAM';
     fLastErr:=0;
     result:=false;
     fFilesProcessed:=0;
     Continue:=true;
     try
        if UseStream then SStr:=fInputStream
        else begin
             if not Fileexists(Source) then exit;
             SStr:=TFilestream.create(Source, fmOpenRead or fmShareDenyNone);
             SStr.seek(0,0);
        end;
        IsBusy:=true;

        SStr.readbuffer(L, sizeof(L));
        SetString(S, PChar(nil), L);
        SStr.ReadBuffer(PChar(S)^,L);
        SStr.readbuffer(fSizeTotal, sizeof(fSizeTotal));
        SStr.readbuffer(fFilesTotal, sizeof(fFilesTotal));
        fProgressSize:=0;
     except
        Raise Exception.Create(errCantreadArchive);
        IsBusy:=false;
        if UseStream then SStr.free;
```

```
        exit;
      end;

      DoRestore:=true;
      repeat
         SStr.readbuffer(L, sizeof(L)); // розмір файлу
         if L > 0 then
         begin
           SetString(S, PChar(nil), L);
           SStr.ReadBuffer(PChar(S)^,L);     //Ім'я файлу

           if copy(s,1,9) = 'NEXT:DISK' then
           begin
              SStr.free;
              SStr:=nil;
              Disk:=copy(S, 10, 3);
              while copy(disk,1,1) = '0' do
              Disk:=copy(Disk, 2, length(Disk)-1);

              repeat
if assigned(fOnNeedDisk) then fOnNeedDisk(self, strtoint(Disk),
                                          Continue)
else Continue:=MessageDlg(Format(cInsertDisk, [Disk]),
             mtInformation, mbOKCancel, 0) = mrOK;
             Source:=changeFileext(source, '.'+copy(S,10,3));
             until Fileexists(Source) or (not Continue);

             If Continue then
             try
SStr:=TFilestream.create(Source,
                        fmOpenRead or fmShareDenyNone);
               SStr.seek(0,0);
               if DoRestore then
               begin
              TStr:=TFilestream.create(CurrentFile, fmOpenWrite);
              TStr.seek(TStr.size, soFromBeginning);
               end;
              DeCompress(SStr, TStr, DoRestore);
             finally
               if DoRestore then
               begin
                 if assigned(fOnFileRestored) then fOnFileRestored(self,
CurrentFile, TStr);
FileSetDate(TStr.handle, FA);
                 TStr.free;
               end;
             end;
           end
           else begin
                // читання файлу
SStr.ReadBuffer(FA, sizeof(FA));
if TargetPath <> '' then CurrentFile:=TargetPath+ExtractFileName(S)
else CurrentFile:=S;
if fRestoreFullPath and (TargetPath <> '') then CurrentFile:=TargetPath +
copy(S, 4, Length(S));
FAT:=FileAge(CurrentFile);
           case fRestoreMode of
rmAll: DoRestore:=true;
rmNoOverwrite: DoRestore:=not Fileexists(CurrentFile);
             rmNewer: DoRestore:=not Fileexists(CurrentFile) or (FA > FAT);
             rmExisting: DoRestore:=Fileexists(CurrentFile);
             rmExistingNewer: DoRestore:=Fileexists(CurrentFile) and (FA >
FAT);
           end;
```

```
            if S = 'FILE:LIST' then DoRestore:=false
            else if assigned(fOnRestoreFile) then fOnRestoreFile(self,
CurrentFile, FA, doRestore);

            if DoRestore then
            begin
ForceDirectories(extractFileDir(CurrentFile));
                 DoRestore:=DirectoryExists(extractFileDir(CurrentFile));
            end;
            if DoRestore then
            try
              try
                TStr:=TFileStream.create(CurrentFile, fmCreate);
                TStr.seek(0, 0);
                DeCompress(SStr, TStr, true);
                inc(fFilesProcessed);
              except
                fLastErr:=idCantReadFile;
                Continue:=false;
              end;
            finally
              if assigned(fOnFileRestored) then fOnFileRestored(self,
CurrentFile, TStr);
              FileSetDate(TStr.handle, FA);
              TStr.free;
            end
            else DeCompress(SStr, nil, false);
          end;
        end;
     until (L = 0) or (not Continue);

     if not UseStream and (SStr <> nil) then SStr.free;
     result:=(fLastErr = 0) and Continue;
     if assigned(fOnProgress) then fOnProgress(self, '', 100, Continue);
     IsBusy:=false;
     if fLastErr <> 0 then MessageError(fLastErr);
end;

