Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
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 оптимізації структури інформаційного ресурсу”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22МЗ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Гаращенко Н.О.
«____» _____ 2023 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 123 *"Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

Гаращенко Наталії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу*

2. Керівник роботи *Дрєєва Ганна Миколаївна, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 36-13 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 оптимізації структури інформаційного ресурсу*

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

Гаращенко Н.О. Дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи оптимізації структури інформаційного ресурсу.

Метою розробки є дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу.

Об'єктом дослідження є процес оптимізації структури інформаційного ресурсу.

Предметом дослідження є методи оптимізації структури інформаційного ресурсу.

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

Результат роботи – програмна реалізація системи оптимізації структури інформаційного ресурсу.

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 RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, оптимізація, інформаційний ресурс

ABSTRACT

Harashchenko N.O. Research and software implementation of the optimization system of the information resource structure. 123 Computer engineering. 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 system of optimizing the structure of the information resource.

The purpose of the development is the research and software implementation of the system for optimizing the structure of the information resource.

The object of the research is the process of optimizing the structure of the information resource.

The subject of the study is methods of optimizing the structure of the information resource.

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

The result of the work is the software implementation of the system for optimizing the structure of the information resource.

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 RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, optimization, information resource

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЗЗУ	–	зовнішній запам'ятовувальний пристрій
ІР	–	інформаційний ресурс
ІС	–	інформаційні системи
ІТ	–	інформаційні технології
НС	–	нейронні мережі
ПАК	–	програмно-апаратний комплекс
МОСІР	–	метод оптимізації структури інформаційного ресурсу

ВСТУП

Актуальність теми. За оцінкою аналітиків, архіви корпоративної інформації подвоюється кожні два роки, причому 80% обсягу даних, збережених в електронному виді, доводиться на неструктуровану інформацію. При існуючій тенденції прогнозується подальше прискорення темпів росту, що ставить перед фахівцями в області інформаційних технологій (IT) завдання забезпечення функціонування інформаційних систем (ІС) із заданими параметрами.

У цей час активно ведуться роботи зі створення єдиних інформаційних просторів, що забезпечують тісну взаємодію територіально розподілених підрозділів за допомогою глобальних обчислювальних мереж, що припускає використання відкритих інформаційних систем. У зв'язку із цим до інформаційних систем пред'являються вимоги інтероперабельності, масштабованості, переносимості, продуктивності, надійності, інтеграції з іншими ІС. Проектування й експлуатація систем пов'язано з вибором состава технічних пристроїв, засобів зв'язку, структури й організації обчислювальної мережі, структури й організації зберігання інформаційного ресурсу (ІР). Саме тому особливий інтерес у цей час здобувають методи, які дозволяють оцінити параметри програмно-апаратного комплексу (ПАК), використовуваного для зберігання ІР.

У відомих роботах вирішувалися, як правило, завдання, пов'язані з підвищенням ефективності пошуку, передачі й аналізу інформації. Разом з тим, у цей час актуальні питання, пов'язані зі зберіганням ІР.

Аналіз практичних розробок показує, що пропонується широкий спектр програмно-апаратних комплексів зберігання, пошуку ІР. Але відсутня методика формування структури ІР, що створює проблеми ефективності їхнього застосування.

Потрібен комплексний аналіз ІС по якісних і кількісних характеристиках, обмеженням і цільовим критеріям з погляду оптимізації структури ІР, що сприяє збільшенню ефективності функціонування інформаційних систем, зниженню ризиків у діяльності організації, зниженню витрат на зберігання ІР.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу.

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

– Огляд існуючих систем оптимізації структури інформаційного ресурсу.

– Дослідження системи оптимізації структури інформаційного ресурсу.

– Програмна реалізація системи оптимізації структури інформаційного ресурсу.

Об'єктом дослідження є процес оптимізації структури інформаційного ресурсу.

Предметом дослідження є методи оптимізації структури інформаційного ресурсу.

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

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

– Удосконалено метод оптимізації структури інформаційного ресурсу.

– Розроблено вітчизняний продукт оптимізації структури інформаційного ресурсу, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі оптимізації структури інформаційного ресурсу.

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 роботи збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу, 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

Рівень розвитку інформації поступово підвищувався, інформаційні технології призвели до швидкого розвитку інших галузей промисловості, швидкий прогрес високотехнологічних галузей поступово став важливим показником розвитку регіону країни, а також напряму, при цьому інформаційні технології також є важливим показником оцінки ступеня модернізації та рівня економічного розвитку [2]. Для розвитку всього народного господарства і життя необхідно ефективне використання управління земельними ресурсами та земельної природної інформації, управління земельними ресурсами стало ключовим завданням. Управління земельними ресурсами відіграло дуже важливу роль у довгостроковому розвитку нашої економіки та суспільства. Поряд із нагальними потребами економічного розвитку та використання земельних ресурсів і мінеральних ресурсів у процесі дедалі серйозніших проблем ці проблеми охоплюють незаконне використання та інші ключові суперечливі питання, такі як окупація землі. У той же час рівень забудови наших міст має терміново піднятися на вищий рівень, через що дефіцит земельних ресурсів і мінеральних ресурсів створює нові виклики в процесі управління земельними ресурсами, тож як покращити швидкість земельних і використання мінеральних ресурсів стало ключовим питанням, яке терміново потребує вирішення в наш час, і відділам управління земельними ресурсами необхідно ефективно поєднувати наступні технології:

– інформаційні технології,
– технології управління інформацією про землю;
– технології управління інформацією про ресурси.

Використання вищезазначених технологій може підвищити ефективність управління інформацією про землю та мінеральні ресурси, а надходження

інформаційних технологій поступово стало напрямком удосконалення розвитку управління безпекою земель [3].

З настанням нової ери комплексне управління земельними ресурсами зараз переживає комплексне реформування та комплексне будівництво. Просування в основному здійснюється з трьох аспектів:

– інформаційних технологій;

– управління інформацією про землю та наукового менеджменту;

– стандартизованого управління інформацією про земельні ресурси.

Земельні ресурси охоплюють різноманітну природну інформацію через прозору систему управління інформацією для досягнення наукового управління. Прозора система може покращити землю.

Система прозорості може підвищити ефективність управління земельними ресурсами, посилити рівень обміну інформацією та протистояти суспільному розвитку, який є більш сприятливим для влади приватного сектору. Згідно з кожною інформацією та аналізом попиту та враховуючи поточну ситуацію розвитку інформаційних технологій, у цьому документі пропонується комплексна система інформаційного обслуговування управління будівництвом міських земельних ресурсів, реалізуючи функціональні модулі в кожному бізнесі управління [4].

Створено платформу обміну інформацією, оптимізовано модель бізнес-процесів управління інформацією та допомога відповідним відділам, які використовують систему, підвищити свою ефективність. В той же час, щоб уникнути відносно ізольованих даних, які не можуть бути ефективно використані, використання інформаційних систем може ефективно використовувати дані, загальна структура яких є моделлю побудови системи з вимогами до додатків як ядром [5].

Земельні ресурси є природним скарбом соціального розвитку та виживання країни, і ефективне управління земельними ресурсами країни може забезпечити сприятливу гарантію для виживання та розвитку населення в цілому, тоді як ефективне використання розвитку природних ресурсів може зробити

значний внесок економічного розвитку нашої країни [6 – 11]. Повне й розумне використання та охорона земельних ресурсів може забезпечити реалізацію політики сталого розвитку. В країні розвиток національного управління земельними ресурсами є основною вимогою національної модернізації, а також неминучою тенденцією, оскільки управління земельними ресурсами охоплює великий обсяг інформації, задіяної в величезній, включаючи сферу роботи інформації, тому для забезпечення ефективності та достовірності земельних ресурсів та інформації має вкрай необхідне практичне значення.

1.2 Область застосування

Областю застосування є інформаційні ресурси. З появою електронного уряду на основі інформації, управління земельними ресурсами також висуває відповідні виклики та можливості.

Інформаційні системи можуть ефективно покращувати розвиток, управління, дослідження та використання земельних ресурсів.

Система електронного урядування може допомогти земельним ресурсам отримати точний аналіз і дослідження, детальний доступ для розуміння стану ресурсів і здатність динамічно розуміти розвиток інформаційних змін у щоденних даних та інформації, що постійно змінюються, а також осягати останні тенденції у земельних ресурсах, тенденції розвитку аналізу та моніторингу ринкових ресурсів, щоб надати клієнтам надійну підтримку прийняття рішень для забезпечення надійної безпеки даних за лаштунками, посилення планування управління ресурсами та надання землі та ресурсів для подальшого розвитку плану в нашій країні.

Проектування та побудова системи управління інформацією електронного уряду має фіксувати фактичні потреби після польових досліджень шляхом налаштування бізнес-процесів і системи для адаптації до бізнес-процесів, а також використовувати уніфіковану мову моделювання для аналізу та моделювання потреб у ресурсах відповідно, щоб гарантувати, що система може задовольнити

потреби користувачів і розробити структурну схему моделі для задоволення інженерів-розробників програмного забезпечення. Архітектура системи може адаптуватися до розвитку технологій, тоді як система може легко адаптуватися.

Архітектуру системи можна адаптувати до розвитку технологій, у той час як систему можна легко оновлювати та розширювати, а вторинну розробку та постобслуговування програмного продукту можна розглядати, щоб забезпечити тривалий життєвий цикл програмного продукту.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу, 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

Інформаційні технології та електронний уряд у Європі та Сполучених Штатах взяли на себе провідну роль у розвитку та появі Сполучених Штатів та інших ранніх розвинутих країн для створення повної системи бізнес-систем. Інформаційні технології були використані для того, щоб зайняти лідерство у сфері державного управління. На початку 1970-х Сполучені Штати вперше використали інформаційні технології, щоб вийти на сферу високошвидкісного управління інформацією, сучасну магістраль для наукового управління, тоді як у 1990-х інформаційні технології широко використовувалися в сфері управління, і дослідницький досвід поступово набув всебічного розвитку. Інформаційна технологія використовується для побудови браузерно-серверної моделі системи управління, а високопродуктивний сервер використовується як серверне ядро обробки на терміналі обробки системи для побудови ієрархічної та централізовано керованої інформаційної системи [12].

Електронне урядування використовується на Заході вже давно і має широкий розвиток, а також накопичений глибокий досвід розвитку [13]. Що стосується управління земельними ресурсами, то американські вчені в 1998 році запропонували концепцію цифрової землі, тобто цифрову структуру та компонування глобальної географічної інформації. Цифрова Земля – це схема мережевої інформатизації інформації про земельні ресурси, віртуалізоване зберігання реалістичної інформації про земельні ресурси, а разом з цим і відповідне народження геоінформаційної системи. З розвитком національної земельної інформатизації наприкінці двадцятого століття західні країни, такі як

Канада, приєдналися одна за одною до всебічної побудови цифрової землі, а наприкінці двадцятого століття різні публікації, карти та земля наукова інформація з різних підрозділів була інтегрована, тому користувачі можуть запитувати, переглядати та оцінювати [14].

У Міністерстві земель та ресурсів, яке активно підтримує міста та муніципалітети, інтегроване управління текстовою інформаційною системою центру земельних ресурсів сприяє побудові інформації про земельні ресурси, побудові інформаційної системи земельних ресурсів та інформаційному моделюванню земельних ресурсів, які може забезпечити результативність та ефективність бізнесу. За останні роки управління земельними ресурсами досягло достатнього прогресу та досягнень, але порівняно з інтернаціоналізацією інформації про земельні ресурси все ще існує певний прогалин у розвитку досліджень інформаційних технологій управління земельною інформацією, які є пізніми, основні інформаційні ресурси є відносно бідні, а національні ресурси та земельний капітал багаті, але реальний ступінь відсутності певного менеджменту, обмежений рівень управлінських знань уряду та відсутність певних інвестицій у реформи, мужність і сміливість у поєднанні з відносною відсутністю таланту в у цій сфері для будівництва земельних ресурсів інформаційні технології не проводять постійних досліджень та накопичення, відсутність досвіду будівництва призводить до відставання інфраструктури інформаційних технологій. Загальний рівень інформаційних технологій низький [15]. Ці прогалини також потребують поступового усунення наступними вченими-дослідниками, і низка проблем, які зараз зустрічаються, в основному показані нижче.

Географічна побудова інформації на основі даних все ще знаходиться на відсталій стадії, побудова географічної інформації почалася пізно, збір і введення географічної інформації мають певні проблеми в розвитку, і ці проблеми обмежують подальший розвиток побудови управління географічними ресурсами. В той же час, існує відносна нестача талантів у цій галузі, ступінь розуміння

лікування управління земельними ресурсами є відносно відсталим, а теоретичні дослідження будівництва земельних ресурсів відносно відсутні [16].

В даний час дослідження та розробка інформаційної платформи земельних ресурсів є актуальною темою, велика кількість вчених поступово визнали важливість цього питання, і велика кількість досліджень і обговорень може допомогти крок за кроком керувати земельними ресурсами. правильний шлях розвитку. Ефект очевидний. Управління інформацією про земельні ресурси є більш упорядкованим. Передова технологія для охоплення всього міського будівництва в мережевому середовищі повністю використовує [17] динамічний дистанційний моніторинг землекористування та дистанційне управління кадастровою інформацією. В останні роки дані про зміну земельної площі будуть використовуватися для отримання найбільш точних записів. Місцеве самоврядування було введено вчасно, придатне для побудови політики. На окремих територіях міста застосовано землевпорядні інформаційні системи. Передові технології зараз просуваються та популяризуються, що є сильним поштовхом до розвитку земельної інформації та технічної команди та зростання. Ця робота буде зосереджена на розробці та впровадженні інформаційних систем земельних ресурсів [18]. Дослідження інформаційних технологій цього періоду є пізніми, базові інформаційні ресурси відносно бідні, країна багата ресурсами та земельним капіталом, але реальний ступінь відсутності певного менеджменту, обмежений рівень урядового пізнання менеджменту та відсутність певних інвестицій у реформи, сміливість, а також сміливість, у поєднанні з відносною відсутністю таланту в цій галузі, для побудови земельних ресурсів інформаційні технології не зробили наполегливих досліджень і накопичення. Відсутність досвіду будівництва призводить до відставання інформаційно-технологічної інфраструктури. Загальний рівень інформаційних технологій низький. Інформаційні технології спонукають до швидкого розвитку інших галузей і безперервного розвитку наукових знань, пов’язаних із встановленням наукового розвитку [19].

Завдяки впровадженню землекористування в систему землеустрою, новому закону про землеустрій країни, плануванню землекористування з охорони орних земель та ряду регуляторних заходів у всіх аспектах швидкий прогрес високотехнологічної промисловості став кроком за кроком важливий показник регіону країни в розвитку, а також напрямок. Продукти автоматизації офісу популяризуються крок за кроком. Системи управління земельною інформацією використовуються для забезпечення хороших операційних інструментів та інтерфейсів для управління земельними ресурсами [20]. Необхідно контролювати обсяг документообігу. Багатопланова, багатовідомча координація. Та спільні консультації для завершення розподілу завдань для подальшого підвищення ефективності та забезпечення якості роботи. Відповідно до вимог побудови єдиного земельного ресурсу та земельного сектору працівники спільно використовують набір джерел даних, які полегшують створення науково обґрунтованої структури даних, структури послуг та середовища виконання [21]. Система може задовольнити потреби користувачів і розробити структурну схему моделі, яка задовольняє інженерів-розробників програмного забезпечення, а архітектура системи може адаптуватися до розвитку технологій, у той час як система може легко оновлювати та розширювати та обробляти вторинну розробку програмних продуктів, а також післяобслуговування забезпечити тривалий життєвий цикл програмних продуктів [22].

2.2 Обґрунтування 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 оптимізації структури інформаційного ресурсу.

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

Рівень забудови міста терміново потребує більш високого рівня, що спричиняє дефіцит земельних ресурсів та мінеральних ресурсів та висуває нові виклики в процесі управління земельними ресурсами, тож як покращити швидкість використання землі та корисних копалин ресурсів стала ключовою проблемою, яка нагально потребує вирішення в даний час, і відділу управління земельними ресурсами необхідно ефективно поєднувати:

– інформаційні технології;

– технології управління інформацією про землю;

– технології управління інформацією про ресурси [23–26].

Використання вищезазначених технологій може підвищити ефективність управління інформацією про землю та мінеральні ресурси, а приплив інформаційних технологій поступово став напрямом удосконалення розвитку управління земельною безпекою.

Поступово зростає рівень інформаційного розвитку. Інформаційні технології призвели до стрімкого розвитку інших галузей. Стрімкий прогрес індустрії високих технологій крок за кроком став важливим індикатором розвитку регіону країни, а також напряму. Ця технологія є не лише результатом опадів, але й прагненням до передових результатів.

Вчені часто хочуть використовувати цей новий продукт, щоб зробити більший крок у своїй власній галузі, тому в рамках цієї роботи прагну ефективного використання управління земельними ресурсами та природною інформацією землі.

Управління земельними ресурсами стало ключовим завданням. Управління земельними ресурсами відіграє дуже важливу роль у

довгостроковому розвитку нашої економіки та суспільства. Коли виникають нові явища, вони також викликають деякі нові труднощі. Нові труднощі створюють нові проблеми для польового персоналу, ці проблеми охоплюють інші ключові суперечливі питання, такі як незаконне використання та окупація землі.

Таким чином, для забезпечення ефективності та точності земельних ресурсів та інформації має дуже необхідне практичне значення, у процесі роботи, для забезпечення чесності та відкритості інформації, забезпечуючи при цьому точність та прозорість інформації. Обмін інформацією має також гарантувати, що всі відділи можуть ефективно використовувати, розглядаючи підтримку системи, сподіваючись на те, щоб прагнути до простоти та створення ідеального та розумного робочого механізму та потоку інформаційних даних. Наступ інформаційної ери також може стати революцією в режимі управління урядовими департаментами, що не тільки означає, що бізнес уряду поступово стає прозорим і відкритим, але також, з іншого боку, може допомогти відповідним державним службовцям використовувати інформаційні технологічні продукти для досягнення повсякденної справи дозволили управління та навіть використання інформаційних технологій для побудови браузерно-серверного режиму системи управління та використання високопродуктивних серверів у обробному терміналі системи. Інформаційна технологія використовується для побудови системи керування в режимі браузера-сервера, а високопродуктивний сервер використовується як ядро внутрішньої обробки для побудови ієрархічної та централізовано керованої інформаційної системи, як показано на рисунку 3.1.

Рівень розвитку інформаційних технологій поступово підвищується, інформаційні технології призвели до швидкого розвитку інших галузей, а також є встановлення наукового розвитку наукових знань, пов’язаних із безперервним розвитком. Завдяки впровадженню землекористування в систему землеустрою, новому закону про землеустрій країни, плануванню землекористування для охорони орних земель та ряду регуляторних заходів у всіх аспектах, швидкий прогрес високотехнологічної промисловості крок за кроком став важливий

показник регіону країни в розвитку та напрямку та зрілих та передових інформаційних систем управління в застосуванні, тоді як інформаційні технології також є ступенем модернізації, рівнем економічного розвитку важливих показників оцінки, географічних інформаційних систем та продуктів автоматизації офісу, який має крок за кроком процес, щоб стати популярним. Система управління земельною інформацією для управління земельними ресурсами забезпечує хороші робочі інструменти та інтерфейси, використовуючи кращі методи та способи для виконання відмінної роботи.

Управління земельними ресурсами відіграло дуже важливу роль у довгостроковому розвитку нашої економіки та суспільства. Це новостворене соціальне явище часто також породжує нові проблеми. У кількох галузях це так. З такими проблемами стикається і управління земельними ресурсами. Ці проблеми охоплюють інші ключові суперечливі питання, такі як незаконне використання та зайняття землі. У той же час, рівень будівництва наших міст терміново потребує підвищення, що спричиняє дефіцит земельних ресурсів та мінеральних ресурсів. В процесі управління земельними ресурсами виникають нові виклики, тому як підвищити швидкість використання землі та мінеральних ресурсів стало ключовим питанням, яке нагально потребує вирішення на даний момент, а відділам управління земельними ресурсами необхідно ефективно поєднувати інформаційні технології, технологія управління інформацією про землю та технологія управління інформацією про ресурси. Використання вищевказаних технологій може підвищити ефективність управління інформацією про землю та мінеральні ресурси. Приплив інформаційних технологій поступово стає напрямком удосконалення розвитку управління земельною безпекою.

3.2 Розробка структурної схеми

Для розрахунку вхідних параметрів структури ІР пропонується функціонально-орієнтований підхід, відображений на структурній схемі (рисунок 3.1), що виконується в наступні етапи:

– збір статистики використання ІР і-м процесом (**етап 1**):

v_i – сумарний обсяг, Мб;

k_i – число користувачів;

z_i – число операцій уведення-виводу за од. часу;

v_{zi} – обсяг інформації, переданої за одну операцію уведення-виводу, Мбіт;

– визначення состава ІР для і-го процесу за критерієм припустимі фінансові витрати (**етап 2**) і відповідному кожному класу параметрів:

n_{ij} – період зміни цінності ІР для і-го процесу й j-го класу ІР;

m_i – період існування ІР, використовуваного і-м -процесом;

– обробка відомостей (**етап 3**) з використанням методів математичної статистики, що дозволяють визначити прогнозні значення параметрів ІР на майбутній період: обсяг, число операцій уведення-виводу, число користувачів;

– формування ІР у вигляді багаторівневої структури (**етап 4**). Число й зміст рівнів визначається на підставі відомостей, отриманих на етапі 1 і 2, а також на підставі вимог до ІС. Для кожного рівня задається множина процесів Е і відповідних їм ІР, розташовуваних на р-м рівні.

$$E_p = \{A_k, v_{kj} (k = 1,q)\}, \quad (3.1)$$

де q – число процесів.

Розрахунок значень параметрів (**етап 5**) для кожного рівня обчислюється по формулах (3.2) – (3.4), які є вихідними даними для проектування оптимальної структури зберігання ІР.

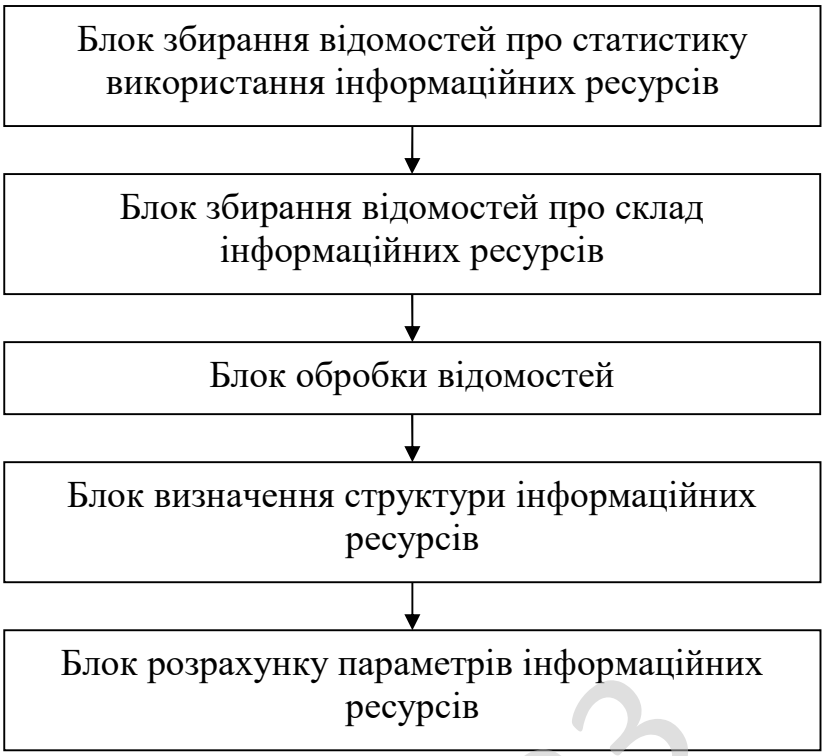

Рисунок 3.1 – Структурна схема системи

Уведемо наступні позначення, що характеризують технічні параметри:

– P – загальна кількість рівнів IP;

– V_p – сумарний обсяг ресурсу для p-го рівня, Мб;

– Ps_p – пропускна здатність для p-го рівня, Мбіт/сек;

– Z_p – число операцій уведення-виводу для p-го рівня;

– t – час доступу, сек;

– k_{raid} – коефіцієнт використання додаткового простору RAID масивом.