function TSystemReservSeachFilePZ.Busy: boolean;
begin
   result:=IsBusy;
end;

procedure TSystemReservSeachFilePZ.Stop;
begin
   Continue:=false;
end;

function TSystemReservSeachFilePZ.GetArchiveTitleFromStream(Source: TStream; var
Filelist: TStringlist): string;
begin
     if Source <> nil then
     try
       fInputStream:=Source;
       Result:=GetArchiveTitle(':STREAM', Filelist);
     except
       result:='';
     end;
end;

function TSystemReservSeachFilePZ.GetArchiveTitle(const Source: String; var
Filelist: TStringlist): string;
var
   L: Integer;
   S: String;
```

```
   SStr: TFilestream;
   FIDStr: TMemorystream;
   UseStream: Boolean;
begin
     UseStream:=Source = ':STREAM';
     result:='';
     fSizeTotal:=0;
     fFilesTotal:=0;
     fFilesProcessed:=0;

     if UseStream then SStr:=TFileStream(fInputStream)
     else begin
        if not Fileexists(Source) then exit;
        try
           SStr:=TFilestream.create(Source, fmOpenRead or
                                    fmShareDenyNone);
           SStr.seek(0,0);
        except;
           fLastErr:=idCantReadArchive;
           MessageError(fLastErr);
        end;
     end;

     try
        SStr.readbuffer(L, sizeof(L));
        SetString(S, PChar(nil), L);
        SStr.ReadBuffer(PChar(S)^,L);
        Result:=S;
        SStr.readbuffer(fSizeTotal, sizeof(fSizeTotal));
        SStr.readbuffer(fFilesTotal, sizeof(fFilesTotal));

        SStr.readbuffer(L, sizeof(L)); // розмір файлу
        if (L > 0) and (Filelist <> nil) then
        begin
          SetString(S, PChar(nil), L);
          SStr.ReadBuffer(PChar(S)^,L);    //Ім'я
          if S = 'FILE:LIST' then
          begin
            FIDStr:=TMemoryStream.create;
            FIDStr.seek(0, 0);
            SStr.ReadBuffer(L, sizeof(L));
            Continue:=true;
            DeCompress(SStr, FIDStr, true);
            FIDStr.seek(0, 0);
            Filelist.loadfromstream(FIDStr);
            FIDStr.free;
          end;
        end;
     except
        fLastErr:=idInvalidFileType;
        MessageError(fLastErr);
     end;
     if not UseStream then SStr.free;
end;

procedure TSystemReservSeachFilePZ.DeCompress(InStream, OutStream: TStream;
DoWrite: Boolean);
var
   InBuffer:  array[0..BufferSize-1] of Byte;
   OutBuffer: array[0..BufferSize-1] of Byte;
   Size, UnSize: Integer;
   Sig: array[0..SizeOf(FSignature)-1] of Char;
   FZRec: TMRAStreamRec;
   is_: boolean;
begin
```

```
  InStream.ReadBuffer(Sig, SizeOf(FSignature));
  if Sig <> FSignature then raise Exception.Create(errInvalidfiletype);
  Size:=-1;
  while (Size <> 0) and Continue do
  begin
    InStream.ReadBuffer(Size, SizeOf(Size));
    if Size <> 0 then
    begin
         is_:=Size > 0;
         Size:=abs(Size);
         if DoWrite then
         begin
             InStream.Readbuffer(InBuffer, Size);

             if is_ then
             begin
                FillChar(FZRec, sizeof(FZRec), 0);
                FZRec.zalloc:=zlibAllocMem;
                FZRec.zfree:=zlibFreeMem;
                FZRec.next_in:=@OutBuffer;
                FZRec.avail_in:=0;
if inflateInit_(FZRec, zlib_version, sizeof(FZRec)) < 0 then
                begin
                   fLastErr:=idCompression;
                   Continue:=false;
                end;

                FZRec.next_in:=@InBuffer;
                FZRec.avail_in:=Size;
                FZRec.next_out:=@OutBuffer;
                FZRec.avail_out:=BufferSize;

                if inflate(FZRec, 0) < 0 then
                begin
                   fLastErr:=idCompression;
                   Continue:=false;
                end;
                unSize:=FZRec.total_out;
                OutStream.Write(OutBuffer, UnSize);
                 inflateEnd(FZRec);
             end
             else begin
                unSize:=Size;
                OutStream.Write(InBuffer, UnSize);
             end;
         end
         else begin
              InStream.position:=InStream.position + Size;
              UnSize:=BufferSize;
         end;
    end;