Сумарний обсяг ресурсу для p-го рівня обчислюється за формулою:

$$V_p = k_{raid} * \sum_{k=1}^{q} v_k . \quad (3.2)$$

Пропускна здатність для p-го рівня обчислюється за формулою:

$$Ps_p = \left[\sum_{k=1}^{q} (z_k * v_{zk}) \right] / t \ . \quad (3.3)$$

Число операцій уведення-виводу для p-го рівня обчислюється за формулою:

$$Z_p = \sum_{k=1}^{q} z_k \,. \tag{3.4}$$

Дані параметри використовуються для проектування оптимальної структури ІР, вибору апаратних ресурсів, їхнє налаштування, конфігурування, оцінки витрат.

Розробку плану структури ІР розглянемо як завдання, що полягає в оптимізації параметрів багаторівневого середовища зберігання ІР із заданими локальними характеристиками кожного рівня й у той же час об'єднаними сукупністю обмежень на все середовище зберігання. Оптимальним планом є номенклатура дискових масивів і кількість зовнішніх запам'ятовувальних пристроїв (ЗЗУ), складових дисковий масив, при мінімальній сумарній вартості зберігання. У такій постановці ми приходимо до завдання математичного програмування із блокової (багаторівневої) структурою.

З обліком вищесказаного дамо формалізований опис завдання. Нехай маємо Р-рівнів і m_p, p = 1..P параметрів, що характеризують ресурс, наявність кожного і-го параметра становить найменше b_{pi} і найбільше B_{pi}, i = 1…m_p, значення у відповідних одиницях вимірів. Ці параметри призначені для формування n_p типів дискових масивів. Кожна одиниця j-го типу дискового масиву містить a_{ij} одиниць і-го параметра ресурсу. Потрібно визначити, які типи дискових масивів і яка кількість дисків необхідно для формування багаторівневого середовища зберігання з найкращими показниками для прийнятого критерію оптимальності – вартості F.

Позначимо через x_{pj} – кількість одиниць j-го типу дискових масивів на р-м рівні, тоді математичну постановку завдання можна записати у вигляді:

$$F = \sum_{p=1}^{P} \sum_{j=1}^{n_p} v_{pj} c_{pj} x_{pj} \rightarrow \min, \tag{3.5}$$

при обмеженнях

$$\sum_{p=1}^{P} \sum_{j=1}^{n_p} a_{ij} x_{pj} \le B_i \,, \quad \text{i} = 1..m_0; \tag{3.6}$$

$$\sum_{p=1}^{P}\sum_{j=1}^{n_p} a_{ij}x_{pj} \geq b_i \quad \text{i} = 1..\text{m}_0; \tag{3.7}$$

$$\sum_{p=1}^{P}\sum_{j=1}^{n_p} x_{pj} \leq b_0 \; ; \tag{3.8}$$

$$\sum_{j=1}^{n_p} a_{ij}x_{pj} \leq B_i \; , \; \text{i} = 1..\text{m}_\text{p}, \; \text{p} = 1..\text{P}; \tag{3.9}$$

$$\sum_{j=1}^{n_p} a_{ij}x_{pj} \geq b_i \; , \; \text{i} = 1..\text{m}_\text{p}, \; \text{p} = 1..\text{P} \; ; \tag{3.10}$$

$$x_j \in \Omega, \qquad \text{j} = 1..\text{n}_\text{p}, \; \text{p} = 1..\text{P}, \tag{3.11}$$

де

– P – загальна кількість локальних блоків;

– m_0 – число обмежень у блоці-зв'язуванню;

– n_p – число змінних в p – м локальному блоці;

– m_p – число обмежень в p-м локальному блоці;

– v_{pj} – обсяг диска j-типу в p-м локальному блоці;

– c_{pj} – вартість зберігання інформації на диску j-типу в p-м локальному блоці;

– b_0 – загальна кількість дисків зберігання ІР;

– b_{pi} – найменше значення параметра ІР;

– B_{pi} – найбільше значення параметра ІР;

– Ω – множина цілих, позитивних чисел.

Умови (3.6), (3.7), (3.8) описують блок-зв'язування, (3.9), (3.10) – окремі блоки (рівні), (3.11) – умова цілочисельного значення змінної x_{pj}.

Аналіз випадків використання

Аналіз варіантів використання – це дослідження на функціональному рівні, яке фіксує функціональність системи з точки зору макросу. Варіанти використання включають його визначення, усі необхідні дії для виконання порядку варіантів використання, стандартні поведінки в різних деформаціях, загальну поведінку всіх ненормальних випадків та очікувані реакції, не розкриваючи припущення про те, що внутрішня структура системи визначає

послідовну поведінку. З точки зору користувача, наведені вище випадки можуть бути ненормальними. З точки зору системи, вони повинні бути описані та додані до розгляду справи. Більш конкретно, використання випадку не є обов'язковим для функціональної специфікації, але також показує та демонструє його вимоги в процесі ілюстрації в UML, використовуючи випадки, представлені еліпсом, у якому виконання кожного варіанту використання є незалежним від іншого. варіанти використання, хоча через той факт, що коли випадки є спільними, об'єкти можуть генерувати неявні кореляції між варіантами використання з причин виконання між варіантом використання, кожен варіант використання, описаний у вертикальному функціональному блоці, виконання цього функціонального блоку може бути переносяться та змішуються разом з іншими футлярами.

Функціональні функції мають зв'язок «багато-до-багатьох» з учасниками системи, де ролі користувачів з різними привілеями можуть виконувати однакові функціональні функції, тоді як кожен клас користувачів ролей може виконувати кілька функціональних функцій. Існують різні складні зв'язки між функціями, такими як залежність, включення та розширення. Розбиваючи функцію на кілька функцій, реалізується ієрархічний зв'язок функціональної структури діаграми варіантів використання. На діаграмі варіантів використання учасники використовуються для системних функцій. На діаграмі варіантів використання, яка має чітку систему учасників, ви можете одночасно бачити чіткі функції кожного учасника. Діаграма варіантів використання дуже виразна. На початку аналізу вимог до виконання ми можемо вивести діаграму варіантів використання та перевірку користувача, і, таким чином, далі в кожному функціональному модулі підсистеми необхідний аспект діаграми варіантів використання у функціональному моделюванні системи.

Розглядаючи сферу побудови інформації про земельні ресурси, поступово було визначено та узагальнено набір відповідних методів управління місцевими земельними ресурсами, в той час як за допомогою популярних сьогодні

інформаційних технологій. У фактичній роботі земельна столиця міста Далянь дійсно зробила великий крок вперед, порівняно з попередньою ефективністю роботи, яка значно зросла, яка була побудована та введена в експлуатацію в деяких пілотних містах та земельних адміністраціях округу. Завдяки використанню системи інформаційних технологій ефект очевидний. Управління інформацією про земельні ресурси є більш упорядкованим. І є накладання передових технологій. Охоплюючи все міське будівництво в мережевому середовищі, повністю використовує динамічний дистанційний моніторинг землекористування та дистанційне управління кадастровою інформацією. В останні роки дані про зміну земельної площі будуть використовуватися для отримання найбільш точних записів. Місцеве самоврядування було впроваджено належним чином, придатне для побудови політики. Інформаційна система землеустрою була використана в певних районах міста, де зараз просуваються та популяризуються передові технології, що стало сильним поштовхом для розвитку земельної інформації та технологічної команди та зростання інформаційних ресурсів землі. Цей документ буде зосереджений на розробці та впровадженні інформаційної системи земельних ресурсів.

Управління земельними ресурсами відіграє дуже важливу роль у довгостроковому розвитку як нашої економіки, так і суспільства. Поряд з гострою потребою економічного розвитку, процес використання землі та мінеральних ресурсів створив дедалі серйозніші проблеми, які охоплюють інші ключові суперечливі питання, такі як незаконне використання та зайняття землі. У той же час рівень забудови наших міст має терміново піднятися на вищий рівень, через що дефіцит земельних ресурсів і мінеральних ресурсів створює нові виклики в процесі управління земельними ресурсами, тож як покращити швидкість земельних і використання мінеральних ресурсів стало ключовим питанням, яке нагально потребує вирішення в даний час, і відділам управління земельними ресурсами необхідно ефективно поєднувати інформаційні технології, технології управління інформацією про землю та технології управління

інформацією про ресурси. Використання вищезазначених технологій може підвищити ефективність управління інформацією про землю та мінеральні ресурси, а надходження інформаційних технологій поступово стає напрямком вдосконалення розвитку управління земельною безпекою.

У той же час, щоб уникнути відносно ізольованих даних, які не можуть бути ефективно використані, використання інформаційних систем може ефективно використовувати дані, загальна структура яких є моделлю побудови системи з вимогами до додатків як ядром. Земельні ресурси є природним скарбом соціального розвитку та виживання країни, і ефективне управління земельними ресурсами країни може забезпечити сприятливу гарантію для виживання та розвитку більшості людей, тоді як ефективне використання та розвиток природних ресурсів може зробити видатний внесок в економічний розвиток нашої країни. Повне та раціональне використання та охорона земельних ресурсів може забезпечити реалізацію політики сталого розвитку, а розвиток національного управління земельними ресурсами є основною вимогою для модернізації країни, а також неминучою тенденцією на національному рівні, оскільки земля Управління ресурсами охоплює великий обсяг інформації, задіяної в широкому діапазоні, включаючи обсяг роботи інформації.

Дизайн специфікації бази даних

Детальний проект бази даних головним чином полягає в концептуалізації атрибутів об’єктів, залучених до системи, розробці відповідного типу для кожного атрибута, побудові інформаційної таблиці бази даних для кожного типу об’єкта, записі змін інформації про об’єкт під час роботи систему, своєчасно синхронізувати з базою даних і підтримувати нормальну роботу системи. Відповідно до моделі зв’язку сутності на етапі концептуального проектування бази даних, для бази даних потрібна певна робота з концептуалізації, і існують деякі теорії для вивчення зв’язку між членами бази даних, які в основному досліджують і вивчають зв’язок між об’єктами, і існує залежність відносини, резервне копіювання та організація. Детальний проект бази даних головним

чином полягає в концептуалізації атрибутів об’єктів, залучених до системи, розробці відповідного типу для кожного атрибута, створенні інформаційної таблиці бази даних для кожного типу об’єкта, записі змін інформації про дані об’єкта під час роботи систему, своєчасно синхронізувати з базою даних і підтримувати нормальну роботу системи.

Згідно з кожною інформацією та аналізом попиту та враховуючи поточну ситуацію розвитку інформаційних технологій, у цьому документі пропонується комплексна система інформаційного обслуговування управління будівництвом міських земельних ресурсів, яка створює платформу обміну інформацією шляхом реалізації функціональних модулів у кожному бізнесі управління , оптимізуючи модель бізнес-процесів управління інформацією та допомагаючи відповідним відділам, які використовують систему, підвищити ефективність роботи, і в той же час, щоб уникнути відносно ізольованих даних, які не можуть бути ефективно використані, використання інформаційних систем може ефективно використовувати дані, загальна структура яких є моделлю побудови системи з вимогами до додатків як ядром.

Оскільки управління земельними ресурсами охоплює великий обсяг інформації та включає всю інформацію в рамках роботи, дуже необхідно забезпечити достовірність і точність земельних ресурсів та інформації, забезпечити справедливість і відкритість інформації в процесі роботи, забезпечувати точність і прозорість інформації, щоб усі відділи могли ефективно використовувати її для обміну інформацією, а також для обробки системи та спрощення роботи з обслуговування. І ефективність робочого середовища можна покращити.

Тест оптимізації системи

Тестування чорного ящика часто використовується для перевірки загальної функціональності програмного забезпечення та функціональності програмного забезпечення з графічним інтерфейсом і зовнішньою структурою програми, а метод тестування чорного ящика вимагає, щоб тестування модулів,

які поділяють систему, було явним, а потім перевірити вхідні та вихідні дані для кожного модуля, а потім фактично перевірити вихідні результати, щоб порівняти фактичний вихід системи з ідеальним виходом користувача, і якщо вони не відповідають, то можна прийняти попереднє рішення, що функція функціонального модуля є проблематичною, з винятками та зазвичай для тестування методу визначення винятку. Для аналізу виявлення потрібен наступний метод тестування білої скриньки. При аналізі з точки зору макро-мікро, методи тестування програмного забезпечення можна розділити на два основні тестування чорного ящика, а також тестування білого ящика. Функціональне тестування так званого чорного ящика використовує програмну систему, яка має непрозору чорну скриньку, з точки зору макросу, яка здатна працювати лише з вхідними даними для системи, щоб визначити вихідні дані системи та чи очікує користувач результати повинні бути послідовними. Тестування чорного ящика в основному зосереджено на системі введення та виведення, системі від системи. Перспектива тестування білої скриньки та методології тестування охоплюють код, щоб охопити всі гілки, усі оператори програмного коду для всіх можливих помилок, повний обхід тесту.

Звичайно, проходження всіх гілок може спростити кількість тестів якості і є гідним питанням для обговорення. Тестування білого ящика широко використовується в роботі з розробки програмного забезпечення, допомагаючи розробникам прагнути знаходити дефекти програмного забезпечення та вчасно їх виправляти. Тестування білого ящика прозоре. Тестування білого ящика в основному походить від тестування, тестування внутрішніх програмних продуктів від деталей і виправлення проблем, виявлених під час процесу тестування. У двох фазах життєвого циклу програмного забезпечення тестування програмного забезпечення, яке зазвичай відбувається після написання кожного модуля, також називається модульним тестуванням. Кодування та модульне тестування належать до однієї фази життєвого циклу програмного забезпечення. Система програмного забезпечення проводить різноманітні комплексні тести

після закінчення цієї фази, яка є іншою фазою життєвого циклу програмного забезпечення

Test case – процес тестування сценарію тестування програмного забезпечення. Тестові випадки використовуються для визначення нормального функціонування програмного продукту. Тестові випадки часто вибираються репрезентативними для деяких типових даних, часто аналітиками даних критичних точок системи, схильних до збоїв точок для аналізу та вилучення, щоб розробити тестові випадки, систему відповідно до різних функціональних модулів і тестових випадків для розробки різних програми. Існують певні функціональні модулі тестування програмного забезпечення, які часто є цільовими, комплексними та репрезентативними.

Тестування програмного забезпечення, контрольний тест, структура змінних параметрів системного програмного продукту та випадок виконання тесту спостерігають за роботою програмного забезпечення. Результати, якщо попередні умови програмного забезпечення є однаковими, можуть вказувати на те, що функціональні модулі в нормальній роботі тесту варіантів використання потребують більше випадків для тестування та перевірки часом і витримують перевірку часом. Система працює стабільно та є надійною системою. Якщо результати тестування та очікувані результати мають розбіжності, показуючи, що існують певні проблеми з функціональними модулями, систему необхідно протестувати та перевірити більш детально, щоб знайти проблеми, які існують в існуючій системі, і вирішити логічні проблеми програми.

Використання інформаційних систем дозволяє ефективно використовувати дані, загальна структура яких є моделлю побудови системи з вимогами до додатків як ядром. Інформаційна система може ефективно покращити розвиток, управління, дослідження та використання земельних ресурсів.

Система електронного уряду може допомогти земельним ресурсам отримати точний аналіз і дослідження, детальний доступ для розуміння стану

ресурсів, динамічне розуміння розвитку інформаційних змін у щоденних даних та інформації, що все більше змінюються, а також оволодіти останніми тенденціями у земельних ресурсах, і тенденція розвитку аналізу та моніторингу ринкових ресурсів, щоб надати клієнтам надійну підтримку прийняття рішень для забезпечення надійної безпеки даних за лаштунками, посилити планування управління ресурсами та надати землю та ресурси для подальшого розвитку плану в Китаї, що може надавати точні інформаційні послуги для державних менеджерів і водночас може відповідати національним тенденціям розвитку. У цьому документі ми провели детальні тести кожного модуля системи, які обробляють модуль керування користувачами системи.

Виберіть деякі типові значення як значення функції, щоб перевірити функціональність системи. Значення системних функцій повинні представляти та відображати функції програмного забезпечення системи. Мати повне розуміння шаблону проектування, функцій і загальної архітектури системи. І протестувати важливі модулі системи. Тестові випадки системи часто потребують ретельного проектування, поєднання з інформаційною системою в різних прикладних сценаріях для комплексного тестування. Тестові випадки часто охоплюють загальні сценарії системи, а також деякі спеціальні випадки нестандартних сценаріїв і комплексне тестування функціональності системи. Використання ефективного способу компенсації нестачі наглядових повноважень також є важливою частиною міської системи зв’язків із надзвичайними ситуаціями. У цьому документі після поглибленого дослідження та аналізу розроблено набір тестових прикладів, які відповідають фактичним сценаріям, і функціональні тести проводяться на кількох модулях системи, щоб гарантувати якість тестування, а також довіру до системи.

Цей тест програмного забезпечення застосовує теорію тестування чорної скриньки та білої скриньки, і система інтегрована та всебічно протестована з програмним модулем як основним блоком. Земельні ресурси охоплюють різноманітну природну інформацію через прозору систему управління

інформацією для досягнення наукового управління. Прозора система може підвищити ефективність управління земельними ресурсами та посилити ступінь обміну інформацією. Стикаючись із суспільним розвитком, він є більш сприятливим для влади приватного сектору, відповідно до аналізу кожної інформації та попиту та з урахуванням поточної ситуації розвитку інформаційних технологій. У цьому документі пропонується побудова інтегрованих інформаційних служб управління міськими земельними ресурсами. У цьому документі ми пропонуємо комплексну систему інформаційного обслуговування управління міськими земельними ресурсами та створюємо платформу обміну інформацією шляхом реалізації функціональних модулів у кожному бізнесі управління. Ми протестували підвищення його ефективності, і результати показали що загальна ефективність покращилася більш ніж на 2 години порівняно з порожнім контролем. У той же час була оптимізована модель бізнес-процесів управління інформацією, що також допомогло відповідним відділам, які використовують систему, підвищити ефективність своєї роботи, і в той же час, щоб уникнути неефективного використання відносно ізольованих даних, використання інформаційних систем дозволяє ефективно використовувати дані.

Побудова інтегрованої системи управління інформаційними службами міських земельних ресурсів створила платформу для обміну інформацією шляхом реалізації функціональних модулів у кожному бізнесі управління, одночасно оптимізуючи модель бізнес-процесу для управління інформацією та в той же час допомагаючи відповідним відділам, які використовують систему підвищити ефективність своєї роботи.

Після польових досліджень та аналізу систему управління інформацією про земельні ресурси можна розділити на чотири функціональні модулі, які включають:

– управління інформацією про основні земельні ресурси;

– управління погодженням землекористування земельних ресурсів;

– управління інформацією про офіційні документи;

– та управління офісним бізнесом.

І ці модулі можуть ефективно завершити роботу з управління земельними ресурсами. Повне та розумне використання та охорона земельних ресурсів може забезпечити реалізацію політики сталого розвитку, а розвиток управління земельними ресурсами є основною вимогою національної модернізації та неминучою тенденцією в країні, а також тому, що управління земельними ресурсами охоплює велику кількість інформації, що включає широкий спектр інформації, у тому числі інформацію в межах роботи, дуже необхідно для забезпечення ефективності та точності земельних ресурсів та інформації. При аналізі з точки зору макро-мікро, методи тестування програмного забезпечення можна розділити на два основні тестування чорного ящика, а також тестування білого ящика. Так зване тестування чорного ящика, функціональне тестування - це програмна система як непрозорий чорний ящик. З точки зору макросу, він здатний працювати лише з вхідними даними для системи, у щоденній мінливій інформації, динамічному розумінні розвитку інформаційних змін і осягнути останні тенденції в земельних ресурсах, тенденції аналізу та моніторингу ринкові ресурси, щоб забезпечити клієнтам надійну підтримку прийняття рішень. Це потужна гарантія фонових даних для клієнтів, які забезпечують надійну підтримку прийняття рішень.

3.3 Розробка функціональної схеми

Розглянемо процес розробки інтелектуального класифікатора, побудованого на базі нейронної мережі (НМ). Описується й обґрунтовується структура нейронної мережі для класифікації ІР, алгоритм її навчання. Під класифікацією ІР розуміється віднесення об'єкта до одному із заздалегідь відомих класів: критична, важлива, неважлива.

Класифікація ІР здійснюється з використанням програмного комплексу й включає наступні етапи обробки інформації (рисунок 3.2).

Перший і другий етапи формують метадані IP, тобто дані про IP, які формуються частково користувачем, частково програмою.

Для здійснення класифікації виділені ознаки, що характеризують групу, до якої належить IP:

– x_1 – обсяг;

– x_2 – інтенсивність використання;

– x_3 – ким створюється;

– x_4 – інтенсивність зміни;

– x_5 – дата створення;

– x_6 – найменування підрозділу;

– x_7 – тип ресурсу;

– x_8 – дата зміни;

– x_9 – найменування програми.

Третій етап, попередня обробка вхідних даних, є найбільш трудомістким етапом класифікації й включає:

– кодування, тому що використовуваний алгоритм класифікації працює тільки із числовою інформацією, тоді як ознаки x_3, x_6, x_7, x_9 приймають значення з дискретного набору. У цьому випадку пропонується використовувати двійковий тип кодування $n \rightarrow n$, n – число категорій;

– нормування, тобто приведення до єдиного масштабу числової змінної на діапазон розкиду її значень. Для нормування пропонується використовувати статистичні характеристики середнє й дисперсія, тому що ознаки x_1, x_2 x_4, x_5 мають великий розкид значень;

– зниження розмірності, тобто відбір найбільш інформативних ознак. У ході експериментів виявлені ознаки, які найбільш впливають на точність роботи класифікатора: x_3, x_6, x_7, x_2, x_5.

Рисунок 3.2 – Функціональна схема системи

Четвертий етап виконує класифікацію ІР на підставі вхідного вектора $X = \{x_3, x_6, x_7, x_2, x_5\}$ до одному із заданої множини класів Y:

– критична;

– важлива;

– неважлива.

Етапи 3 – 4 виконуються автоматично з використанням інтелектуального класифікатора. Розробка інтелектуального класифікатора включає формування навчальної множини, вибір оптимальної архітектури НМ і алгоритму її навчання.

Для побудови класифікатора використовується багатошаровий персептрон, а точніше його двошарова реалізація (рисунок 3.3).

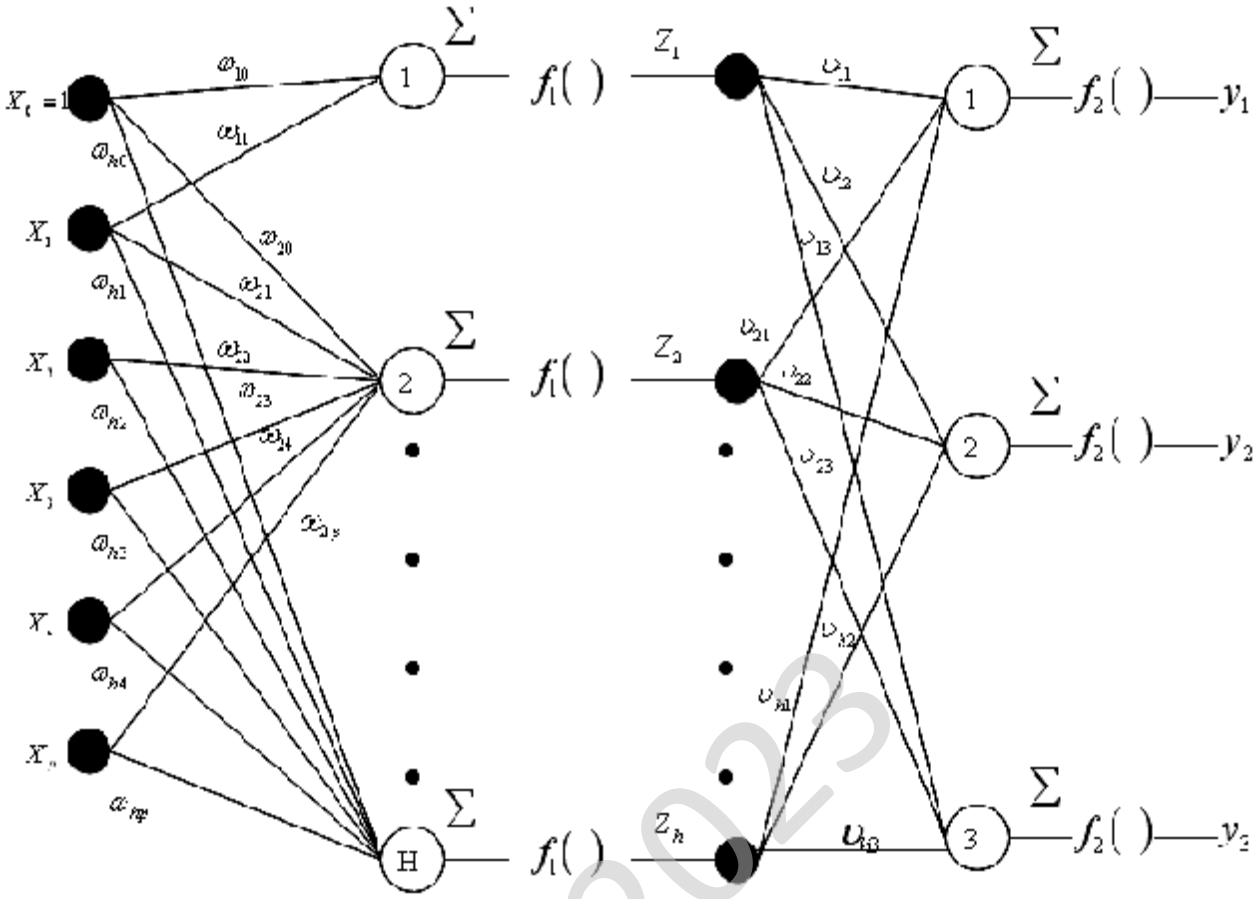

Рисунок 3.3 – Архітектура нейронної мережі – класифікатор IP

На вхід нейронної мережі подається p-мірний вектор метаданих $X = \{x_i,\ i = 1,2,\ldots p\}$...Оптимальна кількість нейронів на схованому шарі H підбирається експериментально.

У вихідному шарі число нейронів збігається із числом класів M, тому що пропонується для кожного класу визначити свій нейрон. Між вхідним і схованим шарами, а також між схованим і вихідним шарами використовується повнозв'язна структура.

З урахуванням цих доповнень опишемо прийняті на рисунку 3.5 позначення:

– p – розмірність вихідних даних (кількість метаданих, використовуваних для класифікації);

– H – число нейронів на схованому шарі;

– x_i – компонента вхідного вектора ознак, i = 1,…,p;

– $x_0 = 1$ – постійний вплив, використовуваний для роботи нейронної мережі;

– w_{ji} – вагарні коефіцієнти між вхідним і схованим шарами, i = 0,1,…,p; j = 1,…,H;

– vjk – вагові коефіцієнти між схованим і вихідним шарами, j = 0,1,…,H; k = 1…m;

– zj – значення виходу j-го нейрона схованого шару; $z_0 = 1$, j = 1,…,H;

– yk – значення виходу k-го нейрона мережі, k = 1…m...

Значення виходів z_j і y_k визначаються по формулах (3.10), (3.11).

$$z_j = f_1(\sum_{i=0}^{p} w_{ji} x_i) \tag{3.10}$$

$$y(x) = f_2(\sum_{j=1}^{H}(v_j z_j) = f_2(\sum_{j}^{H} v_j f_1(w_{j1}x_1 + w_{j2}x_2 + ... + w_{jp}x_p + w_{j0}) + v_0) \ . \tag{3.11}$$

Як функція активації $f_1(x)$ для нейронів схованого шару й $f_2(x)$ для нейронів на виході мережі використовуємо сигмоїдну функцію, позначивши її як f(x):

$$f(x) = \frac{1}{1 + e^{-ax}} . \tag{3.12}$$

Функціонування нейронної мережі залежить від величин, що характеризують зв'язки, тому, задавшись архітектурою нейронної мережі, необхідно знайти оптимальні значення всіх змінних коефіцієнтів w.

Оцінка якості роботи нейронної мережі визначається величиною функції обчислення помилки E, як середньоквадратичне відхилення поточних виходів від необхідних для кожної навчальної пари із заданої вибірки навчальної множини S: {X,Y}.

Процес навчання розглядається як завдання багатомірної оптимізації, що складається в пошуку оптимальних значень вагових коефіцієнтів w:

$$E(w) = \frac{1}{2}\sum_{k}(y_{k,s} - d_{k,s})^2 , \tag{3.13}$$

де

y – поточне значення k-нейрона для s-навчальної пари;

d – необхідне значення k-нейрона для s – навчальної пари.

У випадку визначення помилки для всього множини навчальних пар величина E обчислюється за формулою:

$$E = \frac{1}{N}\sum_{s=1}^{N} E_s(w) \to \min, \tag{3.14}$$

де N – потужність навчальної вибірки.

Тоді навчання нейронної мережі зводиться до знаходження вагових коефіцієнтів w, які давали мінімальне значення цільової функції E для всієї навчальної вибірки:

$$E(w) = \frac{1}{N}\sum_{s=1}^{N}(\frac{1}{2}\sum_{k=1}(y_{k,s} - d_{k,s})^2 = \frac{1}{N}\sum_{s=1}^{N}\left[\frac{1}{2}\sum_{k=1}^{m}(f_2\sum_{1}^{5}(v_j f_1(w_{j1}x_1 + w_{j2}x_2 ... + w_{jp}x_p + w_{j0}) + v_0) - d_{k,s})^2)\right] \to \min. \tag{3.15}$$

Пошук мінімуму функції здійснюється з використанням наступних алгоритмів: методу зворотного поширення помилки, методу сполучених градієнтів, генетичного алгоритму. У таблиці 3.1 наведені результати навчання при наступних параметрах моделі НМ p = 5, H = 8, m = 3, N = 63.

Щонайкраще завдання навчання вирішують генетичний алгоритм і метод сполучених градієнтів. Для поліпшення роботи генетичного алгоритму в процесі експериментів підібрані параметри, при яких досягаються прийнятні параметри швидкості збіжності і якості рішення.

Проведені дослідження підтвердили ефективність застосування нейронних мереж для класифікації IP. При певних налаштуваннях НМ можна домогтися результатів, коли ймовірність правильного розпізнавання становить 97%. Помилки виникають тільки на 2 векторах з 63.

Таблиця 3.1 – Показники алгоритмів навчання нейронної мережі

Показники	Генетичний алгоритм	Алгоритм зворотної поширення помилк	Метод сполучени градієнтів
t, час збіжності, з	18	12	20
E_n, середня сумарна квадратична помилка	0,005121	0,010913	0,134201
E_{min}, максимальна сумарна квадратична помилка	0,036629	0,236629	0,199979
K, кількість ітерацій	39000	4425	18833
Якість роботи, % правильних відповідей	97%	94%	98%

Використання інтелектуального класифікатора дозволить значно знизити тимчасові витрати, пов'язані з обслуговуванням IP, знизити втрати, пов'язані з людським фактором.

Далі з використанням отриманих моделей, методик і алгоритмів:

1. Проведено оптимізацію структури інформаційного ресурсу.

2. Отримано прогнозну оцінку росту обсягу IP і визначені параметри оптимальної структури IP з урахуванням змін обсягів IP.

3. Отримано оцінку ефективності пропонованих рішень оптимізації структури IP.

Оптимізація IP включає комплекс організаційних і апаратно-програмних рішень, спрямованих на забезпечення необхідного рівня функціонування ІС. У цьому зв'язку розроблений ряд організаційних мір, спрямованих на керування IP, що підвищить ефективність його використання.

Розроблено методичні рекомендації, що регламентують процес формування й зберігання IP.

Розроблено модель захисту, що включає перелік груп користувачів і відповідні їм права доступу.

Оцінка ефективності пропонованих рішень по оптимізації структури ІР виконана з урахуванням витрат, які несе організація протягом існування ІР.

Визначення витрат по використанню ІР проведено на підставі сукупної вартості володіння (СВВ) інформаційними ресурсами. Для оцінки СВВ використовується модель, що відбиває повний перелік статей витрат, пов'язаних із впровадженням і обслуговуванням ІС протягом строку функціонування.

$$C_{CCB} = \sum_{1}^{7} c_k \ , \qquad (3.16)$$

де c_k– витрати на k-у статтю витрат ІС.

В умовах росту обсягів ІР і збільшення числа інформаційних процесів, збільшуються витрати, пов'язані з адмініструванням і людським фактором, питома вага яких становить 42%.

Проведений аналіз інформаційних систем і відповідних інформаційних ресурсів дозволив виявити динаміку росту й динаміку зміни цінності ІР. За останні 3 роки ІР збільшився в 3,5 рази, з них на неструктуровану інформацію доводиться 58%. Разом з тим, аналіз ІР показав, що 30% інформації не використовується більше 12 місяців.

Проведені розрахунки СВВ зберігання ІР до оптимізації й після оптимізації його структури на період 2018-2023 рр. показали наступні результати (таблиця 3.2).

Аналіз витрат і оцінка ефективності оптимізації структури ІР, що ставляться до розробки, впровадженню й функціонуванню ІС, зроблений на основі даних відділу інформаційних технологій освітньої установи й відомостей про вартість устаткування провідного виробника систем зберігання даних.

Таким чином, використання оптимізації структури ІР дозволить знизити поточні витрати, пов'язані з його обслуговуванням, а також одноразові витрати на апаратні ресурси.

Таблиця 3.2 – Показники ефективності використання оптимальної структури IP

№ статті	Найменування статті	% витрат	Вартість, тис.грн.	
			до оптимізації	після оптимізації
1	Програмно-апаратне забезпечення	0,25	1 560,00	1 252,00
2	Адміністрування	0,21	1 310,40	420,67
3	Підтримка	0,16	998,40	801,28
4	Розробка	0,06	374,40	300,48
5	Комунікації	0,04	249,60	200,32
6	Людський фактор	0,21	1 310,40	736,18
7	Простої	0,07	436,80	350,56
	СВВ	1	6 240,00	5 008,00
	СВВ період 2011-2015гг		24 960,00	15 299,44
	Ефективність		1,00	0,61

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.4. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

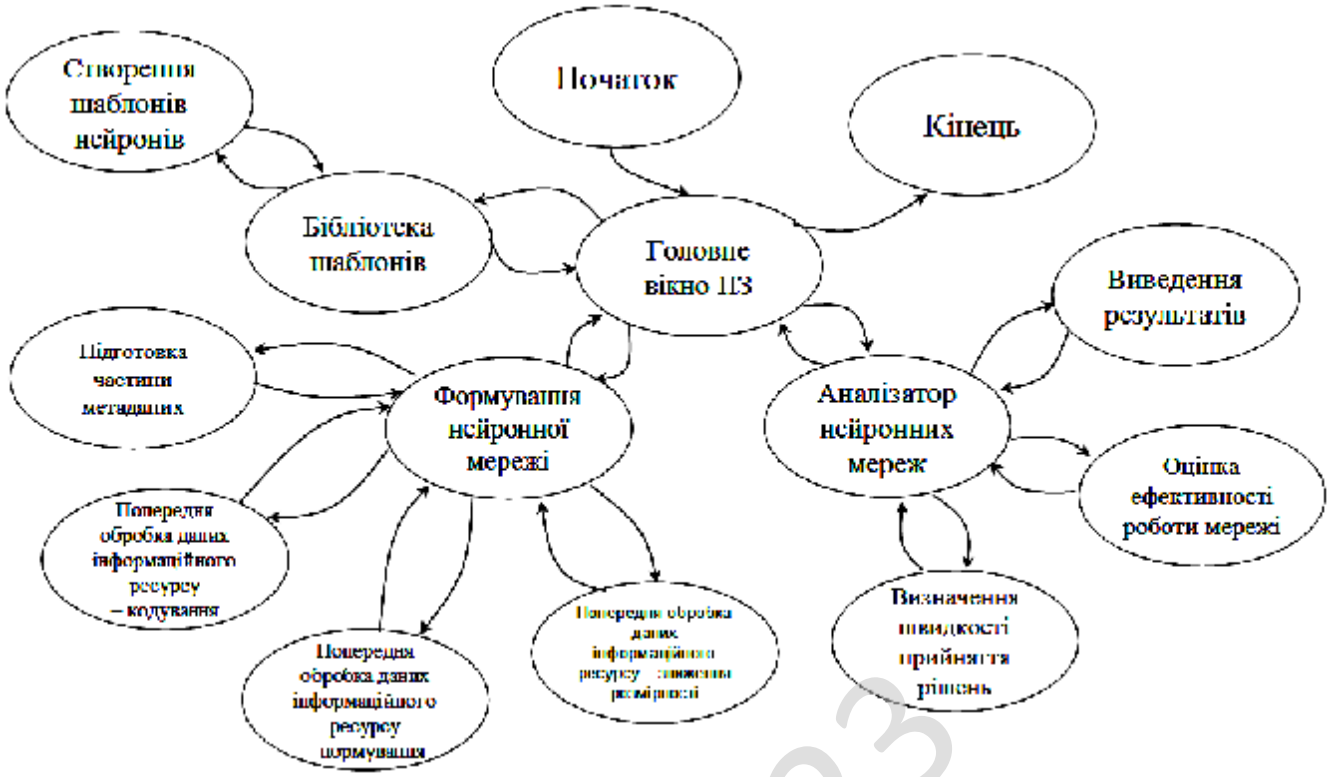

Рисунок 3.4 – Діаграма взаємодії процесів

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю методу оптимізації структури інформаційного ресурсу.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

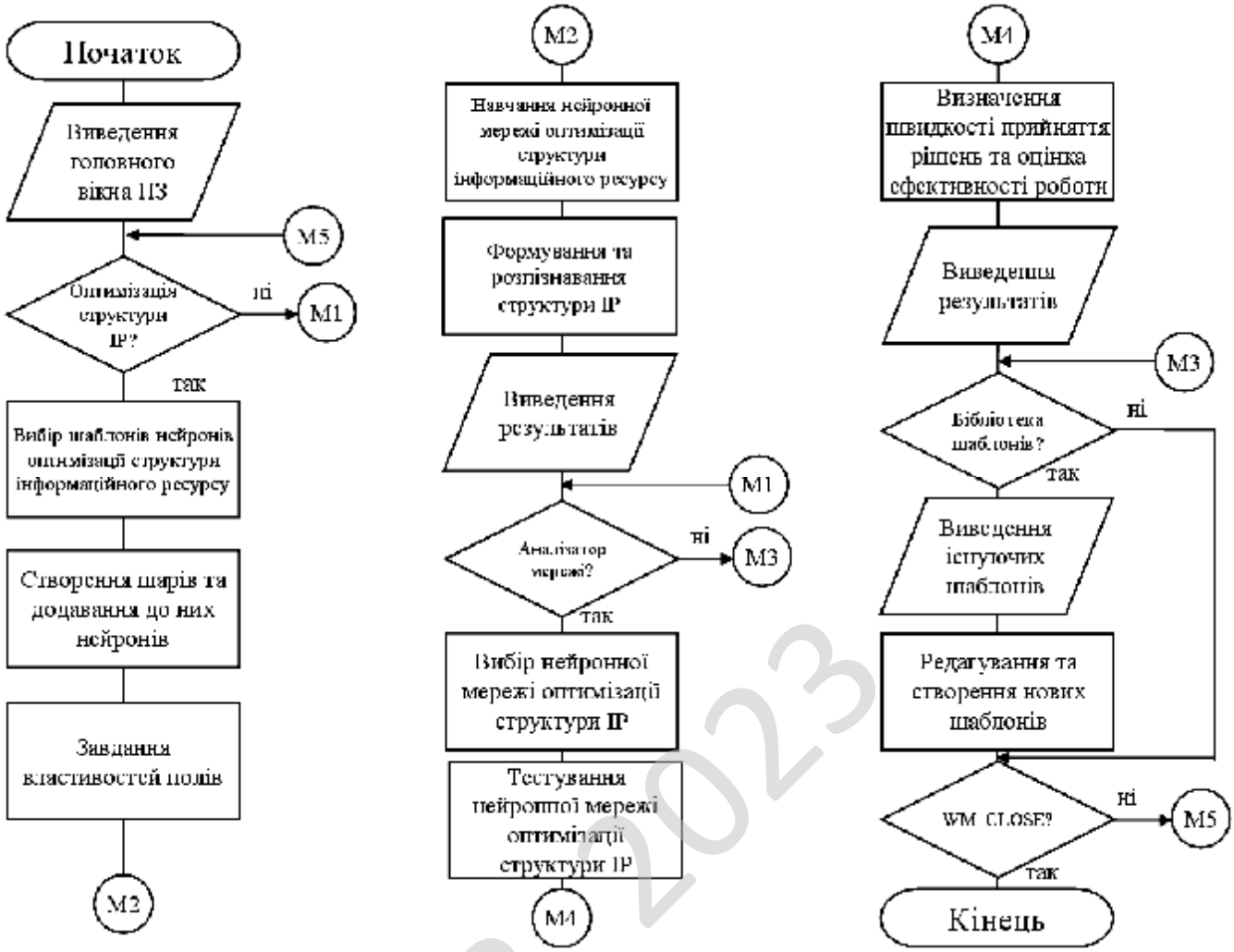

Рисунок 4.1 – Блок-схема основної програми

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись

один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

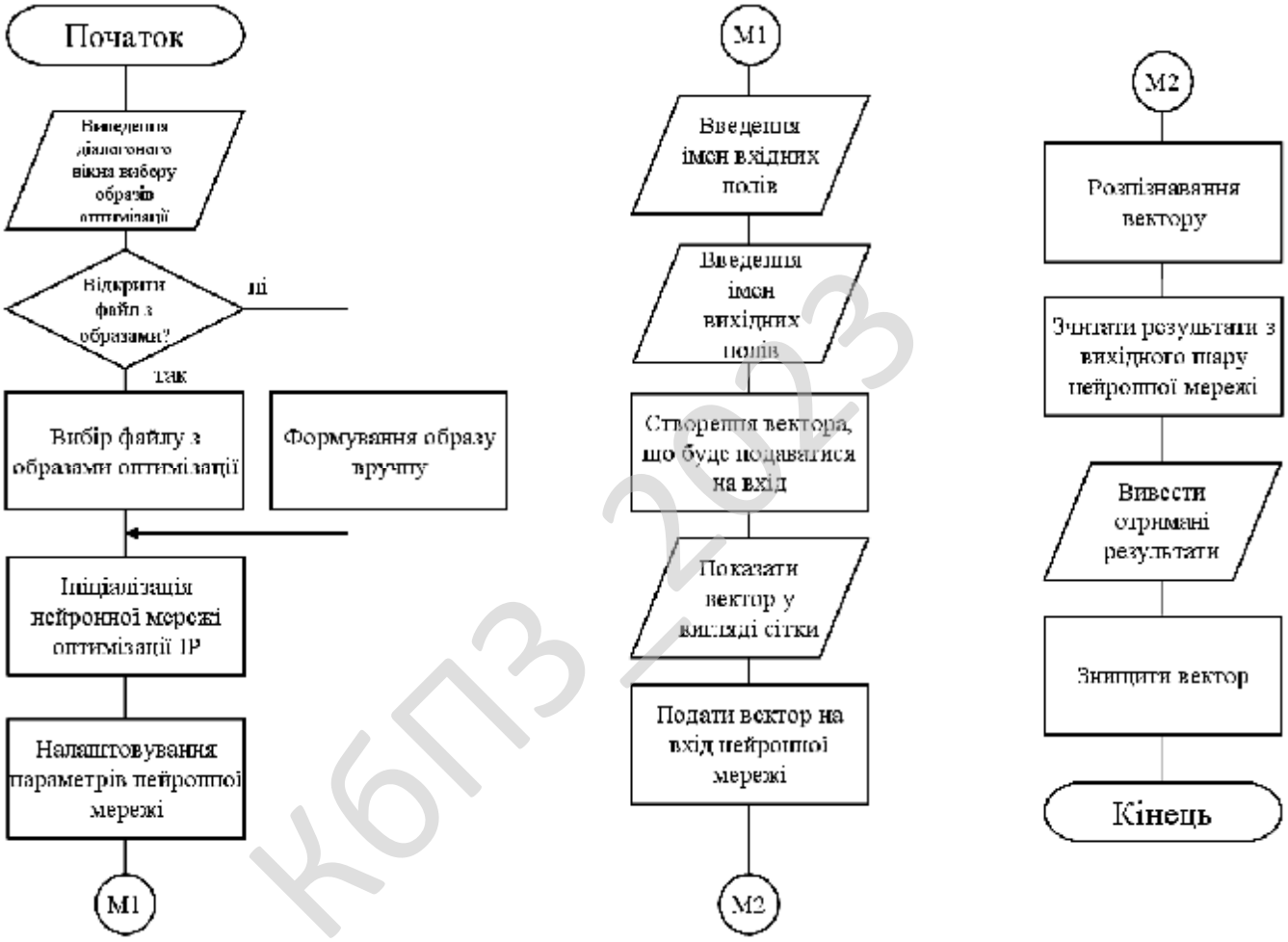

Рисунок 4.2 – Блок-схема роботи підпрограми

Приведемо опис нотифікації ряду функцій, які використовуються у розробленій, у результаті виконання магістерської роботи системи. Опис експортованих функцій.

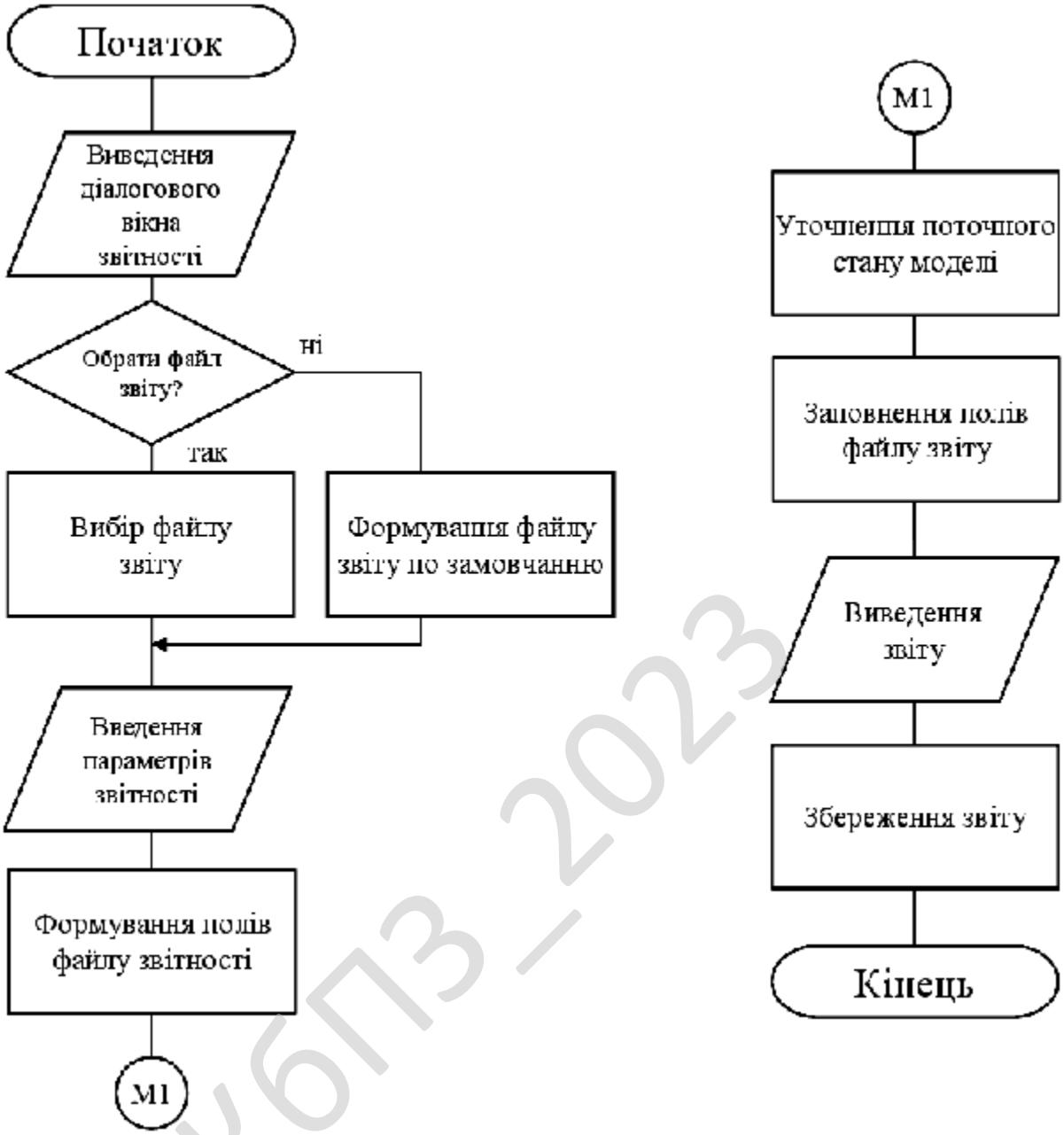

Рисунок 4.3 – Блок-схема роботи підпрограми

Робота з диском та робота зі сховищем:

1. Відкрити сховище загальних даних.