    Application.processmessages;
    if OutStream is_ TFilestream then
    begin
       inc(fProgressSize, UnSize);
       if assigned(fOnProgress) then fOnProgress(self, CurrentFile,
(fProgressSize*100) div fSizeTotal, Continue);
    end;
  end;
end;

procedure TSystemReservSeachFilePZ.DeCompressOldMethod(InStream, OutStream:
TStream; DoWrite: Boolean);
var
```

```
    InBuffer, OutBuffer: BufferArray;
    Size, UnSize, InSize: LongInt;
begin
  InStream.ReadBuffer(InSize, SizeOf(InSize));
  while (InSize > 0) and Continue do
  begin
    InStream.ReadBuffer(Size, SizeOf(Size));
    InStream.ReadBuffer(InBuffer, Size);
    if DoWrite then
    begin
         UnSize:=DeCompression(@InBuffer, @OutBuffer, Size);
         OutStream.WriteBuffer(OutBuffer, UnSize);
    end;
    InSize:=InSize - Size - SizeOf(Size);

    Application.processmessages;
    inc(fProgressSize, UnSize);
    if assigned(fOnProgress) then fOnProgress(self, CurrentFile,
(fProgressSize*100) div fSizeTotal, Continue);
  end;
end;
end. // Кінець файлу
```

Файл третьйого вікна ПЗ (Mr3.pas)

```
unit Mr3; // Початок файлу

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Лістинг ПЗ до магістерської роботи на тему:
Дослідження та програмна реалізація системи
реалізації інструментальних засобів організації
пакетів прикладних програм
Виконав: студент  Краєвський Микола Олегович
2023 рік
}

interface // Секція інтерфейсної частини файлу

Uses // Підключення бібліотек
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Registry;

Type // Визначення типів
  TmRootKey = (rkHKEY_CLASSES_ROOT, rkHKEY_CURRENT_USER, rkHKEY_LOCAL_MACHINE,
    rkHKEY_USERS, rkHKEY_CURRENT_CONFIG);

  TmRegistrySeach = class(TComponent)
  // Визначення класу від TComponent
  private
    fAbout: string;
    fKey: string;
    fRootKey: TmRootKey;
    fValues: array of array[0..1] of string;
    fCanCreate: Boolean;
    fVCount: Integer;
    Data: TStringList;
    procedure SetAbout(Value: string);
    function GetRootKey: HKEY;

  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure ReadFromRegistry(var Data: TStringList);
    function ReadValueFromRegistry(Value: string): string;
    procedure RegData(Value, Data: string);
    function WriteToRegistry: boolean;
    function ClearValue: boolean;
    { Private declarations }

  published
    property RootKey: TmRootKey read fRootKey write fRootKey;
    property Key: string read fKey write fKey;
    property CanCreate: Boolean read fCanCreate write fCanCreate;
    property About: string read fAbout write SetAbout;

    { Published declarations }
  end;

procedure Register;


implementation // Секція реалізації

constructor TmRegistrySeach.Create(AOwner: TComponent);
```

```
begin
  inherited Create(AOwner);
  Data:= TStringList.Create;
  fRootKey:= rkHKEY_LOCAL_MACHINE;
  fKey:= '\software\mas prod.\';

  fCanCreate:= true;
  fVCount:= 0;
end;

destructor TmRegistrySeach.Destroy;
begin
  Data.Free;
  inherited Destroy;
end;

procedure TmRegistrySeach.SetAbout(Value: string);
begin
  Exit;
end;

function TmRegistrySeach.GetRootKey: HKEY;
// Пошук ключів реестру
begin
  Result:= HKEY_LOCAL_MACHINE;
  case fRootKey of
    rkHKEY_CLASSES_ROOT:
      Result:= HKEY_CLASSES_ROOT;
    rkHKEY_CURRENT_USER:
      Result:= HKEY_CURRENT_USER;
    rkHKEY_LOCAL_MACHINE:
      Result:= HKEY_LOCAL_MACHINE;
    rkHKEY_USERS:
      Result:= HKEY_USERS;
    rkHKEY_CURRENT_CONFIG:
      Result:= HKEY_CURRENT_CONFIG;
  end;
end;

procedure TmRegistrySeach.RegData(Value, Data: string);
begin
  Inc(fVCount);
  SetLength(fValues, fVCount);
  if Value = '' then
    Value:= 'Value' + IntToStr(fVCount);
  fValues[fVCount - 1, 0]:= Value;
  fValues[fVCount - 1, 1]:= Data;
end;

function TmRegistrySeach.WriteToRegistry: Boolean;
var
  fReg: TRegistry;
  n: Integer;
begin
  Result:= false;
  fReg:= TRegistry.Create;
  try
    fReg.RootKey:= GetRootKey;
    if fReg.KeyExists(fKey) then
    begin
      if fReg.OpenKey(fKey, false) then
      begin
        for n:= 0 to fVCount - 1 do
```

```
          fReg.WriteString(fValues[n, 0], fValues[n, 1]);
        Result:= true;
      end
    end
    else
    begin

      if fCanCreate then
        if fReg.OpenKey(fKey, true) then
        begin
          for n:= 0 to fVCount - 1 do
            fReg.WriteString(fValues[n, 0], fValues[n, 1]);
          Result:= true;
        end;

    end;
  finally

    fReg.Free;
  end;
end;

procedure TmRegistrySeach.ReadFromRegistry(var Data: TStringList);
var
  fReg: TRegistry;
  fValueNames: TStringList;
  n: Integer;

begin
  fReg:= TRegistry.Create;
  fValueNames:= TStringList.Create;
  try

    fReg.RootKey:= GetRootKey;
    if fReg.OpenKey(fKey, false) then
    begin
      fReg.GetValueNames(fValueNames);
      for n:= 0 to fValueNames.Count - 1 do
      begin
        Data.Add(fReg.ReadString(fValueNames[n]));
      end;
    end
    else
      Data.Add('');

  finally
    fReg.Free;
    fValueNames.Free;
  end;
end;

function TmRegistrySeach.ClearValue: boolean;
var
  fReg: TRegistry;
begin
  fReg:= TRegistry.Create;
  try
    fReg.RootKey:= GetRootKey;
    if fReg.OpenKey(fKey, false) then
      fReg.DeleteKey(fKey);
  finally
    fReg.Free;
  end;
end;
```

```
function TmRegistrySeach.ReadValueFromRegistry(Value: string): string;
var
  fReg: TRegistry;
begin
  fReg:= TRegistry.Create;

  try
    fReg.RootKey:= GetRootKey;
    if fReg.OpenKey(fKey, false) then
      Result:= fReg.ReadString(Value)
    else
      Result:= '';

  finally
    fReg.Free;
  end;

end;

end. // Кінець файлу
```

Файл четвертого вікна ПЗ (Mr4.pas)

```
unit Mr4; // Початок файлу

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Лістинг ПЗ до магістерської роботи на тему:
Дослідження та програмна реалізація системи
реалізації інструментальних засобів організації
пакетів прикладних програм
Виконав: студент  Краєвський Микола Олегович
2023 рік
}

interface // Секція інтерфейсної частини файлу

Uses // Підключення бібліотек
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Type // Визначення типів
  TgMrReg = class(TComponent) // Визначення класу від TComponent
  private
    mKey : String;
    mValue : String;
    mDefault : String;

    Function pGetKey : String;
    Procedure pSetKey(Key:String);

    Function pGetValue : String;
    Procedure pSetValue(Value:String);

    Function pGetDefault:String;
    Procedure pSetDefault(DefaultValue:String);
  protected
  public
    Function SetKey:Boolean;
    Function GetKey:String;
    Function DelKey:Boolean;
  published
    property Key : String read pGetKey write pSetKey;
    property Value : String read pGetValue write pSetValue;
    Property DefaultValue : String Read pGetDefault Write pSetDefault;

  end;

procedure Register;

implementation // Секція реалізації
   Uses // Підключення бібліотек
   Registry;

Function TgMrReg.pGetKey:String;
Begin
   pGetKey:=mKey;
End;

Procedure TgMrReg.pSetKey(Key:String);
Begin
   mKey:=Key;
End;

Function TgMrReg.pGetValue:String;
```

```
Begin
   pGetValue:=mValue;
End;

Procedure TgMrReg.pSetValue(Value:String);
Begin
   mValue:=Value;
End;

Function TgMrReg.pGetDefault:String;
Begin
   pGetDefault:=mDefault;
End;