MOCIP_OpenStorage(lpName, dwTypes);

LpName – ім'я загального файлу-сховища даних методу оптимізації структури інформаційного ресурсу (далі MOCIP).

DwType – тип дій. Може приймати комбінацію з наступних значень:

– OS_CREATE – Створити нове сховище.

– OS_OPEN – Відкрити існуюче сховище. Файли зі сховища даних MOCIP відписуються в тимчасову директорію.

Повертає дескриптор відкритого сховища даних MOCIP або NULL у випадку помилки:

– сховище не існує (при використанні прапора OS_OPEN);

– при відкритті існуючого сховища даних MOCIP на диску перебувають файли, що збігаються по ім'ю з файлами сховища даних MOCIP.

2. Закрити сховище загальних даних.

MOCIP_CloseStorage(hStorage, dwFlag);

HStorage – дескриптор сховища даних MOCIP.

DwFlag – тип дій. Може приймати комбінацію з наступних значень:

– CS_DELETE – закрити й видалити сховище.

– CS_SAVE – закрити зі збереженням.

– CS_FORSE_SAVE – при закритті сховища даних MOCIP всі прикріплені до нього файли заносяться в сховище.

Повертає TRUE у випадку успіху або FALSE у випадку помилки.

3. Видалити сховище.

MOCIP_DeleteStorage (lpName);

LpName – ім'я сховища даних, що видаляється, MOCIP.

Повертає TRUE у випадку успіху або FALSE у випадку помилки:

– сховище не існує.

Робота з файлами

4. Зберегти файл у сховище.

MOCIP_WriteFileToStorage (hStorage, hFile, lpNameInStorage);

HStorage – дескриптор сховища даних MOCIP.

HFile – дескриптор файлу.

lpNameInStorage – ім'я файлу зберігається в загальному сховищі.

Файл приписується до сховища й при закритті сховища даних MOCIP буде в нього занесений. Сам файл диска видаляється.

Повертає TRUE у випадку успіху або FALSE у випадку помилки.

5. Одержати дескриптор файлу зі сховища даних MOCIP.

MOCIP_ReadFileFromStorage (hStorage, lpName);

HStorage – дескриптор сховища даних MOCIP.

LpName – ім'я файлу, що витягається із загального сховища даних MOCIP (під яким він записаний у сховище).

Повертає файлу у випадку успіху або NULL у випадку помилки:

– файлу з таким ім'ям у сховище немає.

6. Відкрити файл.

MOCIP_OpenFreeFile(hStorage, lpName, dwFlag);

HStorage – дескриптор сховища даних MOCIP до якого буде приписаний файл. Якщо NULL, файл до сховища не прив’язується.

lpName – ім'я файлу.

wdType – тип даних. Може приймати комбінацію з наступних значень:

– OSF_CREATE – створити новий файл.

– OSF_OPEN – відкрити існуючий файл.

– CSF_READ – відкрити файл на читання.

– OSF_WRITE – відкрити файл на запис.

– OSF_BINARY – відкрити файл у двійковому виді.

– OSF_IN_MEMORY – не виводити на диск.

– OSF_TEMPORARY – тимчасовий.

Значення покажчику, що повертається, на дані або NULL у випадку помилки:

– Файлу з таким ім'ям не існує (при використанні прапора OSF_ OPEN).

7. Закрити файл.

MOCIP_CloseFreeFile(hFile, dwFlag)

hFile – дескриптор файлу.

dwFlag – указує на дії вироблені при закритті.

Може приймати комбінацію з наступних значень:

– CSF_SAVEDISK – закрити зі збереженням на диску. Файл відкреплюється від сховища даних MOCIP.

– CSF_SAVESTORAGE – закрити зі збереженням у загальному сховищі.

– CSF_DELETE – закрити й видалити файл із відкріпленням від сховища даних MOCIP.

Значення, що повертається: TRUE у випадку успіху.

8. Записати дані на диск.

MOCIP_WriteToFile (hFile, lpData, dwSize);

hFile – покажчик отриманий від Open.

lpData – покажчик на зберігаються данные, що.

dwSize – число зберігаємих байт. Повертає число записаних байт.

9. Прочитати дані з диска.

MOCIP_ReadFromFile (hFile, lpData, dwSize);

hFile – покажчик отриманий від Open;

lpData – покажчик на зчитувальні дані;

dwSize – число зчитувальних байт; Повертає число прочитаних байт.

10. Перемістити покажчик.

MOCIP_SeekFilePointer (hFile, dwBytes, dwFrom);

hFile – покажчик отриманий від MFOpen.

dwBytes – число байт на яке відбувається зсув.

dwFrom – прапор, звідки відбувається зсув. Може приймати одне з наступних значень:

– FS_END – з кінця файлу.

– FS_BEGIN – з початку файлу.

– FS_CUR – з поточного значення. Повертає позицію, починаючи з початку файлу.

11. Прочитати поточний покажчик. Повертає позицію, починаючи з початку файлу.

MOCIP_TellFilePointer (hFile);

hFile – дескриптор файлу, отриманий від Open;

dwBytes – число байт на яке відбувається зсув.

12. Скинути кешовані дані на диск.

MOCIP_FlushFile(hFile);

hFile – дескриптор, отриманий від Open;

Робота з пам'яттю

13. Одержати покажчик на блок пам'яті.

MOCIP_AllocMemory (dwSize, dwFlag);

DwSize – розмір блоку.

DwFlag – атрибути пам'яті. Може приймати одне з наступних значень:

– MAF_GPTR – одержати покажчик на пам'ять. Пам'ять обнуляється. Значення, що повертається – покажчик на пам'ять.

– MAF_GHND – одержати покажчик на глобальну пам'ять. Пам'ять обнуляється. Значення, що повертається – покажчик на глобальний об'єкт.

– MAF_GALL_GMEM_FIXED – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_MOVEABLE – описаний у прапорах GlobalAlloc.

– MAF_GALL_GPTR – описаний у прапорах GlobalAlloc.

– MAF_GALL_GHND – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_DDESHARE – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_DISCARDABLE – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_LOWER – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_NOCOMPACT – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_NODISCARD – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_NOT_BANKED – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_NOTIFY – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_SHARE – описаний у прапорах GlobalAlloc.

– MAF_GALL_GMEM_ZEROINIT – описаний у прапорах GlobalAlloc.

14. Перевиділити блок пам'ять.

MOCIP_ReAllocMemory (hMemory, dwSize, dwFlag);

hMemory – дескриптор блоку пам'яті.

dwSize – реаллокуємий розмір.

dwFlag – параметр, може приймати наступні значення:

– MRF_NEW_MEMORY – виділяє новий блок, копіює в нього вміст старого й звільняє старий.

– MRF_GALL_GMEM_DISCARDABLEGPTR – описаний у прапорах GlobalReAlloc.

– MRF_GALL_GMEM_MOVEABLE – описаний у прапорах GlobalReAlloc.

– MRF_GALL_GMEM_NOCOMPACT – описаний у прапорах GlobalReAlloc.

– MRF_GALL_GMEM_ZEROINIT – описаний у прапорах GlobalReAlloc.

15. Звільнити блок пам'яті.

MOCIP_FreeMemory(hMem);

hMem – звільняється память, що.

16. Закріпити глобальний об'єкт і одержати покажчик на пам'ять. Повертає покажчик на пам'ять або NULL.

MOCIP_LockMemory(hMem);

hMem – покажчик на глобальний об'єкт.

17. Відкріпити глобальний об'єкт.

MOCIP_UnlockMemory(hMem);

hMem – покажчик на глобальний об'єкт.

18. Записати дані з виділеного блоку пам'яті на диск.

MOCIP_WriteMemoryToFile(hMem, lpName);

hMem – покажчик на пам'ять або глобальний об'єкт.

lpName – ім'я файлу, куди зберігати.

Повертає число записаних байт.

19. Виділити блок пам'яті потрібного розміру й зчитати дані з диска до пам’яті.

ReadMemFromFile (PChar lpName, lphMem);

lphMem – (out) посилання на покажчик на пам'ять або глобальний об'єкт.

lpName – (in) ім'я файлу, звідки читати.

Повертає число зчитаних байт.

20. Записати дані з виділеного блоку в сховище.

MOCIP_ReadMemoryFromFile(lpName, phMem);

phMem – покажчик на пам'ять або глобальний об'єкт.

lpName – ім'я, під яким зберігати в сховище.

Повертає число записаних байт.

21. Виділити блок пам'яті потрібного розміру й уважати в нього дані зі сховища даних MOCIP.

MOCIP_ReadMemoryFromStorage(hStorage, lpName, phMem);

hStorage – дескриптор сховища даних MOCIP.

phMem – (out) посилання на покажчик на дескриптор пам'яті, створеної в процесі читання.

lpName – (in) ім'я, під яким вони зберігаються у сховищі.

Повертає число зчитаних байт.

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 7624:2014 («Калина»). Одним із основних алгоритмів симетричного блокового шифрування, що використовуються в Україні, є ДСТУ 7624:2014 («Калина»), який визначає сучасний алгоритм симетричного блокового перетворення для забезпечення конфіденційності й цілісності інформації при її обробці та встановлює режими його роботи.

В алгоритмі шифрування даних «Калина» використовуються криптографічні перетворення, які відповідають сучасним вимогам до рівня криптостійкості та швидкодії.

Даний стандарт розроблено з урахуванням існуючих та потенційних загроз, подальшого інтенсивного розвитку інформаційних технологій та необхідності активного використання протягом кількох наступних десятиліть.

Стандарт блокового симетричного шифрування ДСТУ 7624:2014 визначає десять різних режимів роботи, що широко поширені відповідно до міжнародних стандартів ISO/IEC 10116:2006.

Це спрямовано на забезпечення широкого застосування ДСТУ 7624:2014, у тому числі для захисту інформації, що передається комп'ютерними мережами, прозорого шифрування жорстких дисків і змінних носіїв, електронних документів, ключових даних.

Ефективність реалізації систем, засобів та протоколів криптографічного захисту інформації в інформаційно-телекомунікаційних системах різного призначення може бути забезпечена саме наявністю такої кількості режимів роботи алгоритму.

До блокового шифру «Калина» ставляться такі вимоги: високий рівень криптографічної стійкості з достатнім запасом у разі появи нових атак протягом тривалого часу; висока швидкодія програмної реалізації на сучасних та перспективних платформах; компактність програмної та програмно-апаратної реалізації; можливість ефективної інтеграції декількох алгоритмів в одному засобі криптографічного захисту; прозорість проектування, консервативний підхід до забезпечення стійкості; вища (або однакова) ефективність порівняно з найкращими світовими рішеннями.

Криптографічні алгоритми, які визначаються стандартами ДСТУ 7624:2014 і ДСТУ 7564:2014, є гнучкими, підтримують розмір блоку і довжину ключа від 128 до 512 біт.

Стандарт симетричного блокового шифрування «Калина» є результатом багаторічної плідної співпраці Державної служби спеціального зв'язку та захисту інформації України та провідних українських вчених.

Даний алгоритм шифрування враховує досвід і результати проведення міжнародних та відкритих національних конкурсів криптографічних алгоритмів.

Алгоритм ДСТУ 7624:2014 забезпечує досить високий рівень криптостікості порівняно з міжнародним стандартом AES (ISO/IEC 18033-3:2010), оскільки дає можливість застосовувати блок даних і ключ шифрування розміром аж до 512 біт.

Крім того, він має аналогічну або навіть більш високу швидкодію на сучасних і перспективних програмних та програмно-апаратних платформах.

На даний момент продовжуються роботи зі стандартизації вітчизняних криптографічних алгоритмів та протоколів.

При цьому не обмежується застосування гармонізованих стандартів у сфері захисту конфіденційної інформації.

Також зусилля зосереджено на використанні кращих практик застосування стандартів шифрування даних для захисту інформації в інформаційно-комунікаційних системах [10, 11].

Оскільки в стандартах симетричного блокового шифрування «Калина» та AES використовуються аналогічні криптографічні перетворення, на наш погляд, буде доцільним порівняти ці два алгоритми.

Основними відмінностями «Калина» від «Rijndael» (AES) є: збільшена кількість циклів шифрування (запас стійкості); використання додавання за модулем 264 і за модулем 2 для введення ключової інформації (захист від алгебричних атак, лінійного та диференціального криптоаналізів, інтерполяційної атаки тощо); використання чотирьох блоків нелінійного перетворення (S-блоків) замість одного (додатковий захист від алгебричних атак, поліпшення властивостей розсіювання алгоритму – покращені статистичні властивості, відповідно, більш високий рівень стійкості до диференціального та лінійного

криптоаналізів тощо); використання випадково сформованих чотирьох блоків, відібраних критеріями стійкості до диференціального, лінійного криптоаналізів, ступені нелінійності булевих функцій (на відміну від S-блоку Rijndael/Camellia та інших шифрів, що використовують звернення в полі та, відповідно, квадратичні залежності між входом і виходом, – захист від алгебричних атак); принципово нова схема створення підключів (захист від усіх відомих атак на схеми створення підключів); досить висока продуктивність; можливість відновлення сеансового ключа за окремим підключем (додатковий захист від атак, що виконують відновлення підключів).

Усі поліпшення спрямовані на збільшення стійкості та запобігання потенційним вразливостям відносно Rijndael, виявленим в останні роки [12].

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання магістерської роботи.

Розроблене програмне забезпечення методу оптимізації структури інформаційного ресурсу складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Конструктор; Аналізатор; Бібліотека шаблонів; Двідка.

– Навігаційні панелі.

– Блоку вибору режиму роботи конструктора.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

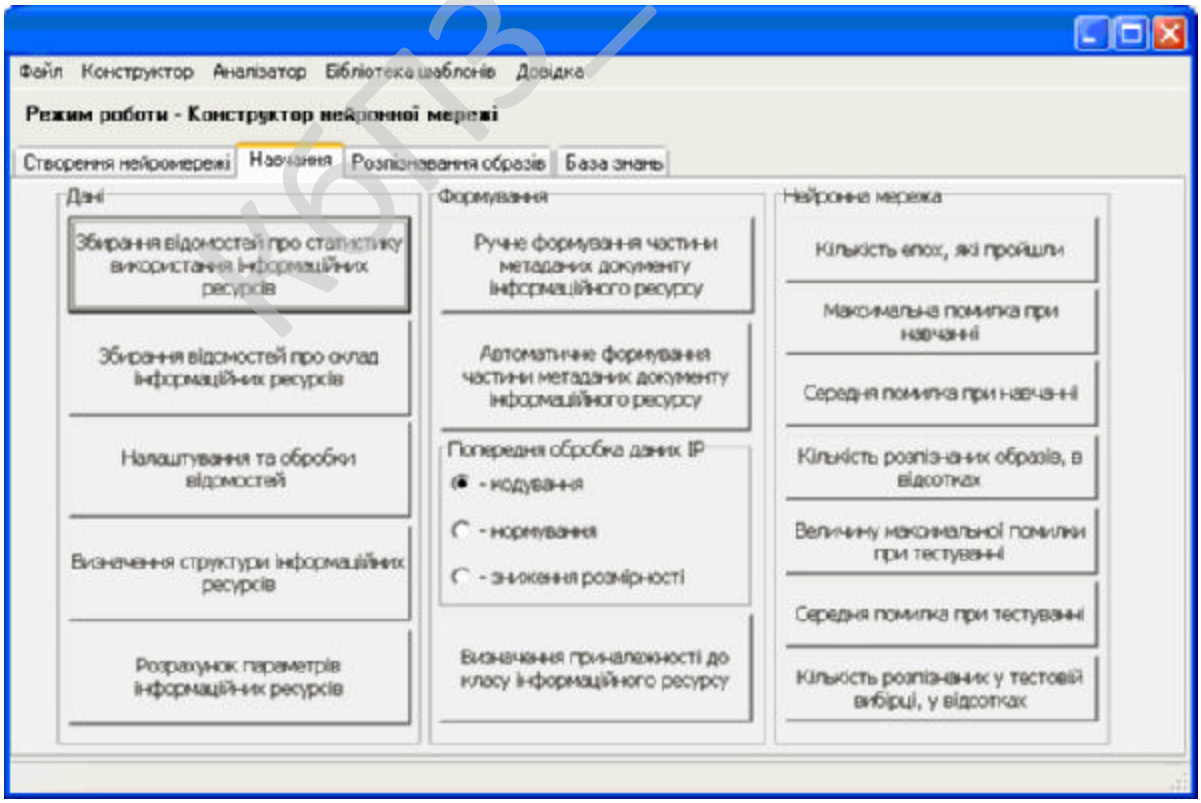

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку «Вікно розробника ПЗ», після чого на екрані з’явиться вікно показане на рисунку 5.2.

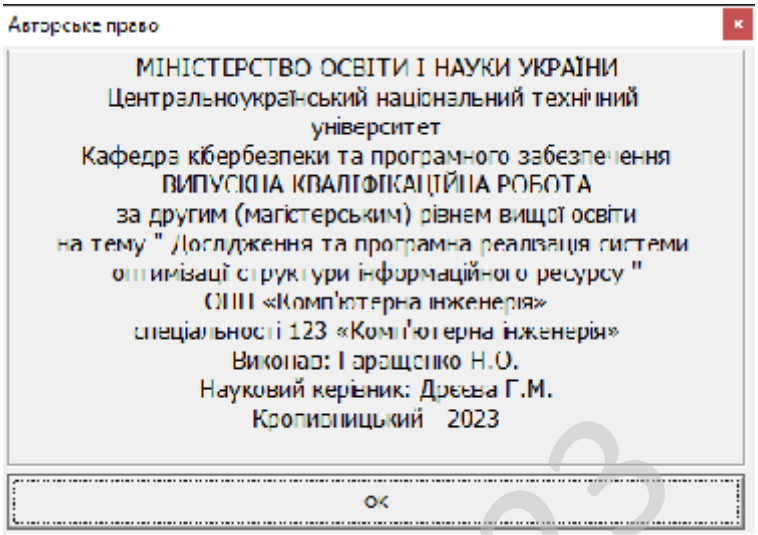

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми.

Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки» пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

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 (не повнофункціональна 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 (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи оптимізації структури інформаційного ресурсу.

Метою розробки є дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу.

Об'єктом дослідження є процес оптимізації структури інформаційного ресурсу.

Предметом дослідження є методи оптимізації структури інформаційного ресурсу.

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

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

– Удосконалено метод оптимізації структури інформаційного ресурсу.

– Розроблено вітчизняний продукт оптимізації структури інформаційного ресурсу, 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 проведено дослідження та виконана програмна реалізація системи оптимізації структури інформаційного ресурсу.

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 дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	24
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

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	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 таблиці 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 ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	24000
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	50
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.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

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

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

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

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

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

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:

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

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 58 = 98 \text{ люд/день.}$$

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	98	Ф 7.1-7.4
Впровадження	13	Д13
Всього	139	–

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: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{139 \cdot 1}{60 - 5} = 2,5 \text{ ставки.}$$

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	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
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:			$З_ч$	39,49

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{39 \cdot 3}{1{,}2} = 97{,}5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 97,5/(60 \cdot 8) = 0,2 \text{ ставки.}$$

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	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	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 до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть 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	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	19038	57114
Продакт-менеджер	0,25	18000	13500
Інженер-програміст	2,5	19000	142500
Інженер-електронщик	0,2	16500	9900
Інженер-системотехнік	0,25	16500	12375
Адміністратор мережі	0,5	16500	24750
Системний програміст	0,25	16500	12375
Дизайнер WEB	0,25	16500	12375
Інженер-верстальник	0,25	16500	12375
Бухгалтер-економіст	0,5	16500	24750
Всього за період розробки	$R_{сп} = 5,95$	-	$Ф_{роб} = 322014$

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{322014}{5,95 \cdot 60} = 902 \ грн.$$

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: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

S_y – 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:

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \qquad (7.10)$$

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

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

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 за прайсом фірми Brain за 24.10.23 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		7771
Процесор	AMD A8-9600 (AD9600AGM44AB) AM4, 4 ядра, 4 потоки, 3.1 GHz, 3.4 GHz, TDP – 65 Вт, 28nm	-
Системна плата	ASRock A520M-HVS сокет – AM4	-
Жорсткий диск	SSD 2.5" 256GB Mibrand (MI2.5SSD/CA256GBST) 256 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s	-
Оперативна пам’ять	DDR4 8GB 3200 MHz Dato (DT8G4DLDND32)	-
DVD-привод	-	-
Корпус	Vinga CS105B-450W Miditower, ATX, Micro – ATX, Mini – ITX, PSU – 450 Вт	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	Подару-нок
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

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	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

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

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	20	-
5. Транспортні засоби	143000	25	
6. Господарський інвентар	28000	20	-
Всього по групам 5, 6	176190	-	35238
7. Нематеріальні активи	24000	10	2400
Разом	$К_р$= 1733406		$А_р$= 170646

Примітка: вартість автомобіля Volkswagen Fox 2005 взята за даним сайту «Авто-РІА», джерело https://auto.ria.com/uk/auto_volkswagen_fox_33599812.html, складає 143000 грн.

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: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 902 \cdot 139/24 = 5225 \text{ грн.}$$

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, %.

$$З_д = 5225 \cdot 10 \cdot 0{,}01 = 523 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ 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, %.

$$С_{оц} = 0{,}01 \cdot 22(5225+523) = 1265 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 5225 \cdot 15 \cdot 0{,}01 = 784 \text{ грн.}$$

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 1,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 210 \cdot 1{,}5 = 315 \text{ грн.}$$

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 – 23,6 грн./шт., DVD-R box – 30 грн./шт.

$$З_{М2} = 30 \cdot 10 = 300 \text{ грн.}$$

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 Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

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

$$З_М = (315 + 300 + 1702)/24 = 97 \text{ грн.}$$

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

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

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 5225 \cdot 15 \cdot 0{,}01 = 784 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 24 прим.):

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

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 170646 \cdot 3/(24 \cdot 12) = 1777 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м\,. \tag{7.21}$$

$$С_п = 5225+523+1265+784+97+784+1777 = 10455 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 10455 = 5228 \text{ грн.}$$

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	2	3
1. Основна зарплата виконавців	$З_о$	5225
2. Додаткова зарплата виконавців	$З_д$	523
3. Відрахування на соціальні потреби	$С_{оц}$	1265
4. Загальногосподарські витрати	$Г_{осп}$	784
5. Витрати на матеріали	$З_М$	97
6. Освоєння нових операційних систем, мов програмування	$О_п$	784
7. Амортизація основних фондів	$А_м$	1777
8. Повна собівартість програмного забезпечення	$С_п$	10455
9. Плановий прибуток	$П_р$	5228
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	15683
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	ПДВ	3136,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	Ц	18819,6

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 7.10.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	18820
Всього капітальних витрат	–	18820

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.

$$З_о = Ч_е О_е М_е (1 + 0{,}01 Н_q)(1 + 0{,}01 Н_c), \qquad (7.23)$$

де $Ч_е$ – чисельність працівників, чол.;

$О_е$ – заробітна плата, грн./год;

$М_е$ – час, що витрачається на нарахування ЗП.

Після купівлі нового програмного забезпечення час на нарахування зменшився з 630 годин на рік до 393 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{о\,баз} = 1 \cdot 100 \cdot 630 \cdot 1{,}1 \cdot 1{,}22 = 84546\ грн,$$

до:

$$З_{о\,нов} = 1 \cdot 100 \cdot 393 \cdot 1{,}1 \cdot 1{,}22 = 52741\ грн.$$

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Заробітна плата(основна, додаткова, відрахування в соціальні фонди)	$З_р$	84546	52741
2. Витрати на електроенергію	$З_{ел}$	14688	13600
3. Витрати на амортизацію	$З_{ам}$	0	4705
Всього витрат за рік	І	99234	71046

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 ($Ц_{ел}$):

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

$$З_{ел\ баз} = 0{,}54 \cdot 8500 \cdot 3{,}2 = 14688 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}50 \cdot 8500 \cdot 3{,}2 = 13600 \text{ грн.}$$

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	25	–	18820	–	4705
Всього відрахувань	-	–	18820	–	4705

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.

$$Ев = (15683 - 10455) \cdot 24 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,25 \cdot 33190 + 0,2 \cdot 143000 + 0,1 \cdot 24000) \cdot 3/12 = 82396 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: K_p^* – балансова вартість основних фондів розробника.

$$T_в = \frac{1733406}{(15683 - 10455) \cdot 24 \cdot 12 / 3} = 3,45 \text{ років}.$$

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.

$$E_{сп} = (99234\text{-}71046)\text{-}0,25 \cdot 18820 = 23483 \text{ грн.}$$

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.	24
2. Повна собівартість розробленої програми	Грн.	10455
3. Ціна розробленої програми	Грн.	15683
4. Плановий прибуток від реалізації розробленої програми	Грн.	5228
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1733406
7. Загальний прибуток від реалізації програмної продукції	Грн.	125472
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	82396
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	3,45
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	18820
11.Величина економічного ефекту у користувача програмної продукції	Грн.	23483
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,7

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

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

$$T_{сп} = \frac{18820}{99234 - 71046} = 0,66 \text{ років}.$$

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

З метою запобігання пошкоджень, які можуть трапитися через ураження електричним струмом, коротким замиканням, загоряння тощо, було розроблено стандарт безпеки IEC 950. Для країн Європейскої співдружності загальним стандартом електробезпечності є Cemark.

Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути не менше площі перерізу фазового провідника.

Програмісти у процесі роботи отримують негативний вплив на органи зору, а також мають значну розумову напругу і нервово-емоційне навантаження. Руки (м’язи рук та суглоби пальців) при роботі з клавіатурою мають теж істотне навантаження. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій спеціалісти відносять високочастотні електромагнітні коливання роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: “Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин” ДсанПіН 3.3.2-007-98, та “Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями” НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.[6]

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Шкідливими факторами при роботі з персональним комп'ютером є неонізуюче випромінювання промислової частоти, збільшене нервово-емоційне навантаження на оператора, збільшення навантаження на органи зору та дрібні стереостатичні рухи кінцівок.

Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти ті інші захворювання.

Комп'ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп'ютером, повинні мати першу(або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп'ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров'я України "Про затвердження Порядку проведення медичних оглядів працівників певних категорій" від 21 травня 2007 року №246.[8]

8.3 Аналіз умов праці на робочому місці програміста

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 3,8 м×5,2 м×2,9 м. Одна з її більших стін має шість двостулкових вікон, розмірами 1,4 м×1,9 м, які виходять на південний захід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита леноліумом, всі стіни пофарбовані світло синього кольору до висоти 2,8 м, а далі розташована підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер, принтери). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1230 мм×845 мм. Висота столів 755 мм. Висота стільців від рівня підлоги становить 420 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 $м^2$ на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007-98 [2], вона повинна становити 20 $м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 [1] роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007-98 [2] рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних, прийому хворих в медпунктах	71	61	54	49	45	42	40	38	50

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А) при роботі

всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг. Даного місця програміста не мають регульованих параметрів. Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 8.2.

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765(740)	655(600)	450(440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

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

8.4 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *80·50·6 мм*., (згідно з ДСТУ 8769:2018 «Кутики сталеві гарячекатані нерівнополичні. Сортамент») довжиною *L=1,7 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проведемо 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=1,7 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:

$$T=t+L/2=0,6+1,7/2=1,45 \text{ м.}$$

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]:

$$Dв = 0{,}95 \cdot К = 0{,}95 \cdot 65 = 59{,}85 \text{ мм.} = 0{,}62 \text{ м.}$$

де $К = (80+50)/2 = 65$ мм. – середній розмір полиці метелевого кутка.

Відношення $A/L = 1{,}7/1{,}7 = 1$

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

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

$$= 0{,}366(54{,}5/1{,}7)[(\lg(2 \cdot 1{,}7/0{,}62) + (1/2)\lg((4 \cdot 1{,}45+1{,}7)/(4 \cdot 1{,}45-1{,}7))] =$$

$$= 21{,}9 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев = 0{,}8$ при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [12].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{3H} = 4$ Ом:

$$N = R_O / (Кев \, R_{3H}) = 21{,}9 / (0{,}8 \cdot 4) = 6{,}85 \approx 7 \text{ шт.}$$

Визначаємо длину з'єднуючої полоси:

$$L_П = 1{,}05 \cdot A \cdot N = 1{,}05 \cdot 1{,}7 \cdot 7 = 12{,}22 \approx 12 \text{ м.}$$

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

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

$$= 0{,}366(40 \cdot 5/12) \cdot \lg((2 \cdot 12^2)/(0{,}06 \cdot 0{,}6)) = 23{,}5 \text{ Ом.}$$

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

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

Загальний опір розтіканню електричного струму заземлювача [12]:

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

$$= (21,9 \cdot 23,5) / (21,9 \cdot 0,75 + 9 \cdot 23,5 \cdot 0,8) = 3,48 \text{ Ом.}$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [12].

Умова $R \leq R_{ЗН}$ виконується ($3,48 \leq 4$).

Так як при *7* вертикальних електродах *R* суттєво менше $R_{ЗН}$, зменшимо кількість вертикальних електродів *N* до *6* і виконаємо перерахунок. У результаті остаточно отримали: *R =3,98 Ом.* при кількості вертикальних електродів *N=6*.

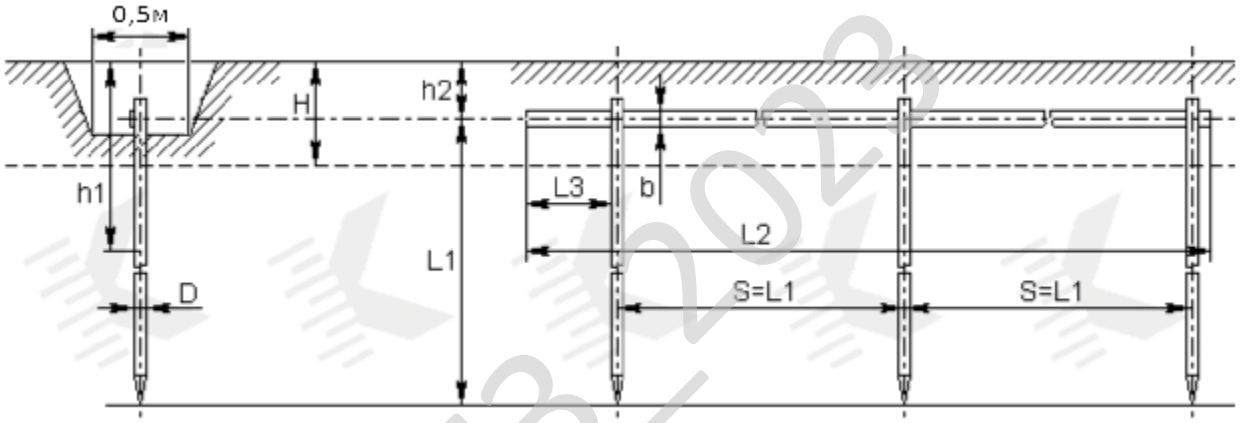

Рисунок 8.1 – Схема штучного заземлення.

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому. З цих міркувань було здійснено аналіз умов праці на робочому місці програміста, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. 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 оптимізації структури інформаційного ресурсу.

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 оптимізації структури інформаційного ресурсу.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем оптимізації структури інформаційного ресурсу.

– Досліджена система оптимізації структури інформаційного ресурсу.

– На основі отриманих результатів досліджень створена програмна реалізація системи оптимізації структури інформаційного ресурсу.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання оптимізації структури інформаційного ресурсу.

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 RAD Studio Delphi 10.4. 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 ДСТУ 7624:2014.

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 23483 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гаращенко Н.О. Дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. C. Ren, Z. Li, and H. Zhang, “Integrated multi-objective stochastic fuzzy programming and AHP method for agricultural water and land optimization allocation under multiple uncertainties,” *Journal of Cleaner Production*, vol. 210, pp. 12–24, 2019.

3. H. Xu, C. Tian, and Y. Li, “Emergency evacuation simulation and optimization for a complex rail transit station: a perspective of promoting transportation safety,” *Journal of Advanced Transportation*, vol. 2020, Article ID 8791503, 12 pages, 2020.

4. R. González-Bravo, J. Mahlknecht, and J. M. Ponce-Ortega, “Water, food and power grid optimization at macroscopic level involving multi- stakeholder approach,” *Energy Procedia*, vol. 153, pp. 347–352, 2018.

5. S. Hatfield, “Indigenous knowledge of the land and resources for optimization: redefining what management really means,” *Ecology*, vol. 99, no. 7, pp. 1701-1702, 2018.

6. Y. L. Xie, D. X. Xia, L. Ji, and G. H. Huang, “An inexact stochastic-fuzzy optimization model for agricultural water allocation and land resources utilization management under considering effective rainfall,” *Ecological Indicators*, vol. 92, pp. 301–311, 2018.

7. M. Zhao, L. Li, Y. Fang et al., “Optimization of intensive land use in blocks of Xi'an from the perspective of bicycle travel,” *Alexandria Engineering Journal*, vol. 60, no. 1, pp. 241–249, 2021.

8. N. Bavrovska and T. Shlikhta, “Land resources of the Zvenigorod district of Cherkasy region: assessment of the state and optimization,” *Zemleustrìj, Kadastr ì Monìtorìng Zemel*, vol. 4, no. 4, pp. 53–60, 2018.

9. M. Anis, A. Idrus, H. Amijaya, and S. Subagyo, “Utilizing coal remaining resources and post-mining land use planning based on GIS-based optimization method : study case at PT Adaro coal mine in South Kalimantan,” *Journal of Geoscience, Engineering, Environment, and Technology*, vol. 2, no. 2, pp. 141–148, 2017.

10. H. Qin, C. B. Andrews, F. Tian et al., “Groundwater-pumping optimization for land-subsidence control in Beijing plain, China,” *Hydrogeology Journal*, vol. 26, no. 4, pp. 1061–1081, 2018.

11. S. Harasimowicz, J. Janus, S. Bacior, and J. Gniadek, “Shape and size of parcels and transport costs as a mixed integer programming problem in optimization of land consolidation,” *Computers and Electronics in Agriculture*, vol. 140, pp. 113–122, 2017.

12. Y. Nie, S. Avraamidou, X. Xiao, E. N. Pistikopoulos, and J. Li, “Two-stage land use optimization for a Food-Energy-Water Nexus system: a case study in Texas Edwards Region,” *Computer Aided Chemical Engineering*, vol. 47, pp. 205–210, 2019.

13. Q. Wang, R. Liu, C. Men, and L. Guo, “Application of genetic algorithm to land use optimization for non-point source pollution control based on CLUE-S and SWAT,” *Journal of Hydrology*, vol. 560, pp. 86–96, 2018.

14. Z. Li, X. Deng, A. Arowolo, Q. Jiang, and H. Yan, “Adapting water scarcity for river basin: optimization of land uses,” vol. 1, pp. 19–50, 2019.

15. N. Xiao and A. T. Murray, “Spatial optimization for land acquisition problems: a review of models, solution methods, and GIS support,” *Transactions in GIS*, vol. 23, no. 4, pp. 645–671, 2019.

16. A. Singh, C. Chhablani, and L. Goel, “Moth flame optimization for land cover feature extraction in remote sensing images,” in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, Delhi, India, 2017.

17. E. V. Panin, I. V. Yaurova, and A. A. Kharitonov, “Improvement of information and technological management of land resources and regulation of land and property relations,” *Vestnik Of Voronezh State Agrarian University*, vol. 1, no. 60, pp. 226–233, 2019.

18. K. S. Harmanny and Z. Malek, “Adaptations in irrigated agriculture in the Mediterranean region: an overview and spatial analysis of implemented strategies,” *Regional Environmental Change*, vol. 19, no. 5, pp. 1401–1416, 2019.

19. Y. A. Maglinets, K. V. Raevich, and G. M. Tsibulskii, “Architecture of the information system of evaluating of the land resources based on processing of spatial data,” *Journal of Siberian Federal University: Engineering & Technologies*, vol. 11, no. 1, pp. 52–60, 2018.

20. N. V. Nechyporuk, “Information support for the land registration: directions for upgrading of statistical reporting,” *Sustainability*, vol. 80, no. 1, pp. 24–29, 2018.

21. M. R. España-Villanueva and L. M. Valenzuela-Montes, “The role of information in plans for progressing in IWLRM,” *Land Use Policy*, vol. 67, pp. 327–339, 2017.

22. P. Danoedoro, “Multidimensional land-use information for local planning and land resources assessment in Indonesia: classification scheme for information extraction from high-spatial resolution imagery,” *The Indonesian Journal of Geography*, vol. 51, no. 2, pp. 131–146, 2019.

23. J. H. Wu, W. Wei, L. Zhang et al., “Risk assessment of hypertension in steel workers based on LVQ and fisher-SVM deep excavation,” *Ieee Access*, vol. 7, pp. 23109–23119, 2019.

24. F. Orujov, R. Maskeliūnas, R. Damaševičius, W. Wei, and Y. Li, “Smartphone based intelligent indoor positioning using fuzzy logic,” *Future Generation Computer Systems*, vol. 89, pp. 335–348, 2018.

25. W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, “Accurate and fast URL phishing detector: a convolutional neural network approach,” *Computer Networks*, vol. 178, article 107275, 2020.

26. W. Wang, N. Kumar, J. Chen et al., “Realizing the potential of the Internet of Things for smart tourism with 5G and AI,” *IEEE Network*, vol. 34, no. 6, pp. 295–301, 2020.

27. Aggarwal C.C. Neural Networks and Deep Learning: A Textbook 1st ed. 2018 Edition. – Springer, 2018. – 520 p

28. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 c.

29. Brink H., Richards J., Fetherolf M. Real-World Machine Learning. – Manning, 2016. – 474 p.

30. 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.

31. Fenner M. Machine Learning with Python for Everyone (Addison-Wesley Data & Analytics Series) 1st Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 586 p.

32. Foreman J.W. Data Smart: Using Data Science to Transform Information into Insight 1st Edition. – Wiley, 2013. – 432 p.

33. Hurbans R. Grokking Artificial Intelligence Algorithms. – Manning, 2020. – 631 p.

34. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 p.

35. Kotu V., Deshpande B. Data Science: Concepts and Practice. – Elsevier Science, 2018. – 953 p.

36. Knowledge Base A Complete Guide – 2021 Edition // The Art of Service – Knowledge Base Publishing, 2020. – 306 p.

37. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

38. Mattmann C. Machine Learning with TensorFlow, Second Edition. – Manning, 2020. – 1124 p.

39. Mueller J.P., Massaron L. Machine Learning For Dummies. – Wiley, 2016. – 714 p.

40. Teofili T. Deep Learning for Search. – Manning, 2019. – 695 p.

41. Rungta K. TensorFlow in 1 Day: Make your own Neural Network. – Publishdrive, 2019. – 587 p.

42. Weidman S. Deep Learning from Scratch: Building with Python from First Principles. – O'Reilly. 2019. – 252 p.

43. Rajasekaran S., Vijayalakshmi Pai G.A. Neural networks, fuzzy logic, and genetic algorithms: synthesis and applications (with cd-rom) Kindle Edition. – PHI, 2013. – 628 p.

44. 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.

45. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022, pp. 1-12.

46. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

47. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

48. Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., Smirnova, T. «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and Systems. Volume 152, 2021, Pages 85-98.

49. 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.

50. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». CEUR Workshop Proceedings Volume 3101, 2021, Pages 192-207.

51. 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.

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

53. 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.

54. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

55. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

56. 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.

57. 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.

58. 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.

59. 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.

60. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». CEUR Workshop Proceedings, Vol 2588, P. 215-227, 2019.

61. 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.

62. 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.

63. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

64. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

65. 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.

66. Smirnov, S., Bulekbaeva, G., Kikvidze, O.G., Lakhno, V., Brzhanov, R., Tabylov, A. «Computer simulation in the MathCAD package of plastic deformation of the deposited layer on the flat surface of the part». Journal of Theoretical and Applied Information Technology Volume 97, Issue 20, 2019, Pages 2467-2484. (Scopus).

67. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

68. 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.

69. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95

70. 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.

71. 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.

72. 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.

73. Смірнов, С.А., Смірнова, Т.В., Минайленко, Р.М., Доренський, О.П., Сисоєнко С.В. «Хмарна автоматизована система інтелектуальної підтримки прийняття рішень для технологічних процесів». Вісник Черкаського державного технологічного університету. Технічні науки. №4, 2020, С. 84-92.

74. 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.

75. 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).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи оптимізації структури інформаційного ресурсу.

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. № 36-13 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 оптимізації структури інформаційного ресурсу.

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 оптимізації структури інформаційного ресурсу;

– 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 RAD Studio Delphi 10.4.

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 програміста.

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 – 104 аркуші.

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 14.12.2023 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

_____ Дрєєва Г.М.

Дослідження та програмна реалізація системи оптимізації структури інформаційного ресурсу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 63

Літера: РП

Кропивницький – 2023 року

Neural_network_Editor.pas – розробка нейронних мереж

```
unit Neural_network_Editor;

interface

uses
  Classes,
  DesignIntf, DesignEditors,
  Neural_network_Comp, Neural_network_EditorForm,
  SysUtils, Dialogs, Controls, Neural_network_EditorFieldsForm;

type
  TNeuronsInLayerFieldsProperty = class(TStringProperty)
  public
    function GetAttributes : TPropertyAttributes; override;
    function GetValue : string; override;
    procedure Edit; override;
  end;

  TFileNameFieldsProperty = class(TStringProperty)
  public
    function GetAttributes : TPropertyAttributes; override;
    procedure Edit; override;
  end;

  TOptionsFieldsProperty = class(TStringProperty)
  public
    function GetAttributes : TPropertyAttributes; override;
    function GetValue : string; override;
    procedure Edit; override;
  end;

procedure Register;

implementation

function TOptionsFieldsProperty.GetAttributes;
begin
  Result := [paDialog];
end;

procedure TOptionsFieldsProperty.Edit;
var
  i: integer;
  xNeuralNetExtended: TNeuralNetExtended;
  frmNeuroFields: TfrmNeuroFields;
  xCurrent: integer;
begin
  frmNeuroFields := TfrmNeuroFields.Create(nil);
  try
    with frmNeuroFields do
    begin
      xNeuralNetExtended := (GetComponent(0) as TNeuralNetExtended);
      for i := 0 to xNeuralNetExtended.AvailableFieldsCount - 1 do
        ltbFieldName.Items.Add(xNeuralNetExtended.Fields[i].Name);
      xCurrent := 0;
      rdgFieldType.ItemIndex := xNeuralNetExtended.Fields[xCurrent].Kind;
      rdgNormType.ItemIndex := xNeuralNetExtended.Fields[xCurrent].NormType;
      edtAlpha.Text := FloatToStr(xNeuralNetExtended.Fields[xCurrent].Alpha);
      sttMin.Caption :=
FloatToStr(xNeuralNetExtended.Fields[xCurrent].ValueMin);
      sttMax.Caption :=
FloatToStr(xNeuralNetExtended.Fields[xCurrent].ValueMax);
      NeuralNetExtended := xNeuralNetExtended;
      ShowModal;
    end;
```

```
  except
    frmNeuroFields.Free;
  end;
end;

function TOptionsFieldsProperty.GetValue;
var
  i: integer;
begin
  // внести зміни
  Result := '[';
  with (GetComponent(0) as TNeuralNetExtended) do
    if AvailableFieldsCount > 1 then
    begin
      for i := 0 to AvailableFieldsCount - 1 do
        Result := Result + Fields[i].Name + ',';
      Result[Length(Result)] := ']'
    end
    else
      Result[Length(Result) + 1] := ']';
end;

function TNeuronsInLayerFieldsProperty.GetAttributes;
begin
  Result := [paDialog];
end;

function TNeuronsInLayerFieldsProperty.GetValue;
var
  i: integer;
begin
  // внести зміни
  Result := '[';
  with (GetComponent(0) as TNeuralNetBP) do
    if LayerCount > 0 then
    begin
      for i := 0 to LayerCount - 1 do
        Result := Result + IntToStr(LayersBP[i].NeuronCount) + ',';
      Result[Length(Result)] := ']';
    end
    else
      Result[Length(Result) + 1] := ']';
end;

procedure TNeuronsInLayerFieldsProperty.Edit;
var
  i: integer;
  xPreviousCount: integer;
  xNeuralNetBP: TNeuralNetBP;
  frmNeuronsInLayer: TfrmNeuronsInLayer;
  xChangesMade: boolean;
begin
  xChangesMade := False;
  frmNeuronsInLayer := TfrmNeuronsInLayer.Create(nil);
  try
    with frmNeuronsInLayer do
    begin
      xNeuralNetBP := (GetComponent(0) as TNeuralNetBP);
      speLayers.Value := xNeuralNetBP.LayerCount;
      stgNeuronsInLayer.RowCount := xNeuralNetBP.LayerCount + 1;
      for i := 0 to xNeuralNetBP.LayerCount - 1 do
      begin
        stgNeuronsInLayer.Cells[0, i + 1] := IntToStr(i);
        stgNeuronsInLayer.Cells[1, i + 1] :=
IntToStr(xNeuralNetBP.LayersBP[i].NeuronCount);
      end;
      if ShowModal = mrOk then
      begin
        xNeuralNetBP.ResetLayers;
```

```
        if speLayers.Value = 0 then
        begin
          xNeuralNetBP.ResetLayers;
          Exit;
        end;
        if xNeuralNetBP.LayerCount <> speLayers.Value then
          xChangesMade := True
        else
          for i := 0 to xNeuralNetBP.LayerCount - 1 do
            if xNeuralNetBP.LayersBP[i].NeuronCount <>
StrToInt(stgNeuronsInLayer.Cells[1, i + 1]) then
            begin
              xChangesMade := True;
              Break;
            end;
        if xChangesMade then
        begin
          if xNeuralNetBP.LayerCount = speLayers.Value then
            for i := 0 to speLayers.Value - 1 do
              xNeuralNetBP.NeuronsInLayer[i] := stgNeuronsInLayer.Cells[1, i +
1]
          else
            if xNeuralNetBP.LayerCount < speLayers.Value then
            begin
              for i := 0 to xNeuralNetBP.LayerCount - 1 do
                xNeuralNetBP.NeuronsInLayer[i] := stgNeuronsInLayer.Cells[1, i +
1];
              for i := xNeuralNetBP.LayerCount to speLayers.Value - 1 do
                xNeuralNetBP.AddLayer(StrToInt(stgNeuronsInLayer.Cells[1, i +
1]));
            end
            else
            begin
              if speLayers.Value > 0 then
                for i := 0 to speLayers.Value - 1 do
                  xNeuralNetBP.NeuronsInLayer[i] := stgNeuronsInLayer.Cells[1, i
+ 1];
              xPreviousCount := xNeuralNetBP.LayerCount -  speLayers.Value;
              for i := 1 to xPreviousCount do
                xNeuralNetBP.DeleteLayer(xNeuralNetBP.LayerCount - 1);
            end;
          if not xNeuralNetBP.AutoInit then
            xNeuralNetBP.AutoInit := True;
        end;
      end;
    end;
  except
    frmNeuronsInLayer.Free;
  end;
end;

function TFileNameFieldsProperty.GetAttributes;
begin
  Result := [paDialog];
end;

procedure TFileNameFieldsProperty.Edit;
var
  xOpenDialog: TOpenDialog;
begin
  xOpenDialog := TOpenDialog.Create(nil);
  if UpperCase(GetName) = 'FILENAME' then
    xOpenDialog.Filter := 'Нейронна мережа(*.nnw)|*.nnw|всі файли (*.*)|*.*';
  if UpperCase(GetName) = 'SOURCEFILENAME' then
    xOpenDialog.Filter := 'Текстові файли (*.txt)|*.txt|всі файли (*.*)|*.*';

  if xOpenDialog.Execute then
  begin
    if UpperCase(GetName) = 'FILENAME' then
```

```
      (GetComponent(0) as TNeuralNetExtended).FileName := xOpenDialog.FileName;
    if UpperCase(GetName) = 'SOURCEFILENAME' then
      (GetComponent(0) as TNeuralNetExtended).SourceFileName :=
xOpenDialog.FileName;
  end;
  xOpenDialog.Free;
end;

procedure Register;
begin
  RegisterPropertyEditor(TypeInfo(TStrings), TNeuralNetBP, 'NeuronsInLayer',
TNeuronsInLayerFieldsProperty);
  RegisterPropertyEditor(TypeInfo(TFileName), TNeuralNetExtended, '',
TFileNameFieldsProperty);
  RegisterPropertyEditor(TypeInfo(string), TNeuralNetExtended, 'Options',
TOptionsFieldsProperty);
end;

end.
```