Procedure TgMrReg.pSetDefault(DefaultValue:String);
Begin
   mDefault:=DefaultValue;
End;
Function TgMrReg.SetKey:Boolean;
Var
   Registre : TRegistry;
   ECantOpenRegistryKey : Exception;
begin
   Registre:=TRegistry.Create;
   If Not Registre.OpenKey(mKey,True) Then
   Begin
      Raise ECantOpenRegistryKey;
      Result:=False;
      Exit;
   End;
   Registre.WriteString(mKey,mValue);
   Registre.CloseKey;
   Registre.Destroy;
   ECantOpenRegistryKey.Destroy;
   Result:=True;
end;
Function TgMrReg.GetKey: String;
Var
   Registre : TRegistry;
Begin
   Registre:=TRegistry.Create;
   If Not Registre.OpenKey(mKey,False) Then
   Begin
      Result:=mDefault;
      Exit;
   End;
   Result:=Registre.ReadString(mKey);
   Registre.CloseKey;
   Registre.Destroy;
End;
Function TgMrReg.DelKey:Boolean;
Var
   Registre : TRegistry;
Begin
   Registre:=TRegistry.Create;
   If Not Registre.DeleteKey(mKey) Then
   Begin
      Result:=False;
      Exit;
   End;
   Result:=True;
   Registre.Destroy;
End;
end. // Кінець файлу
```

Файл п'ятого вікна ПЗ (Mr5.pas)

```
unit Mr5; // Початок файлу

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Лістинг ПЗ до магістерської роботи на тему:
Дослідження та програмна реалізація системи
реалізації інструментальних засобів організації
пакетів прикладних програм
Виконав: студент  Краєвський Микола Олегович
2023 рік
}

interface // Секція інтерфейсної частини файлу

Uses // Підключення бібліотек
 Windows, Messages, SysUtils, Classes;

 Type // Визначення типів
 TmINI = class(TObject) // Визначення класу від TObject
  private
   FAboutText : String;
   FMainData : TStringList;
   FTempData : TStringList;
   function GetValue(str : string) : string;
   {отримання даних зі списку }
   function GetSectionStart(section : string) : integer;
   {Повернення індексу початку секції}
   function GetSectionEnd(section : string; start : integer):
   integer;
   {Повернення індексу кінця секції}
   function ReadTempData : integer; {Читання даних з FTempData}
   function GetEntryPosition(section, Key : string) : integer;
  protected
   function GetCharPos(str : string; character : char) : integer;
   public
   constructor Create(filename : string);
   destructor Destroy; override;
   function Parse : integer;
   procedure LoadFromStream(thestream : TStream);
   procedure LoadFromFile(filename : String);
   procedure AddFromStream(thestream : TStream);
   procedure AddFromFile(filename : String);
   procedure SaveToStream(thestream : TStream);
 // Збереження даних у файл
   procedure SaveToFile(filename : String);
   procedure RawToFile(filename : string);
   procedure RawToStream(output : TStream);
   procedure RawFromFile(filename : string);
   procedure RawFromStream(input : TStream);
   property About : String read FAboutText;
// Авторське право
   property TempData : TStringList read FTempData;
  end;

implementation // Секція реалізації

const //введення констант
{строкові константи пошуку}
 SectionChar = '[';
 SectionEnd  = ']';
 ValueEquals = '=';
```

```
 CommentChar = ';';
 Delimiter   = ',';
{Введення типу пошуку з параметрами: 1-знайдено, 0-ні}
 BoolOne     = '1';
 BoolZero    = '0';
{Конструктор та деструктор класу}
constructor TMRMrINI.Create(filename : string);
// робота конструктора
begin
FAboutText:=copyrighttext;
FMainData:=TStringList.Create;
FTempData:=TStringList.Create;
if filename <> '' then LoadFromFile(filename);
end;

destructor TMRMrINI.Destroy;
// Робота деструктора
begin
ClearDatabase;
FMainData.Free;
FTempData.Free;
end;



{Функції аналізу Ini файлів}

function TMRMrINI.BuildINIData : integer;
var
 position : integer;
 working, Working2 : string;
 Section : string;
begin
Section:='';
For position:=0 to fMaindata.count-1 do
 begin
Working:=FMainData.strings[position];
 Working2:=copy(working, 1, Pos(Delimiter,working)-1);
 delete(working, 1, Pos(Delimiter,working));
 if Working2 <> Section then
  begin
  if position <> 0 then Ftempdata.Add('');
  FTempData.Add(SectionChar+Working2+SectionEnd);
  Section:=Working2;
  end;