Neural_network_EditorForm.pas – редактор нейронних мереж

```
unit Neural_network_EditorForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, StdCtrls, ExtCtrls, Neural_network_Comp, Spin, Neural_network_Types;

type
  TfrmNeuronsInLayer = class(TForm)
    Bevel1: TBevel;
    btnOk: TButton;
    btnCancel: TButton;
    stgNeuronsInLayer: TStringGrid;
    Label1: TLabel;
    speLayers: TSpinEdit;
    procedure stgNeuronsInLayerGetEditMask(Sender: TObject; ACol,
      ARow: Integer; var Value: String);
    procedure stgNeuronsInLayerSetEditText(Sender: TObject; ACol,
      ARow: Integer; const Value: String);
    procedure speLayersChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmNeuronsInLayer: TfrmNeuronsInLayer;

implementation

{$R *.DFM}

procedure TfrmNeuronsInLayer.stgNeuronsInLayerGetEditMask(Sender: TObject;
  ACol, ARow: Integer; var Value: String);
begin
  Value := '0000';
end;

procedure TfrmNeuronsInLayer.stgNeuronsInLayerSetEditText(Sender: TObject;
  ACol, ARow: Integer; const Value: String);
begin
{  with stgNeuronsInLayer do
    try
      Cells[ACol, ARow] := IntToStr(StrToInt(trim(Value)));
    except
      Cells[ACol, ARow] := IntToStr(DefaultNeuronCount);
    end;}
end;

procedure TfrmNeuronsInLayer.speLayersChange(Sender: TObject);
var
  i: integer;
begin
  stgNeuronsInLayer.RowCount := speLayers.Value + 1;
  for i := 1 to stgNeuronsInLayer.RowCount do
    if trim(stgNeuronsInLayer.Cells[1, i]) = '' then
    begin
      stgNeuronsInLayer.Cells[0, i] := IntToStr(i);
      stgNeuronsInLayer.Cells[1, i] := IntToStr(DefaultNeuronCount);
    end;
end;

procedure TfrmNeuronsInLayer.FormCreate(Sender: TObject);
```

```
begin
  stgNeuronsInLayer.Cells[0,0] := '# шару';
  stgNeuronsInLayer.Cells[1,0] := 'нейронів';
end;

end.
```

Neural_network_EditorFieldsForm.pas – редактор властивостей полів нейроної мережі

```
unit Neural_network_EditorFieldsForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Neural_network_Comp;

type
  TfrmNeuroFields = class(TForm)
    ltbFieldName: TListBox;
    rdgFieldType: TRadioGroup;
    rdgNormType: TRadioGroup;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    sttMin: TStaticText;
    sttMax: TStaticText;
    edtAlpha: TEdit;
    btnOk: TButton;
    btnCancel: TButton;
    Bevel2: TBevel;
    Label4: TLabel;
    procedure ltbFieldNameClick(Sender: TObject);
    procedure rdgFieldTypeClick(Sender: TObject);
    procedure rdgNormTypeClick(Sender: TObject);
    procedure edtAlphaChange(Sender: TObject);
  private
    { Private declarations }
  public
    NeuralNetExtended: TNeuralNetExtended;
    { Public declarations }
  end;

var
  frmNeuroFields: TfrmNeuroFields;

implementation

{$R *.DFM}

procedure TfrmNeuroFields.ltbFieldNameClick(Sender: TObject);
begin
  rdgFieldType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind;
  rdgNormType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType;
  edtAlpha.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha);
  sttMin.Caption :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMin);
  sttMax.Caption :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMax);
end;

procedure TfrmNeuroFields.rdgFieldTypeClick(Sender: TObject);
var
  i: integer;
begin
  if ltbFieldName.SelCount = 1 then
    NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind :=
rdgFieldType.ItemIndex
```

```
  else
    if ltbFieldName.SelCount > 1 then
    for i := 0 to ltbFieldName.Items.Count - 1 do
      if ltbFieldName.Selected[i] then
        NeuralNetExtended.Fields[i].Kind := rdgFieldType.ItemIndex;
end;

procedure TfrmNeuroFields.rdgNormTypeClick(Sender: TObject);
var
  i: integer;
begin
  if ltbFieldName.SelCount = 1 then
    NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType :=
rdgNormType.ItemIndex
  else
    if ltbFieldName.SelCount > 1 then
    for i := 0 to ltbFieldName.Items.Count - 1 do
      if ltbFieldName.Selected[i] then
        NeuralNetExtended.Fields[i].NormType := rdgNormType.ItemIndex;
end;

procedure TfrmNeuroFields.edtAlphaChange(Sender: TObject);
begin
  NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha :=
StrToFloat(edtAlpha.Text);
end;

end.
```

Neural_network_Types.pas – ініціалізація базових констант

```
unit Neural_network_Types;

interface

const

  DefaultAlpha = 1;               // Параметр активаційної функції
  DefaultEpochCount = 10000;      // Кількість етапів для навчання
  DefaultErrorValue = 0.05;       // Помилка за замовчуванням для всіх видів
  DefaultHopfLayerCount = 2;      // Кількість шарів у мережі Хопфилда
  DefaultLayerCount = 0;          // Мінімальна кількість шарів у мережі back-
propagation
  DefaultMaxIterCount = 10;       // Максимальна кількість ітерацій в алгоритмі
Хопфилда
  DefaultMomentum = 0.9;          // Імпульс - момент
  DefaultNeuronCount = 0;         // Кількість нейронів у прихованому шарі за
замовчуванням
  DefaultPatternCount = 0;        // Кількість прикладів
  DefaultTeachRate = 0.1;         // Швидкість навчання
  DefaultTeachIdentCount = 100;   // Необхідний відсоток розпізнаних прикладів з
навчальної множини
  DefaultTestIdentCount = 100;    // Необхідний відсоток розпізнаних прикладів з
тестової множини
  DefaultUseForTeach = 100;       // Відсоток прикладів використовуваних як
навчальна множина
  DefaultDeltaBarAcceleratingConst = 0.095;
  DefaultDeltaBarDecFactor = 0.85;
  DefaultRPropInitValue = 0.1;
  DefaultRPropMaxStepSize = 50;
  DefaultRPropMinStepSize = 1 E-6;
  DefaultRPropDecFactor = 0.5;
  DefaultRPropIncFactor = 1.2;
  DefaultSuperSABDecFactor = 0.5;
  DefaultSuperSABIncFactor = 1.05;

  SensorLayer = 0;                // Сенсорний шар
  BiasNeuron = 1;                 // Зсув у мережі back-propagation

  Separators = ['.', ','];
  Letters    = ['a'..'z'];
  Capitals   = ['A'..'Z'];
  DigitChars = ['0'..'9'];
  SpaceChar = #9;

resourcestring

  SFieldNorm = 'Не існує типу нормалізації %d';
  SFieldKind = 'Не існує типу поля %d';
  SFieldIndexRange = 'Неправильно зазначений номер поля %d';
  SInFieldCount = 'Неправильно встановлена кількість вхідних полів';
  SInNeuronCount = 'Неправильно встановлена кількість вхідних нейронів';
  SInVectorCount = 'Неправильно встановлена розмірність вхідного вектора';
  SLayerRangeIndex = 'Неправильно зазначений номер шару %d';
  SNeuronRangeIndex = 'Неправильно зазначений номер нейрона %d';
  SNeuronCount = 'Неправильно зазначена кількість нейронів';
  SOutFieldCount = 'Неправильно встановлена кількість вихідних полів';
  SOutNeuronCount = 'Неправильно встановлена кількість вихідних нейронів';
  SOutVectorCount = 'Неправильно встановлена розмірність вихідного вектора';
  SPatternRangeIndex = 'Вихід за межі масиву прикладів %d';
  SStreamCannotRead = 'Помилка читання з потоку';
  SWeightRangeIndex = 'Неправильно зазначений номер ваги %d';
  SWrongFileName = 'Неправильно зазначене ім'я файлу %s';
  SCannotBeNumber = 'Помилка, вираз %s неможливо привести до числового типу';
  SBPStopCondition = 'Не задана умова зупинки процесу навчання';
```

```
type

  TVectorInt = array of integer;
  TVectorFloat = array of double;
  TVectorString = array of string;
  TMatrixInt = array of array of integer;
  TMatrixFloat = array of array of double;
  TNormalize = (nrmLinear, nrmSigmoid, nrmAuto,  nrmNone,
                nrmLinearOut, nrmAutoOut);
  TNeuroFieldType = (fdInput, fdOutput, fdNone);

implementation


end.
```