 Working2:=copy(working, 1, Pos(Delimiter,working)-1);
 Delete(working, 1, Pos(Delimiter,working));
 FtempData.add(working2+Valueequals+working);
 end;
Result:=0;
end;

function TMRMrINI.ReadTempData : integer;
{Читання INI даних}
var
 test : integer;
 pos1 : integer;
 Section : string;
 temp, temp2, temp3 : string;
begin
For pos1:=0 to FTempData.count-1 do
 begin
 temp:=FTempData[pos1];
 if length(temp) > 0 then  {Перевірка довжини }
```

```
  begin
  case temp[1] of
   SectionChar : begin {Знайдена нова секція }
                 Delete(temp,1,1);
                 Delete(Temp,length(temp), length(temp));
                 Section:=temp;
                 end;
commentchar : begin
                 end;
   else
Temp2:=temp;
Temp3:=temp;
Test:=Pos(ValueEquals, temp);
   if test <> 0 then
    begin

    Delete(temp2,test,length(temp2));
    Delete(temp3,1,test);             {видалення}
    FMainData.Add(section+delimiter+temp2+Delimiter+temp3);
    end;
   end;
  end;
 end;
FTempData.Clear;
 FMainData.Sort;
result:=0;
end;

function TMRMrINI.Parse : integer;
begin
result:=ReadtempData;
end;

function TMRMrINI.GetSectionStart(section : string):
                          integer;
var
 position : integer;
 temp : integer;
 found : boolean;
 finalposition : integer;
 temp2 : string;
 str : string;
begin
Found:=false;
finalposition:=-1;
str:=Section;
for position:=0 to FMainData.Count-1 do
 begin
 temp:=pos(str, FMainData.Strings[position]);
 if (temp = 1)
 then
  begin
  Temp2:=FMainData.Strings[position];
  if(temp2[length(str)+1] = Delimiter) then
   begin                              {знайдено поле}
   found:=true;
   finalPosition:=position;
   Break;
   end;
  end;
 end;
if found = false then Result:=-1
 else Result:=finalposition; {повернення результату}
end;
```

```
function TMRMrINI.GetSectionEnd(section:string; start:
                            integer):integer;
var
 position : integer;
 temp : integer;
 found : boolean;
 finalposition : integer;
 temp2 : string;
 str : string;
begin
Found:=false;
finalposition:=-1;
str:=Section;
for position:=start to FMainData.Count-1 do
 begin
 temp:=pos(str, FMainData.Strings[position]);
 temp2:=FMainData.Strings[position];
 if (temp <> 1) and (temp2[length(str)+1] <> Delimiter)
 then
  begin
  Finalposition:=position-1;
  found:=true;
  break;
  end;
 end;
if found = false then Result:=-1
 else Result:=finalposition;
end;

function TMRMrINI.GetEntryPosition(section, Key : string)
                            :integer;
var
 position : integer;
 temp : integer;
 found : boolean;
 finalposition : integer;
 temp2 : string;
 str : string;
begin
Found:=false;
finalposition:=-1;
str:=Section+Delimiter+Key;
for position:=0 to FMainData.Count-1 do
 begin
 temp:=pos(str, FMainData.Strings[position]);
 if (temp = 1)
 then
  begin
  Temp2:=FMainData.Strings[position];
  if(temp2[length(str)+1] = Delimiter) then
   begin
   found:=true;
finalPosition:=position;
   Delete(CorrectStr, 1,length(str)+1);
   Break;
   end;
  end;
 end;
if found = false then Result:=-1
 else Result:=finalposition; {Return value data}
end;

function TMRMrINI.GetValue(str : string) : string;
var
 position : integer;
```

```
 temp : integer;
 found : boolean;
 CorrectStr : string;
 temp2 : string;
begin
Found:=false;
for position:=0 to FMainData.Count-1 do
 begin
 temp:=pos(str, FMainData.Strings[position]);
 if (temp = 1)
 then
  begin
  Temp2:=FMainData.Strings[position];
  if(temp2[length(str)+1] = Delimiter) then
   begin
   found:=true;
   CorrectStr:=FMainData.Strings[position];
   Delete(CorrectStr, 1,length(str)+1);
   Break;
   end;
  end;
 end;
if found = false then Result:=StartDefault
{повернення результату,якщо не знайдено}
else
 begin
 Result:=Correctstr; {повернення результату,якщо знайдено}
 end;
end;

function TMRMrINI.GetCharPos(str:string; character:
                                        char):integer;
{Пошук символу у файлі Ini}
var
 posi, position: integer;
begin
position:=0;

for posi:=1 to length(str) do
 begin
 if str[posi]=character then
  begin
  position:=posi;
  Break;
  end;
 end;
result:=position;
end;

{Процедури завантаження та зберігання Ini файлів}
procedure TMRMrINI.LoadFromStream(thestream : TStream);
begin
ClearDatabase;
FTempData.Clear;
FTempData.LoadFromStream(TheStream);
ReadTempData;
end;

procedure TMRMrINI.LoadFromFile(filename : String);
begin
ClearDataBase;
FTempData.Clear;
FTempData.LoadFromFile(filename);
ReadTempData;
end;
```

```
procedure TMRMrINI.AddFromStream(thestream : TStream);
begin
FTempData.Clear;
FTempData.LoadFromStream(TheStream);
ReadTempData;
end;

procedure TMRMrINI.AddFromFile(filename : String);
// Процедура додавання
begin
FTempData.Clear;
FTempData.LoadFromFile(filename);
ReadTempData;
end;

procedure TMRMrINI.SaveToStream(thestream : TStream);
{Процедура збереження}
begin
FTempData.Clear;
BuildINIData;
FtempData.SavetoStream(thestream);
FTempData.clear;
end;

procedure TMRMrINI.SaveToFile(filename : String);
{Save encoded INI data to a file}
begin
FTempData.Clear;
BuildINIData;
FtempData.SavetoFile(filename);
FTempData.clear;
end;

procedure TMRMrINI.RawToFile(filename : string);
begin
  FMainData.SaveToFile(filename);
end;

procedure TMRMrINI.RawToStream(output : TStream);
begin
  FmainData.SaveToStream(output);
end;

procedure TMRMrINI.RawFromFile(filename : string);
begin
  FMainData.LoadFromFile(filename);
end;

procedure TMRMrINI.RawFromStream(input : TStream);
begin
  FMainData.LoadFromStream(input);
end;

{Функці управління}
procedure TMRMrINI.ClearDatabase;
begin
{Очищення БД}
 FMainData.clear;
 FTempData.clear;
end;

procedure TMRMrINI.EraseSection(Section: string);
{Видалення секції}
var
```

```
 linestart, lineend, count : integer;
begin
linestart:=GetSectionstart(section);
lineend:=GetSectionEnd(section,linestart);
for count:=linestart to lineend do
 begin
 FMainData.Delete(linestart);
 end;
end;

procedure TMRMrINI.DeleteKey(Section, Key: string);
{Видалення ключа Ini файлу}
var
 pos : integer;
begin
Pos:=GetEntryPosition(section, Key);
FMaindata.Delete(pos);
end;

procedure TMRMrINI.ReadSections(Strings: TStrings);
{Читання Секції}
var
 position : integer;
 working, Working2 : string;
 Section : string;
begin
Section:='';
For position:=0 to fMaindata.count-1 do
 begin
 {Отримання ім'я секції }
 Working:=FMainData.strings[position];
 Working2:=copy(working, 1, Pos(Delimiter,working)-1);
 delete(working, 1, Pos(Delimiter,working));
 if Working2 <> Section then {зміна та додавання секції}
  begin
  Section:=Working2;
  strings.Add(Working2);
  end;
 end;
end;

procedure TMRMrINI.ReadSection(Section: string; Strings:
                                        TStrings);
{Читання всіх ключів секції та додавання їх до string list}
var
 position : integer;
 working, Working2, working3 : string;
 CSection : string;
 havefound : boolean;
begin
cSection:=''; havefound:=false;
For position:=0 to fMaindata.count-1 do
 begin
 {отримання ім'я секції }
 Working:=FMainData.strings[position];
 Working2:=copy(working, 1, Pos(Delimiter,working)-1);
 delete(working, 1, Pos(Delimiter,working));
 if Working2 = Section then
  begin
  havefound:=true;
  cSection:=Working2;
  Working3:=copy(working, 1, Pos(Delimiter,working)-1);
  Delete(working, 1, Pos(Delimiter,working));
  Strings.add(working3{+Valueequals+working});
  end;
```