Neural_network_Comp.pas – формування бібліотеки шаблонів та дослідження нейронних мереж

```
unit Neural_network_Comp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, PumpData, Neural_network_Types, IniFiles, Math;

type

  // Класи виключень
  EInOutDimensionError = class(Exception);
  ENeuronCountError  = class(Exception);
  ENeuronNotEqualFieldError  = class(Exception);
  EBPStopCondition  = class(Exception);

  // Процедурні типи
  TActivation = function (Value: double): double of object;

  // Упереджуюче оголошення класів
  TNeuron = class;
  TLayer = class;

  // Базовий клас нейрона

  TNeuron = class(TObject)
  private
    FOutput: double;
    // Вектор ваг
    FWeights: TVectorFloat;
    // Покажчик на шар, у якому перебуває нейрон
    Layer: TLayer;
    function GetWeights(Index: integer): double;
    procedure SetWeights(Index: integer; Value: double);
    procedure SetWeightCount(const Value: integer);
  public
    constructor Create(ALayer: TLayer); virtual;
    destructor Destroy; override;
    // Ініціалізація ваг
    procedure InitWeights;  virtual;
    // Зважена сума
    procedure ComputeOut(const AInputs: TVectorFloat); virtual;
    property Output: double read FOutput write FOutput;
    property WeightCount: integer  write SetWeightCount;
    property Weights[Index: integer]: double read GetWeights write SetWeights;
  end;

  // Клас нейрона для мережі Хопфілда

  TNeuronHopf = class(TNeuron)
  public
    procedure ComputeOut(const AInputs: TVectorFloat); override;
  end;

  // Клас нейрона для мережі

  TNeuronBP = class(TNeuron)
  private
    // Локальна помилка
    FDelta: double;
    // Значення швидкості навчання на попередньому етапі
    FLearningRate: TVectorFloat;
    // Значення частинної похідної на попередньому етапі
    FPrevDerivative: TVectorFloat;
```

```
    // Значення корекції ваги на попередньому етапі
    FPrevUpdate: TVectorFloat;
    // Функція активації
    FOnActivation: TActivation;
    // Похідна функції активації
    FOnActivation: TActivation;
    function GetPrevUpdate(Index: integer): double;
    function GetPrevDerivative(Index: integer): double;
    function GetLearningRate(Index: integer): double;
    function GetPrevUpdateCount: integer;
    procedure SetPrevDerivative(Index: integer; const Value: double);
    procedure SetPrevDerivativeCount(const Value: integer);
    procedure SetDelta(Value: double);
    procedure SetPrevUpdate(Index: integer; Value: double);
    procedure SetPrevUpdateCount(const Value: integer);
    procedure SetLearningRate(Index: integer; const Value: double);
    procedure SetLearningRateCount(const Value: integer);
  public
    destructor Destroy; override;
    procedure ComputeOut(const AInputs: TVectorFloat); override;
    property Delta: double read FDelta write SetDelta;
    property LearningRate[Index: integer]: double read GetLearningRate write
SetLearningRate; //
    property LearningRateCount: integer write SetLearningRateCount;
    property PrevDerivativeCount: integer write SetPrevDerivativeCount;
    property PrevDerivative[Index: integer]: double read GetPrevDerivative write
SetPrevDerivative; //
    property PrevUpdateCount: integer read GetPrevUpdateCount write
SetPrevUpdateCount;
    property PrevUpdate[Index: integer]: double read GetPrevUpdate write
SetPrevUpdate;
    property OnActivation: TActivation read FOnActivation write FOnActivation;
    property OnActivation: TActivation read FOnActivation write FOnActivation;
  end;

  // Базовий клас шару
  TLayer = class(TPersistent)
  private
    FNumber: integer;
    // Розмірність NeuronCount
    FNeurons: array of TNeuron;
    function GetNeurons(Index: integer): TNeuron;
    function GetNeuronCount: integer;
    procedure SetNeurons(Index: integer; Value: TNeuron);
    procedure SetNeuronCount(Value: integer);
  public
    constructor Create(ALayerNumber: integer; ANeuronCount: integer); virtual;
    destructor Destroy; override;
    procedure Assign(Source: TPersistent); override;
    property Neurons[Index: integer]: TNeuron read GetNeurons write SetNeurons;
    property NeuronCount: integer read GetNeuronCount write SetNeuronCount;
  end;

  // Клас шару для мережі Хопфілда
  TLayerHopf = class(TLayer)
  public
    constructor Create(ALayerNumber: integer; ANeuronCount: integer); override;
  end;

  // Клас шару для мережі
  TLayerBP = class(TLayer)
  private
    function GetNeuronsBP(Index: integer): TNeuronBP;
    procedure SetNeuronsBP(Index: integer; Value: TNeuronBP);
  public
    constructor Create(ALayerNumber: integer; ANeuronCount: integer); override;
    destructor Destroy; override;
    procedure Assign(Source: TPersistent); override;
```

```
    property NeuronsBP[Index: integer]: TNeuronBP read GetNeuronsBP write
SetNeuronsBP;
  end;

  // Базовий клас мережі
  TNeuralNet = class(TComponent)
  private
    // Масив шарів
    FLayers: array of TLayer;
    // Число вибірок
    FPatternCount: integer;
    // Розмірність FPatternCount, InputNeuronCount
    FPatternsInput: TMatrixFloat;
    // Розмірність FPatternCount, OutputNeuronCount
    FPatternsOutput: TMatrixFloat;
    function GetLayers(Index: integer): TLayer;
    function GetOutputNeuronCount: integer;
    function GetPatternsOutput(PatternIndex: integer; OutputIndex: integer):
double;
    function GetPatternsInput(PatternIndex: integer; InputIndex: integer):
double;
    procedure SetLayers(Index: integer; Value: TLayer);
    procedure SetPatternsInput(PatternIndex: integer; InputIndex: integer;
Value: double);
    procedure SetPatternsOutput(PatternIndex: integer; InputIndex: integer;
Value: double);
  protected
    function GetLayerCount: integer; virtual;
    function GetInputNeuronCount: integer; virtual;
    procedure Clear; virtual;
    procedure ResizeInputDim; virtual;
    procedure ResizeOutputDim; virtual;
    procedure SetPatternCount(const Value: integer); virtual;
    procedure SetLayerCount(Value: integer); virtual;
    property PatternCount: integer read FPatternCount write SetPatternCount;
  public
    destructor Destroy; override;
    procedure AddLayer(ANeurons: integer); virtual; abstract;
    procedure AddPattern(const AInputs: TVectorFloat; const AOutputs:
TVectorFloat); overload; virtual;
    procedure DeleteLayer(Index: integer); virtual; abstract;
    procedure DeletePattern(Index: integer); virtual;
    procedure Init(const ANeuronsInLayer: TVectorInt); overload; virtual;
    property InputNeuronCount: integer read GetInputNeuronCount;
    property LayerCount: integer read GetLayerCount write SetLayerCount;
    property Layers[Index: integer]: TLayer read GetLayers write SetLayers;
    property OutputNeuronCount: integer read GetOutputNeuronCount;
    property PatternsInput[PatternIndex: integer; InputIndex: integer]: double
read GetPatternsInput write SetPatternsInput;
    property PatternsOutput[PatternIndex: integer; InputIndex: integer]: double
read GetPatternsOutput write SetPatternsOutput;
    procedure ResetPatterns; virtual;
  end;

  // Клас мережі Хопфілда
  TNeuralNetHopf = class(TNeuralNet)
  private
    FAutoInit: boolean;
    FInputNeuronCount: integer;
    FMaxIterCount: integer;
    FPatternCount: integer;
    FPatterns: TMatrixInt;
    FOnAfterInit: TNotifyEvent;
    FOnBeforeInit: TNotifyEvent;
    FOnPatternRecognized: TNotifyEvent;
    function GetInput(Index: integer): double;
    function GetPatterns(InputIndex: integer; PatternIndex: integer): integer;
    function Stabled: boolean;
    procedure SetInput(Index: integer; Value: double);
```

```
    procedure SetPatterns(InputIndex: integer; PatternIndex: integer; Value:
integer);
  protected
    function GetInputNeuronCount: integer; override;
    function GetLayerCount: integer; override;
    procedure SetInputNeuronCount(Value: integer);
    procedure SetPatternCount(const Value: integer); override;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure AddPattern(const ANewPattern: TVectorInt); reintroduce; overload;
    procedure Calc; virtual;
    procedure DeletePattern(Index: integer); override;
    procedure Init; reintroduce; overload;
    procedure InitWeights; virtual;
    procedure ResetPatterns; override;
    procedure ResizePatternsDim; virtual;
    property Input[Index: integer]: double read GetInput write SetInput;
    property LayerCount: integer read GetLayerCount write SetLayerCount;
    property Patterns[InputIndex: integer; PatternIndex: integer]: integer read
GetPatterns write SetPatterns;
  published
    property AutoInit: boolean read FAutoInit write FAutoInit;
    property InputNeuronCount: integer read GetInputNeuronCount write
SetInputNeuronCount;
    property MaxIterCount: integer read FMaxIterCount write FMaxIterCount;
    property OnAfterInit: TNotifyEvent read FOnAfterInit write FOnAfterInit;
    property OnBeforeInit: TNotifyEvent read FOnBeforeInit write FOnBeforeInit;
    property OnPatternRecognized: TNotifyEvent read FOnPatternRecognized write
FOnPatternRecognized;
    property PatternCount: integer read FPatternCount write SetPatternCount;
  end;

  // Клас мережі
  TNeuralNetBP = class(TNeuralNet)
  private
    // Коефіцієнт крутості граничної сигмоїдальної функції
    FAlpha: double;
    // Прапор автоініціалізації топології мережі
    FAutoInit: boolean;
    // Прапор продовження навчання
    FContinueTeach: boolean;
    // Бажаний вихід нейромережі розмірність OutputNeuronCount
    FDesiredOut: TVectorFloat;
    // Прапор зупинки при досягненні FEpochCount
    FEpoch: boolean;
    // Лічильник етапів (пред'явлення мережі всіх прикладів з навчальної
вибірки)
    FEpochCount: integer;
    // Номер поточної епохи
    FEpochCurrent: integer;
    // Значення помилки, при якій приклад вважається розпізнаним
    FIdentError: double;
    // Значення максимальної помилки на навчальній множині
    FMaxTeachResidual: double;
    // Значення максимальної помилки на тестовій множині
    FMaxTestResidual: double;
    // Значення середньої помилки на навчальній множині
    FMidTeachResidual: double;
    // Значення середньої помилки на тестовій множині
    FMidTestResidual: double;
    // Помилка на навчальній множині
    FTeachError: double;
    // Коефіцієнт інерційності
    FMomentum: double;
    // Кількість нейронів у шарах
    FNeuronsInLayer: TStrings;
    // Подія після ініціалізації
    FOnAfterInit: TNotifyEvent;
```

```
    FOnAfterNeuronCreated: TNotifyEvent;
    // Подія після навчання
    FOnAfterTeach: TNotifyEvent;
    // Подія до ініціалізації
    FOnBeforeInit: TNotifyEvent;
    // Подія до початку навчання
    FOnBeforeTeach: TNotifyEvent;
    // Подія після проходження одного етапу
    FOnEpochPassed: TNotifyEvent;
    // Число прикладів у навчальній безлічі
    FPatternCount: integer;
    // Масив утримуючий псевдовипадкову послідовність
    FRandomOrder: TVectorInt;
    // Лічильник розпізнаних прикладів на навчальній множині
    FRecognizedTeachCount: integer;
    // Лічильник розпізнаних прикладів на навчальній множині
    FRecognizedTestCount: integer;
    // Прапор зупинки навчання
    FStopTeach: boolean;
    FTeachStopped: boolean;
    // Коефіцієнт швидкості навчання - величина градієнтного кроку
    FTeachRate: double;
    // Число прикладів у тестовій множині
    FTestSetPatternCount: integer;
    // Розмірність FTestSetPatternCount, InputNeuronCount
    FTestSetPatterns: TMatrixFloat;
    // Розмірність FTestSetPatternCount, InputNeuronCount
    FTestSetPatternsOut: TMatrixFloat;
    function GetDesiredOut(Index: integer): double;
    function GetLayersBP(Index: integer): TLayerBP;
    function GetTestSetPatterns(InputIndex, PatternIndex: integer): double;
    function GetTestSetPatternsOut(InputIndex, PatternIndex: integer): double;
    procedure NeuronCountError;
    procedure NeuronsInLayerChange(Sender: TObject);
    procedure SetAlpha(Value: double);
    procedure SetDesiredOut(Index: integer; Value: double);
    procedure SetEpochCount(Value: integer);
    procedure SetLayersBP(Index: integer; Value: TLayerBP);
    procedure SetMomentum(Value: double);
    procedure SetTeachRate(Value: double);
    procedure SetTestSetPatternCount(const Value: integer);
    procedure SetTestSetPatterns(InputIndex, PatternIndex: integer; const Value:
double);
    procedure SetTestSetPatternsOut(InputIndex, PatternIndex: integer; const
Value: double);
    // Перетасування набору даних
    procedure Shuffle;
  protected
    function GetLayerCount: integer; override;
    function GetOutput(Index: integer): double; virtual;
    // Активаційна функція
    function Activation(Value: double): double; virtual;
    // Похідна активаційної функції
    function Activation(Value: double): double; virtual;
    // Середня квадратична помилка
    function QuadError: double; virtual;
    // Підстроювання ваг
    procedure AdjustWeights; virtual;
    // Розраховує  локальну помилку - дельту
    procedure CalcLocalError; virtual;
    // Перевірка мережі на тестовій множині
    procedure CheckTestSet; virtual;
    procedure DoOnAfterInit; virtual;
    procedure DoOnAfterNeuronCreated(ALayerIndex, ANeuronIndex: integer);
virtual;
    procedure DoOnAfterTeach; virtual;
    procedure DoOnBeforeInit; virtual;
    procedure DoOnBeforeTeach; virtual;
    procedure DoOnEpochPassed; virtual;
```

```
    // Ініціалізація ваг мережі псевдовипадковими значеннями
    procedure InitWeights; virtual;
    // Пред'явлення мережі вхідних значень приклада
    procedure LoadPatternsInput(APatternIndex :integer); virtual;
    // Пред'явлення мережі вхідних значень приклада
    procedure LoadPatternsOutput(APatternIndex :integer); virtual;
    // Поширює сигнал у прямому напрямку
    procedure Propagate; virtual;
    // Установка значень за замовчуванням
    procedure SetDefaultProperties; virtual;
    procedure SetPatternCount(const Value: integer); override;
    // Струс мережі
    procedure ShakeUp; virtual;
    property TeachStopped: boolean read FTeachStopped write FTeachStopped;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure AddLayer(ANeurons: integer); override;
    procedure Compute(AVector: TVectorFloat); virtual;
    procedure DeleteLayer(Index: integer); override;
    procedure Init; reintroduce; overload;
    procedure ResetLayers; virtual;
    procedure TeachOffLine; virtual;
    property DesiredOut[Index: integer]: double read GetDesiredOut write
SetDesiredOut;
    property EpochCurrent: integer read FEpochCurrent;
    property IdentError: double read FIdentError write FIdentError;
    property LayersBP[Index: integer]: TLayerBP read GetLayersBP write
SetLayersBP;
    property LayerCount: integer read GetLayerCount write SetLayerCount;
    property Output[Index: integer]: double read GetOutput;
    property StopTeach: boolean read FStopTeach write FStopTeach;
    property TeachError: double read FTeachError;
    property MaxTeachResidual: double read FMaxTeachResidual;
    property MaxTestResidual: double read FMaxTestResidual;
    property MidTeachResidual: double read FMidTeachResidual;
    property MidTestResidual: double read FMidTestResidual;
    property RecognizedTeachCount: integer read FRecognizedTeachCount;
    property RecognizedTestCount: integer read FRecognizedTestCount;
    property TestSetPatternCount: integer read FTestSetPatternCount write
SetTestSetPatternCount;
    property TestSetPatterns[InputIndex: integer; PatternIndex: integer]: double
read GetTestSetPatterns write SetTestSetPatterns;
    property TestSetPatternsOut[InputIndex: integer; PatternIndex: integer]:
double read GetTestSetPatternsOut write SetTestSetPatternsOut;
  published
    property Alpha: double read FAlpha write SetAlpha;
    property AutoInit: boolean read FAutoInit write FAutoInit;
    property ContinueTeach: boolean read FContinueTeach write FContinueTeach;
    property Epoch: boolean read FEpoch write FEpoch;
    property EpochCount: integer read FEpochCount write SetEpochCount;
    property Momentum: double read FMomentum write SetMomentum;
    property NeuronsInLayer: TStrings read FNeuronsInLayer write
FNeuronsInLayer;
    property OnAfterInit: TNotifyEvent read FOnAfterInit write FOnAfterInit;
    property OnAfterNeuronCreated: TNotifyEvent read FOnAfterNeuronCreated write
FOnAfterNeuronCreated;
    property OnAfterTeach: TNotifyEvent read FOnAfterTeach write FOnAfterTeach;
    property OnBeforeInit: TNotifyEvent read FOnBeforeInit write FOnBeforeInit;
    property OnBeforeTeach: TNotifyEvent read FOnBeforeTeach write
FOnBeforeTeach;
    property OnEpochPassed: TNotifyEvent read FOnEpochPassed write
FOnEpochPassed;
    property PatternCount: integer read FPatternCount write SetPatternCount;
    property TeachRate: double read FTeachRate write SetTeachRate;
  end;

  // Клас мережі back-propagation TNeuralNetExtended }
  TNeuralNetExtended = class(TNeuralNetBP)
```

```
private
  // Файл даних
  FNeuroDataSource: TNeuroDataSource;
  // Ім'я файлу даних *.txt
  FSourceFileName: TFileName;
  // Ім'я конфігураційного файлу *.nnw
  FFileName: TFileName;
  // Конфігураційний файл
  FNnwFile: TIniFile;
  // Поля
  FFields: TNeuroFields;
  // Кількість доступних полів
  FAvailableFieldsCount: integer;
  FMaxTeachError: boolean;
  FMaxTeachErrorValue: double;
  FMaxTestError: boolean;
  FMaxTestErrorValue: double;
  FMidTeachError: boolean;
  FMidTeachErrorValue: double;
  FMidTestError: boolean;
  FMidTestErrorValue: double;
  FOptions: string;
  FSettingsLoaded: boolean;
  FTestAsValid: boolean;
  FTeachIdent: boolean;
  FTeachIdentCount: integer;
  FTestIdent: boolean;
  FTestIdentCount: integer;
  FUseForTeach: integer;
  FIdentError: double;
  FRealOutputIndex: TVectorInt;
  FRealInputIndex: TVectorInt;
  function GetFields(Index: integer): TNeuroField;
  function GetInputFieldCount: integer;
  function GetOutputFieldCount: integer;
  function GetRealInputIndex(Index: integer): integer;
  function GetRealOutputIndex(Index: integer): integer;
  procedure SetFields(Index: integer; Value: TNeuroField);
  procedure SetFileName(Value: TFilename);
  procedure SetAvailableFieldsCount(Value: integer);
  procedure SetUseForTeach(const Value: integer);
  procedure SetTeachIdentCount(const Value: integer);
  procedure SetRealOutputIndex(Index: integer; const Value: integer);
  procedure SetRealOutputIndexCount(const Value: integer);
  procedure SetRealInputIndex(Index: integer; const Value: integer);
  procedure SetRealInputIndexCount(const Value: integer);
protected
  function GetOutput(Index: integer): double; override;
  procedure DoOnBeforeTeach; override;
  procedure DoOnEpochPassed; override;
  procedure SetDefaultProperties; override;
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  procedure ComputeUnPrepData(AVector: TVectorFloat);
  // Завантажує дані з текстового файлу
  procedure LoadDataFrom;
  // Завантажує настроювання мережі
  procedure LoadNetwork;
  // Завантажує настроювання мережі
  procedure LoadPhase1;
  // Завантажує настроювання мережі
  procedure LoadPhase2;
  // Завантажує настроювання мережі
  procedure LoadPhase4;
  // Нормалізує набір даних
  procedure NormalizeData;
  // Зберігає настроювання мережі
  procedure SaveNetwork;
```

```
    // Зберігає настроювання мережі
    procedure SavePhase1;
    // Зберігає настроювання мережі
    procedure SavePhase2;
    // Зберігає настроювання мережі
    procedure SavePhase4;
    // Навчання нейронної мережі
    procedure Train;
    property AvailableFieldsCount: integer read FAvailableFieldsCount write
SetAvailableFieldsCount;
    property Fields[Index: integer]: TNeuroField read GetFields write SetFields;
    property InputFieldCount: integer read GetInputFieldCount;
    property OutputFieldCount: integer read GetOutputFieldCount;
    property SettingsLoaded: boolean read FSettingsLoaded write FSettingsLoaded;
    property RealOutputIndex[Index: integer]: integer read GetRealOutputIndex
write SetRealOutputIndex;
    property RealOutputIndexCount: integer write SetRealOutputIndexCount;
    property RealInputIndex[Index: integer]: integer read GetRealInputIndex
write SetRealInputIndex;
    property RealInputIndexCount: integer write SetRealInputIndexCount;
    property NnwFile: TIniFile read FNnwFile write FNnwFile;
  published
    property FileName: TFileName read FFileName write SetFileName;
    property IdentError: double read FIdentError write FIdentError;
    property MaxTeachError: boolean read FMaxTeachError write FMaxTeachError;
    property MaxTeachErrorValue: double read FMaxTeachErrorValue write
FMaxTeachErrorValue;
    property MaxTestError: boolean read FMaxTestError write FMaxTestError;
    property MaxTestErrorValue: double read FMaxTestErrorValue write
FMaxTestErrorValue;
    property MidTeachError: boolean read FMidTeachError write FMidTeachError;
    property MidTeachErrorValue: double read FMidTeachErrorValue write
FMidTeachErrorValue;
    property MidTestError: boolean read FMidTestError write FMidTestError;
    property MidTestErrorValue: double read FMidTestErrorValue write
FMidTestErrorValue;
    property Options: string read FOptions write FOptions;
    property SourceFileName: TFileName read FSourceFileName write
FSourceFileName;
    property TestAsValid: boolean read FTestAsValid write FTestAsValid;
    property TeachIdent: boolean read FTeachIdent write FTeachIdent;
    property TeachIdentCount: integer read FTeachIdentCount write
SetTeachIdentCount;
    property TestIdent: boolean read FTestIdent write FTestIdent;
    property TestIdentCount: integer read FTestIdentCount write FTestIdentCount;
    property UseForTeach: integer read FUseForTeach write SetUseForTeach;
  end;


procedure Register;

implementation

{$R *.RES}

{ TNeuron }

constructor TNeuron.Create(ALayer: TLayer);
begin
  inherited Create;
  // покажчик на шар у якому перебуває нейрон
  Layer := ALayer;
end;

destructor TNeuron.Destroy;
begin
  WeightCount := 0;
  FWeights := nil;
  Layer := nil;
```

```
  inherited;
end;

procedure TNeuron.ComputeOut(const AInputs: TVectorFloat);
var
  i: integer;
begin
  FOutput := 0;
  // Підраховується зважена сума нейрона
  for i := Low(AInputs) to High(AInputs) do
    FOutput := FOutput + FWeights[i] * AInputs[i];
end;

function TNeuron.GetWeights(Index: integer): double;
begin
  try
    Result := FWeights[Index];
  except
    on E: ERangeError do
      raise E.CreateFmt(SWeightRangeIndex, [Index])
  end;
end;

procedure TNeuron.InitWeights;
var
  i: integer;
begin
  // Ініціалізація ваг нейрона
  for i := Low(FWeights) to High(FWeights) do
    FWeights[i] := Random
end;

procedure TNeuron.SetWeightCount(const Value: integer);
begin
  SetLength(FWeights, Value);
end;

procedure TNeuron.SetWeights(Index: integer; Value: double);
begin
  try
    FWeights[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SWeightRangeIndex, [Index])
  end;
end;

{ Кінець опису TNeuron }

{ TNeuronHopf }

procedure TNeuronHopf.ComputeOut(const AInputs: TVectorFloat);
begin
  inherited;
  // гранична функція
  if FOutput >= 0 then
    FOutput := 1
  else
    FOutput := -1
end;

{ Кінець опису TNeuronHopf}

{ TNeuronBP }

destructor TNeuronBP.Destroy;
begin
  FOnActivation := nil;
  FOnActivation := nil;
```

```
  PrevUpdateCount := 0;
  FPrevUpdate := nil;
  inherited;
end;

function TNeuronBP.GetLearningRate(Index: integer): double;
begin
  Result := FLearningRate[Index];
end;

function TNeuronBP.GetPrevDerivative(Index: integer): double;
begin
  Result := FPrevDerivative[Index];
end;

function TNeuronBP.GetPrevUpdateCount: integer;
begin
  Result := High(FPrevUpdate) + 1;
end;

function TNeuronBP.GetPrevUpdate(Index: integer): double;
begin
  Result := FPrevUpdate[Index];
end;

procedure TNeuronBP.ComputeOut(const AInputs: TVectorFloat);
begin
  inherited;
  // Задає зсув нейрона
  FOutput := FOutput + Weights[High(AInputs) + 1];
  FOutput := OnActivation(FOutput);
end;

procedure TNeuronBP.SetDelta(Value: double);
begin
  FDelta := Value;
end;

procedure TNeuronBP.SetLearningRate(Index: integer; const Value: double);
begin
  FLearningRate[Index] := Value;
end;

procedure TNeuronBP.SetLearningRateCount(const Value: integer);
begin
  SetLength(FLearningRate, Value)
end;

procedure TNeuronBP.SetPrevUpdate(Index: integer; Value: double);
begin
  FPrevUpdate[Index] := Value;
end;

procedure TNeuronBP.SetPrevUpdateCount(const Value: integer);
begin
  SetLength(FPrevUpdate, Value)
end;

procedure TNeuronBP.SetPrevDerivative(Index: integer; const Value: double);
begin
  FPrevDerivative[Index] := Value;
end;

procedure TNeuronBP.SetPrevDerivativeCount(const Value: integer);
begin
  SetLength(FPrevDerivative, Value)
end;

{ Кінець опису TNeuronBP }
```

```
{ TLayer }

procedure TLayer.Assign(Source: TPersistent);
var
  i: integer;
begin
  FNumber := (Source as TLayer).FNumber;
  NeuronCount := (Source as TLayer).NeuronCount;
  // Створюються нейрони
  for i := 0 to NeuronCount - 1 do
    FNeurons[i] := TNeuron.Create(Self);
end;

constructor TLayer.Create(ALayerNumber: integer; ANeuronCount: integer);
var
  i: integer;
begin
  inherited Create;
  FNumber := ALayerNumber;
  NeuronCount := ANeuronCount;
  for i := 0 to ANeuronCount - 1 do
    FNeurons[i] := TNeuron.Create(Self);
end;

destructor TLayer.Destroy;
var
  i: integer;
begin
  for i := 0 to NeuronCount - 1 do
    FNeurons[i].Free;
  NeuronCount := 0;
  FNeurons := nil;
  inherited;
end;

function TLayer.GetNeuronCount: integer;
begin
  Result := High(FNeurons) + 1;
end;

function TLayer.GetNeurons(Index: integer): TNeuron;
begin
  Result := FNeurons[Index];
end;

procedure TLayer.SetNeuronCount(Value: integer);
begin
  if Value <> High(FNeurons) + 1 then
    SetLength(FNeurons, Value);
end;

procedure TLayer.SetNeurons(Index: integer; Value: TNeuron);
begin
  try
    FNeurons[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SNeuronRangeIndex, [Index])
  end;
end;

{ TLayerHopf }

constructor TLayerHopf.Create(ALayerNumber: integer; ANeuronCount: integer);
var
  i: integer;
begin
  FNumber := ALayerNumber;
```

```
  NeuronCount := ANeuronCount;
  for i := 0 to ANeuronCount - 1 do
    FNeurons[i] := TNeuronHopf.Create(Self);
end;

{ TLayerBP }

procedure TLayerBP.Assign(Source: TPersistent);
var
  i: integer;
begin
  FNumber := (Source as TLayerBP).FNumber;
  NeuronCount := (Source as TLayerBP).NeuronCount;
  for i := 0 to NeuronCount - 1 do
    FNeurons[i] := TNeuronBP.Create(Self);
end;

constructor TLayerBP.Create(ALayerNumber: integer; ANeuronCount: integer);
var
  i: integer;
begin
  FNumber := ALayerNumber;
  NeuronCount := ANeuronCount;
  for i := 0 to ANeuronCount - 1 do
    FNeurons[i] := TNeuronBP.Create(Self);