```
 if working2 <> Section then
  begin
  cSection:=working2;
  if havefound then break;
  end;
 end;
end;

procedure TMRMrINI.ReadSectionValues(const Section: string; Strings: TStrings);
var
 position : integer;
 working, Working2, working3 : string;
 CSection : string;
 havefound : boolean;
begin
cSection:=''; havefound:=false;
For position:=0 to fMaindata.count-1 do
 begin
Working:=FMainData.strings[position];
 Working2:=copy(working, 1, Pos(Delimiter,working)-1);
 delete(working, 1, Pos(Delimiter,working));
 if Working2 = Section then
  begin
  havefound:=true;
  cSection:=Working2;
  Working3:=copy(working, 1, Pos(Delimiter,working)-1);
  Delete(working, 1, Pos(Delimiter,working));
  Strings.add(working3+Valueequals+working);
  end;
 if working2 <> Section then
  begin
  cSection:=working2;
  if havefound then break;
  end;
 end;
end;

{Фунції читання запису}
function TMRMrINI.ReadString(Section, Key, Vdefault :
                                    string): string;
var temp : string;
begin
temp:=GetValue(Section+Delimiter+Key);
if temp = StartDefault then temp:=VDefault;
Result:=temp;
end;

function TMRMrINI.ReadInteger(Section, Key : string; Vdefault : integer):
integer;
{Читання даних та перетворення їх до числа }
var
temp : string;
int : integer;
begin
temp:=GetValue(Section+Delimiter+Key);
if temp = StartDefault then int:=VDefault
 else int:=strtoint(temp);
Result:=int;
end;

function TMRMrINI.ReadBool(Section, Key : string): boolean;
{Читання значення формату boolean}
var
 temp : string;
begin
```

```
temp:=GetValue(Section+Delimiter+Key);
Temp:=Uppercase(temp);
if (temp = 'YES')
 or (temp = 'Y')
 or (temp = '1')
 or (temp = 'TRUE')
  then result:=true
 else Result:=false;
end;

function TMRMrINI.ReadFloat(Section, Key : string; Vdefault : extended):
extended;
{Читання значення формату float }
var
 temp : string;
 int : extended;
begin
temp:=GetValue(Section+Delimiter+Key);
if temp = StartDefault then int:=VDefault
 else
  try {Exception handler for converter}
  int:=strtofloat(temp);
  except
  on EConvertError do int:=0;
  end;
Result:=int;
end;

procedure TMRMrINI.WriteString(Section, Key, value :
                                        string);
var
 pos : integer;
begin
Pos:=GetEntryPosition(section, Key);
if Pos <> -1 then
 begin
 FMainData.strings[pos]:=Section+Delimiter+key+delimiter+value;
 end
else
 begin
 FMainData.add(Section+Delimiter+key+delimiter+value);
 end;
FMainData.Sort;
end;

procedure TMRMrINI.WriteInteger(Section, Key : string;
                                value : integer);
{Читання значення формату integer }
var
 pos : integer;
 temp : string;
begin
{Конвертація змінної }
temp:=inttostr(value);
Pos:=GetEntryPosition(section, Key);
if Pos <> -1 then
 begin
 FMainData.strings[pos]:=Section+Delimiter+key+delimiter+temp;
 end
else
 begin
 FMainData.add(Section+Delimiter+key+delimiter+temp);
 end;
FMainData.Sort;
end;
```

```
procedure TMRMrINI.WriteBool(Section, Key : string; value
                                        : boolean);
{Запис значення формату boolean}
var
 pos : integer;
 temp : string;
begin
{Конвертація змінної }
if value then temp:=BoolOne
 else temp:=BoolZero;
Pos:=GetEntryPosition(section, Key);
if Pos <> -1 then
 begin
 FMainData.strings[pos]:=Section+Delimiter+key+delimiter+temp;
 end
else
 begin
 FMainData.add(Section+Delimiter+key+delimiter+temp);
 end;
FMainData.Sort;
end;

procedure TMRMrINI.WriteFloat(Section, Key : string; value : extended);
{Запис значення формату float }
var
 pos : integer;
 temp : string;
begin
{Конвертація змінної}
temp:=FloatToStrF(value, ffGeneral, 18, 0);
Pos:=GetEntryPosition(section, Key);
if Pos <> -1 then
 begin
 FMainData.strings[pos]:=Section+Delimiter+key+delimiter+temp;
 end
else
 begin
 FMainData.add(Section+Delimiter+key+delimiter+temp);
 end;
FMainData.Sort;
end;

end. // Кінець файлу
```