end;

destructor TLayerBP.Destroy;
begin
  inherited;
end;

function TLayerBP.GetNeuronsBP(Index: integer): TNeuronBP;
begin
  Result := FNeurons[Index] as TNeuronBP;
end;

procedure TLayerBP.SetNeuronsBP(Index: integer; Value: TNeuronBP);
begin
  FNeurons[Index] := Value as TNeuronBP;
end;

{ TNeuralNet }

destructor TNeuralNet.Destroy;
begin
  Clear;
  SetLength(FPatternsInput, 0, 0);
  FPatternsInput := nil;
  SetLength(FPatternsOutput, 0 ,0);
  FPatternsOutput := nil;
  FLayers := nil;
  inherited;
end;

procedure TNeuralNet.Clear;
var
  i, xCount: integer;
begin
  xCount := LayerCount;
  if  xCount > 0 then
  begin
    for i := 0 to xCount - 1 do
      FLayers[i].Free;
    LayerCount := 0;
  end;
end;

function TNeuralNet.GetInputNeuronCount: integer;
```

```
begin
  Result := Layers[SensorLayer].NeuronCount;
end;

function TNeuralNet.GetLayerCount: integer;
begin
  Result := High(FLayers) + 1;
end;

function TNeuralNet.GetLayers(Index: integer): TLayer;
begin
  Result := FLayers[Index];
end;

function TNeuralNet.GetOutputNeuronCount: integer;
begin
  Result := Layers[LayerCount - 1].NeuronCount;
end;

function TNeuralNet.GetPatternsInput(PatternIndex: integer; InputIndex:
integer): double;
begin
  Result := FPatternsInput[PatternIndex, InputIndex];
end;

procedure TNeuralNet.AddPattern(const AInputs: TVectorFloat; const AOutputs:
TVectorFloat);
var
  i: integer;
begin
  if InputNeuronCount <> High(AInputs) + 1 then
    raise EInOutDimensionError.Create(SInVectorCount);
  if OutputNeuronCount <> High(AOutputs) + 1 then
    raise EInOutDimensionError.Create(SOutVectorCount);
  PatternCount := PatternCount + 1;
  ResizeInputDim;
  ResizeOutputDim;
  for i := Low(AInputs) to High(AInputs) do
    PatternsInput[PatternCount - 1, i] := AInputs[i];
  for i := Low(AOutputs) to High(AOutputs) do
    PatternsOutput[PatternCount - 1, i] := AOutputs[i];
end;

procedure TNeuralNet.DeletePattern(Index: integer);
var
  i, j: integer;
begin
  try
    // видаляє вхідні значення приклада Index
    for i := Index to FPatternCount - 2 do
      for j := 0 to InputNeuronCount - 1 do
        FPatternsInput[i, j] := FPatternsInput[i + 1, j];
    // видаляє вихідні значення приклада Index
    for i := Index to FPatternCount - 2 do
      for j := 0 to OutputNeuronCount - 1 do
        FPatternsOutput[i, j] := FPatternsOutput[i + 1, j];
    Dec(FPatternCount);
    ResizeInputDim;
    ResizeOutputDim;
  except
    on E: ERangeError do
      raise E.CreateFmt(SPatternRangeIndex, [Index])
  end;
end;

procedure TNeuralNet.ResetPatterns;
begin
  FPatternCount := DefaultPatternCount;
  ResizeInputDim;
```

```
  ResizeOutputDim;
end;

procedure TNeuralNet.SetPatternCount(const Value: integer);
begin
  if Value < DefaultPatternCount then
    FPatternCount := DefaultPatternCount
  else
    FPatternCount := Value;
  ResizeInputDim;
  ResizeOutputDim;
end;

procedure TNeuralNet.SetPatternsOutput(PatternIndex: integer; InputIndex:
integer; Value: double);
begin
  FPatternsOutput[PatternIndex, InputIndex] := Value;
end;

procedure TNeuralNet.SetPatternsInput(PatternIndex: integer; InputIndex:
integer; Value: double);
begin
  FPatternsInput[PatternIndex, InputIndex] := Value;
end;

procedure TNeuralNet.Init(const ANeuronsInLayer: TVectorInt);
var
  i,j: integer;
begin
  LayerCount := High(ANeuronsInLayer) + 1;
  // FLayers[0] нульовий шар і виконує роль розподільного,
  // використовується тільки поле Output
  FLayers[0] := TLayer.Create(0, ANeuronsInLayer[0]);
  // для нульового шару не потрібні вагові коефіцієнти
  for i := 1 to LayerCount - 1 do
  begin
    FLayers[i] := TLayer.Create(i, ANeuronsInLayer[i]);
    for j := 0 to ANeuronsInLayer[i] - 1 do
    with FLayers[i].FNeurons[j] do
      // задає кількість елементів у векторі ваг нейрона j в
      //  шарі i рівним кількості виходів попереднього шару
      WeightCount := FLayers[ i-1].NeuronCount;
  end;
end;

procedure TNeuralNet.ResizeInputDim;
begin
  SetLength(FPatternsInput, FPatternCount, InputNeuronCount)
end;

procedure TNeuralNet.ResizeOutputDim;
begin
  SetLength(FPatternsOutput, FPatternCount, OutputNeuronCount)
end;

procedure TNeuralNet.SetLayerCount(Value: integer);
begin
  SetLength(FLayers, Value);
end;

procedure TNeuralNet.SetLayers(Index: integer; Value: TLayer);
begin
  try
    FLayers[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SLayerRangeIndex, [Index])
  end;
end;
```

```
{  TNeuralNetHopf }

constructor TNeuralNetHopf.Create(AOwner: TComponent);
begin
  inherited;
  PatternCount := DefaultPatternCount;
  InputNeuronCount := DefaultNeuronCount;
  MaxIterCount := DefaultMaxIterCount;
  AutoInit := False;
end;

destructor TNeuralNetHopf.Destroy;
begin
  FOnAfterInit := nil;
  FOnBeforeInit := nil;
  FOnPatternRecognized := nil;
  SetLength(FPatterns, 0, 0);
  FPatterns := nil;
  inherited;
end;

function TNeuralNetHopf.GetInput(Index: integer): double;
begin
  Result := Layers[1].Neurons[Index].Output;
end;

function TNeuralNetHopf.GetInputNeuronCount: integer;
begin
  Result := Layers[SensorLayer].NeuronCount;
end;

function TNeuralNetHopf.GetLayerCount: integer;
begin
  Result := DefaultHopfLayerCount;
end;

function TNeuralNetHopf.GetPatterns(InputIndex: integer; PatternIndex: integer):
integer;
begin
  Result := FPatterns[InputIndex, PatternIndex];
end;

function TNeuralNetHopf.Stabled: boolean;
var
  i: integer;
begin
  // Порівнює вихідні значення попередньої
  //  ітерації зі значеннями поточної
  Result := True;
  for i := 0 to InputNeuronCount - 1  do
    if FLayers[1].FNeurons[i].FOutput <> FLayers[0].FNeurons[i].FOutput then
    begin
      Result := False;
      Exit
    end;
end;

procedure TNeuralNetHopf.AddPattern(const ANewPattern: TVectorInt);
var
  i: integer;
begin
  if InputNeuronCount <> High(ANewPattern)+ 1 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  PatternCount := PatternCount + 1;
  ResizePatternsDim;
  for i := 0 to FInputNeuronCount - 1 do
    FPatterns[FPatternCount - 1, i] := ANewPattern[i];
  if AutoInit then
```

```
    InitWeights;
end;

procedure TNeuralNetHopf.Calc;
var
  i: integer;
  xCurrentIter: integer;
  xArray: TVectorFloat;
begin
  SetLength(xArray, InputNeuronCount);
  // Цикл працює поки не стабілізуються виходи
  xCurrentIter := 0;
  repeat
    for i := 0 to InputNeuronCount - 1 do
    begin
      // Запам'ятовує попередній крок ітерації, для
      //  цього використовується нульовий шар
      Layers[SensorLayer].Neurons[i].Output := Layers[1].Neurons[i].Output;
      xArray[i] := Layers[1].Neurons[i].Output;
    end;
    for i := 0 to InputNeuronCount - 1 do
      with Layers[1].Neurons[i] do
        // Розраховується новий стан нейронів і аксонів
        ComputeOut(xArray);
    Inc(xCurrentIter);
  until Stabled or (MaxIterCount = xCurrentIter);
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
  SetLength(xArray, 0);
  xArray := nil;
end;

procedure TNeuralNetHopf.DeletePattern(Index: integer);
var
  i, j: integer;
begin
  try
    for i := Index to FPatternCount - 2 do
      for j := 0 to FInputNeuronCount - 1 do
        FPatterns[i, j] := FPatterns[i + 1, j];
    Dec(FPatternCount);
    ResizePatternsDim;
    if AutoInit then
      InitWeights;
  except
    on E: ERangeError do
      raise E.CreateFmt(SPatternRangeIndex, [Index])
  end;
end;

procedure TNeuralNetHopf.Init;
var
  i, j: integer;
begin
  if Assigned(FOnBeforeInit) then
    FOnBeforeInit(Self);
  LayerCount := DefaultHopfLayerCount;
  for i := 0 to LayerCount - 1 do
    FLayers[i] := TLayerHopf.Create(i, FInputNeuronCount);
  // Для нульового шару не потрібні вагові коефіцієнти
  for j := 0 to FInputNeuronCount - 1 do
    with FLayers[1].FNeurons[j] do
      // задає кількість елементів у векторі
      WeightCount := FInputNeuronCount;
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
end;

procedure TNeuralNetHopf.InitWeights;
```

```
var
  i, j, k : integer;
begin
  // Ініціалізує вагову матрицю
  for i := 0 to InputNeuronCount - 1 do
    for j := 0 to InputNeuronCount - 1 do
    with Layers[1].Neurons[i] do
    begin
      Weights[j] := 0;
      if i <> j then
        for k := 0 to PatternCount - 1 do
          Weights[j] := Weights[j]+ Patterns[k, i] * Patterns[k, j]
    end;
end;

procedure TNeuralNetHopf.ResetPatterns;
begin
  PatternCount := DefaultPatternCount;
  if AutoInit then
    InitWeights;
end;

procedure TNeuralNetHopf.ResizePatternsDim;
begin
  SetLength(FPatterns, FPatternCount, FInputNeuronCount);
end;

procedure TNeuralNetHopf.SetInput(Index: integer; Value: double);
begin
  try
    Layers[1].Neurons[Index].Output := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SPatternRangeIndex, [Index])
  end;
end;

procedure TNeuralNetHopf.SetInputNeuronCount(Value: integer);
begin
  if Value > DefaultNeuronCount then
    FInputNeuronCount := Value
  else
    FInputNeuronCount := DefaultNeuronCount;
  ResizePatternsDim;
  Init;
end;

procedure TNeuralNetHopf.SetPatternCount(const Value: integer);
begin
  if Value < DefaultPatternCount then
    FPatternCount := DefaultPatternCount
  else
    FPatternCount := Value;
end;

procedure TNeuralNetHopf.SetPatterns(InputIndex: integer; PatternIndex: integer;
Value: integer);
begin
  FPatterns[InputIndex, PatternIndex] := Value;
end;

{ TNeuralNetBP }

constructor TNeuralNetBP.Create(AOwner: TComponent);
var
  i: integer;
begin
  inherited;
  FNeuronsInLayer := TStringList.Create;
```

```
  for i := 0 to DefaultLayerCount do
    AddLayer(DefaultNeuronCount);
  TStringList(FNeuronsInLayer).OnChange := NeuronsInLayerChange;
  AutoInit := True;
  StopTeach := False;
  TeachStopped := False;
  NeuronsInLayerChange(Self);
  SetDefaultProperties;
end;

destructor TNeuralNetBP.Destroy;
begin
  FNeuronsInLayer.Free;
  SetLength(FRandomOrder, 0);
  FRandomOrder := nil;
  SetLength(FDesiredOut, 0);
  FDesiredOut := nil;
  SetLength(FTestSetPatterns, 0, 0);
  FTestSetPatterns := nil;
  SetLength(FTestSetPatternsOut, 0, 0);
  FTestSetPatternsOut := nil;
  FOnAfterInit := nil;
  FOnAfterTeach := nil;
  FOnBeforeInit := nil;
  FOnBeforeTeach := nil;
  FOnEpochPassed := nil;
  inherited;
end;

function TNeuralNetBP.GetLayersBP(Index: integer): TLayerBP;
begin
  Result := FLayers[Index] as TLayerBP;
end;

function TNeuralNetBP.GetLayerCount: integer;
begin
  Result := High(FLayers) + 1;
end;

function TNeuralNetBP.GetDesiredOut(Index: integer): double;
begin
  Result := FDesiredOut[Index];
end;

function TNeuralNetBP.GetOutput(Index: integer): double;
begin
  try
    Result := LayersBP[LayerCount - 1].NeuronsBP[Index].Output;
  except
    on E: ERangeError do
      raise E.CreateFmt(SNeuronRangeIndex, [Index])
  end;
end;

function TNeuralNet.GetPatternsOutput(PatternIndex: integer; OutputIndex:
integer): double;
begin
  Result := FPatternsOutput[PatternIndex, OutputIndex];
end;

function TNeuralNetBP.QuadError: double;
var
  i: integer;
begin
  // розраховує середньоквадратичну помилку
  Result := 0;
  for i := 0 to OutputNeuronCount - 1 do
    Result := Result + sqr(LayersBP[LayerCount - 1].NeuronsBP[i].Output -
DesiredOut[i]);
```

```
  Result := Result/2;
end;

function TNeuralNetBP.Activation(Value: double): double;
begin
  // Активаційна функція - сигмоїд
  Result := 1/( 1 + exp(-FAlpha * Value) )
end;

function TNeuralNetBP.Activation(Value: double): double;
begin
  // Похідна сигмоїди
  Result := FAlpha * Value * (1 - Value)
end;

function TNeuralNetBP.GetTestSetPatterns(InputIndex, PatternIndex: integer):
double;
begin
  Result := FTestSetPatterns[InputIndex, PatternIndex];
end;

function TNeuralNetBP.GetTestSetPatternsOut(InputIndex, PatternIndex: integer):
double;
begin
  Result := FTestSetPatternsOut[InputIndex, PatternIndex];
end;

procedure TNeuralNetBP.AddLayer(ANeurons: integer);
begin
  if ANeurons < DefaultNeuronCount then
    NeuronCountError
  else
    NeuronsInLayer.Add(IntToStr(ANeurons));
end;

procedure TNeuralNetBP.AdjustWeights;
var
  i, j, k: integer;
  xCurrentUpdate: double;
begin
  // Підстроювання ваг починаючи з першого шару
  for i := 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
    begin
      for k := 0 to LayersBP[ i-1].NeuronCount  do
      with LayersBP[i].NeuronsBP[j] do
      begin
        // коректує вагу з'єднуючого j-нейрона шару i
        //  з k-нейроном шару i-1:  добутком дельта j-нейрона
        //  на вихід k-нейрона шару i-1
        if k = LayersBP[ i-1].NeuronCount then
           // якщо це нейрон, що задає зсув
           xCurrentUpdate := -TeachRate * Delta + Momentum * PrevUpdate[k]
        else
           xCurrentUpdate := -TeachRate * Delta *
           LayersBP[ i-1].NeuronsBP[k].Output + Momentum * PrevUpdate[k];
        Weights[k]:= Weights[k] + xCurrentUpdate;
        PrevUpdate[k] := xCurrentUpdate;
      end;
    end
end;

procedure TNeuralNetBP.CalcLocalError;
var
  i, j, k: integer;
begin
  // Дельта-правило з останнього шару до першого
  for i := LayerCount - 1 downto 1 do
    // для останнього шару
```

```
    if i = LayerCount - 1 then
      for j := 0 to LayersBP[i].NeuronCount - 1 do
        LayersBP[i].NeuronsBP[j].Delta := (LayersBP[i].NeuronsBP[j].Output-
DesiredOut[j])
        * Activation(LayersBP[i].NeuronsBP[j].Output)
    else
      for j := 0 to LayersBP[i].NeuronCount - 1 do
        with LayersBP[i].NeuronsBP[j] do
        begin
          Delta := 0;
          // Підсумує добуток локальної помилки k-нейрона шару i+1
          //  на вагу з'єднуючий k-нейрон шару i+1 з j-нейроном шару i
          for k := 0 to LayersBP[i+1].NeuronCount - 1 do
            Delta := Delta + LayersBP[i+1].NeuronsBP[k].Delta *
                             LayersBP[i+1].NeuronsBP[k].Weights[j];
          Delta := Delta * Activation(Output)
        end;
end;

procedure TNeuralNetBP.CheckTestSet;
var
  i, j: integer;
  xArray: TVectorFloat;
  xFirstTestSample: boolean;
  xQuadError: double;
  // функція розраховує середньоквадратичну помилку
  function QuadError(APatternCount: integer): double;
  var
    i: integer;
  begin
    Result := 0;
    for i := 0 to OutputNeuronCount - 1 do
      Result := Result + sqr(LayersBP[LayerCount - 1].NeuronsBP[i].Output -
TestSetPatternsOut[APatternCount, i]);
    Result := Result/2;
  end;
begin
  SetLength(xArray, InputNeuronCount);
  xFirstTestSample := True;
  FRecognizedTestCount := 0;
  FMidTestResidual := 0;
  FMaxTestResidual := 0;
  for i := 0 to TestSetPatternCount - 1 do
  begin
    for j := 0 to InputNeuronCount - 1 do
      xArray[j] := TestSetPatterns[i, j];
    Compute(xArray);
    xQuadError := QuadError(i);
    // перевірка - чи розпізнаний приклад з тестової множини
    if xQuadError < IdentError then
      Inc(FRecognizedTestCount);
    FMidTestResidual := FMidTestResidual + xQuadError;
    // максимальна помилка на тестовій множині
    if xFirstTestSample then
    begin
      FMaxTestResidual := xQuadError;
      xFirstTestSample := False;
    end
    else
      if FMaxTestResidual < xQuadError then
        FMaxTestResidual := xQuadError;
  end;
  // середня помилка на тестовій множині
  FMidTestResidual := FMidTestResidual/TestSetPatternCount;
  SetLength(xArray, 0);
  xArray := nil;
end;

procedure TNeuralNetBP.Compute(AVector: TVectorFloat);
```

```
var
  i: integer;
begin
  if InputNeuronCount <> High(AVector)+ 1 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  for i := Low(AVector) to High(AVector) do
    LayersBP[SensorLayer].NeuronsBP[i].Output :=  AVector[i];
  Propagate;
end;

procedure TNeuralNetBP.DoOnAfterInit;
begin
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
end;

procedure TNeuralNetBP.DoOnAfterNeuronCreated(ALayerIndex, ANeuronIndex:
integer);
var
  i: integer;
begin
  with LayersBP[ALayerIndex].NeuronsBP[ANeuronIndex] do
    for i := 0 to PrevUpdateCount - 1 do
      PrevUpdate[i] := 0;
  if Assigned(FOnAfterNeuronCreated) then
    FOnAfterNeuronCreated(Self);
end;

procedure TNeuralNetBP.DoOnAfterTeach;
begin
  if Assigned(FOnAfterTeach) then
    FOnAfterTeach(Self);
end;

procedure TNeuralNetBP.DoOnBeforeInit;
begin
  if Assigned(FOnBeforeInit) then
    FOnBeforeInit(Self);
end;

procedure TNeuralNetBP.DoOnBeforeTeach;
begin
  if Assigned(FOnBeforeTeach) then
    FOnBeforeTeach(Self);
end;

procedure TNeuralNetBP.DoOnEpochPassed;
begin
  if Assigned(FOnEpochPassed) then
    FOnEpochPassed(Self);
end;

procedure TNeuralNetBP.DeleteLayer(Index: integer);
var
  i: integer;
begin
  try
    NeuronsInLayer.Delete(Index);
    for i := Index to LayerCount - 2 do
      LayersBP[i].Assign(LayersBP[i + 1]);
    FLayers[LayerCount - 1].Free;
    LayerCount := LayerCount - 1;
  except
    on E: ERangeError do
      raise E.CreateFmt(SLayerRangeIndex, [Index])
  end;
end;

procedure TNeuralNetBP.Init;
```

```
var
  i, j: integer;
begin
  DoOnBeforeInit;
  if NeuronsInLayer.Count > 0 then
  begin
  LayerCount := NeuronsInLayer.Count;
    // FLayers[0] нульовий шар, використовується тільки поле Output
    FLayers[0] := TLayerBP.Create(0, StrToInt(NeuronsInLayer.Strings[0]));
    // для нульового шару не потрібні вагові коефіцієнти
    for i := 1 to LayerCount - 1 do
    begin
      FLayers[i] := TLayerBP.Create(i, StrToInt(NeuronsInLayer.Strings[i]));
      for j := 0 to StrToInt(NeuronsInLayer.Strings[i]) - 1 do
      with LayersBP[i].NeuronsBP[j] do
      begin
        // задає кількість елементів у векторі ваг + зсув
        WeightCount := LayersBP[ i-1].NeuronCount + BiasNeuron;
        // задає кількість  у векторі утримуючих попередню
        //  корекцію елементів  + зсув
        PrevUpdateCount := LayersBP[ i-1].NeuronCount + BiasNeuron;
        PrevDerivativeCount := LayersBP[ i-1].NeuronCount + BiasNeuron; // для
швидких алгоритмів
        LearningRateCount := LayersBP[ i-1].NeuronCount + BiasNeuron;   // для
швидких алгоритмів
        OnActivation := Activation;
        OnActivation := Activation;
        Randomize;
        DoOnAfterNeuronCreated(i, j);
      end
    end;
    // установлює розмірність масиву виходів
    //  число нейронів в останньому шарі = числу виходів
    SetLength(FDesiredOut, OutputNeuronCount);
  end;
  DoOnAfterInit;
end;

procedure TNeuralNetBP.InitWeights;
var
  i,j: integer;
begin
  Randomize;
  // Ініціалізація ваг
  for i := 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
       LayersBP[i].NeuronsBP[j].InitWeights;
end;

procedure TNeuralNetBP.LoadPatternsInput(APatternIndex :integer);
var
  i: integer;
begin
  for i := 0 to InputNeuronCount - 1 do
    LayersBP[SensorLayer].NeuronsBP[i].Output :=  PatternsInput[APatternIndex,
i];
end;

procedure TNeuralNetBP.LoadPatternsOutput(APatternIndex :integer);
var
  i: integer;
begin
  for i := 0 to OutputNeuronCount - 1 do
    DesiredOut[i] := PatternsOutput[APatternIndex, i];
end;

procedure TNeuralNetBP.NeuronsInLayerChange(Sender: TObject);
begin
  if AutoInit then
```

```
    Init;
end;

procedure TNeuralNetBP.NeuronCountError;
begin
  raise ENeuronCountError.Create(SNeuronCount)
end;

procedure TNeuralNetBP.Propagate;
var
  i, j, xIndex: integer;
  xArray: TVectorFloat;
begin
   // Поширення сигналу в прямому напрямку з першого шару
  for i := 1 to LayerCount - 1 do
  begin
    // формування масиву входів з виходів попереднього шару
    SetLength(xArray, LayersBP[ i-1].NeuronCount);
    for xIndex := 0 to LayersBP[ i-1].NeuronCount - 1 do
       xArray[xIndex] := LayersBP[ i-1].NeuronsBP[xIndex].Output;
    // обчислення виходу нейрона
    for j := 0 to LayersBP[i].NeuronCount - 1 do
      with LayersBP[i].NeuronsBP[j] do
        ComputeOut(xArray);
    for xIndex := 0 to LayersBP[ i-1].NeuronCount - 1 do
       xArray[xIndex] := 0;
  end;
  SetLength(xArray, 0);
  xArray := nil;
end;

procedure TNeuralNetBP.ResetLayers;
begin
  Clear;
  FNeuronsInLayer.Clear;
end;

procedure TNeuralNetBP.SetDesiredOut(Index: integer; Value: double);
begin
  FDesiredOut[Index] := Value;
end;

procedure TNeuralNetBP.SetLayersBP(Index: integer; Value: TLayerBP);
begin
  FLayers[Index] := Value as TLayerBP;
end;

procedure TNeuralNetBP.SetAlpha(Value: double);
begin
  if (Value > 10) or (Value < 0.01) then
    FAlpha := DefaultAlpha
  else
    FAlpha := Value;
end;

procedure TNeuralNetBP.SetTeachRate(Value: double);
begin
  if (Value > 1) or (Value <= 0) then
    FTeachRate := DefaultTeachRate
  else
    FTeachRate := Value;
end;

procedure TNeuralNetBP.SetTestSetPatterns(InputIndex, PatternIndex: integer;
const Value: double);
begin
  FTestSetPatterns[InputIndex, PatternIndex] := Value;
end;
```

```
procedure TNeuralNetBP.SetTestSetPatternsOut(InputIndex, PatternIndex: integer;
const Value: double);
begin
  FTestSetPatternsOut[InputIndex, PatternIndex] := Value;
end;

procedure TNeuralNetBP.SetTestSetPatternCount(const Value: integer);
begin
  FTestSetPatternCount := Value;
  SetLength(FTestSetPatterns, FTestSetPatternCount, InputNeuronCount);
  SetLength(FTestSetPatternsOut, FTestSetPatternCount, OutputNeuronCount);
end;

procedure TNeuralNetBP.SetMomentum(Value: double);
begin
  if (Value > 1) or (Value < 0) then
    FMomentum := DefaultMomentum
  else
    FMomentum := Value;
end;

procedure TNeuralNetBP.SetEpochCount(Value: integer);
begin
  if Value < 1 then
    FEpochCount := 1
  else
    FEpochCount := Value;
end;

procedure TNeuralNetBP.ShakeUp;
var
  i, j, k: integer;
begin
  Randomize;
  for i := 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
      for k := 0 to LayersBP[ i-1].NeuronCount  do
        with LayersBP[i].NeuronsBP[j] do
          Weights[k]:= Weights[k] + Random*0.1-0.05;
end;

procedure TNeuralNetBP.Shuffle;
var
  i, j, xNewInd, xLast: integer;
  xIsUnique : boolean;
begin
  xNewInd := 0;
  FRandomOrder[0] := Round(Random(FPatternCount));
  xLast := 0;
  for i := 1 to PatternCount - 1 do
  begin
    xIsUnique := False;
    while not xIsUnique do
    begin
      xNewInd := Round((Random(FPatternCount)));
      xIsUnique := True;
      for j := 0 to xLast do
        if xNewInd = FRandomOrder[j] then
          xIsUnique := False;
    end;
    FRandomOrder[i] := xNewInd;
    xLast := xLast +1;
  end;
end;

procedure TNeuralNetBP.TeachOffLine;
var
  j: integer;
  xQuadError: double;
```

```
  xNewEpoch: boolean;
begin
  DoOnBeforeTeach;
  if not ContinueTeach then
  begin
    // ваги ініціалізуються, якщо мережа навчається з "нуля"
    InitWeights;
    FEpochCurrent := 1;
  end;
  Randomize;
  SetLength(FRandomOrder, FPatternCount);
  TeachStopped := False;
  while (FEpochCurrent <= EpochCount) do
  begin
    FTeachError := 0;
    FMaxTeachResidual := 0;
    FRecognizedTeachCount := 0;
    xNewEpoch := True;
    Shuffle;
    for j := 0 to PatternCount - 1 do
    begin
      LoadPatternsInput(FRandomOrder[j]);
      LoadPatternsOutput(FRandomOrder[j]);
      Propagate;
      xQuadError := QuadError;
      // перевірка - чи розпізнаний приклад з навчальної множини
      if xQuadError < IdentError then
        Inc(FRecognizedTeachCount);
      FTeachError := FTeachError + xQuadError;
      // максимальна помилка на навчальній множині
      if xNewEpoch then
      begin
        FMaxTeachResidual := xQuadError;
        xNewEpoch := False;
      end
      else
        if MaxTeachResidual < xQuadError then
          FMaxTeachResidual := xQuadError;
      CalcLocalError;
      AdjustWeights;
    end;
    // середня помилка на навчальній множині
    FMidTeachResidual := TeachError/PatternCount;
    // перевірка мережі на узагальнення
    if TestSetPatternCount > 0 then
      CheckTestSet;
    DoOnEpochPassed;
    if StopTeach then
    begin
      TeachStopped := True;
      Exit;
    end;
    Inc(FEpochCurrent);
  end;
  DoOnAfterTeach;
end;

procedure TNeuralNetBP.SetPatternCount(const Value: integer);
begin
  FPatternCount := Value;
  inherited;
end;

procedure TNeuralNetBP.SetDefaultProperties;
begin
  // параметри встановлювані за замовчуванням
  Alpha := DefaultAlpha;
  ContinueTeach := False;
  Epoch := True;
```

```
  EpochCount := DefaultEpochCount;
  Momentum := DefaultMomentum;
  TeachRate := DefaultTeachRate;
  ResizeInputDim;
  ResizeOutputDim;
end;

{ TNeuralNetExtended }

constructor TNeuralNetExtended.Create(AOwner: TComponent);
begin
  inherited;
  SetDefaultProperties;
end;

destructor TNeuralNetExtended.Destroy;
var
  i: integer;
begin
  if Assigned(FNnwFile) then
    FNnwFile.Free;
  FNeuroDataSource.Free;
  for i := 0 to FAvailableFieldsCount - 1 do
    FFields[i].Free;
  inherited;
end;

function TNeuralNetExtended.GetFields(Index: integer): TNeuroField;
begin
  Result := FFields[Index];
end;

function TNeuralNetExtended.GetInputFieldCount: integer;
var
  i: integer;
begin
  Result := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if Fields[i].KindName = fdInput then
      Inc(Result);
end;

function TNeuralNetExtended.GetOutputFieldCount: integer;
var
  i: integer;
begin
  Result := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if Fields[i].KindName = fdOutput then
      Inc(Result);
end;

function TNeuralNetExtended.GetOutput(Index: integer): double;
var
  xTmp: double;
begin
  with Fields[RealOutputIndex[Index]] do
    case NormTypeName of
      nrmAuto: begin
                 xTmp := -ln(1/LayersBP[LayerCount - 1].NeuronsBP[Index].Output -
1);
                 LayersBP[LayerCount - 1].NeuronsBP[Index].Output := xTmp *
Dispersion + ValueMid;
               end;
      nrmLinear:  Result := (LayersBP[LayerCount - 1].NeuronsBP[Index].Output +
1)*(ValueMax - ValueMin)/2 + ValueMin;
      nrmLinearOut: Result := LayersBP[LayerCount -
1].NeuronsBP[Index].Output*(ValueMax - ValueMin) + ValueMin;
```

```
      nrmSigmoid:  Result := - Ln(1/LayersBP[LayerCount -
1].NeuronsBP[Index].Output - 1)/Alpha;
    end;
end;

function TNeuralNetExtended.GetRealInputIndex(Index: integer): integer;
begin
  Result := FRealInputIndex[Index];
end;

function TNeuralNetExtended.GetRealOutputIndex(Index: integer): integer;
begin
  Result := FRealOutputIndex[Index];
end;

procedure TNeuralNetExtended.ComputeUnPrepData(AVector: TVectorFloat);
var
  i: integer;
  xTmp: double;
begin
  if InputNeuronCount <> High(AVector)+ 1 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  for i := Low(AVector) to High(AVector) do
  with FFields[RealInputIndex[i]] do
    case NormTypeName of
      nrmAuto: begin
                 xTmp := (LayersBP[SensorLayer].NeuronsBP[i].Output -
ValueMid)/Dispersion;
                 LayersBP[SensorLayer].NeuronsBP[i].Output := 1/(1 + exp(-
xTmp));
               end;
      nrmLinear: LayersBP[SensorLayer].NeuronsBP[i].Output := 2*(AVector[i] -
ValueMin)/(ValueMax - ValueMin) - 1;
      nrmLinearOut: LayersBP[SensorLayer].NeuronsBP[i].Output := (AVector[i] -
ValueMin)/(ValueMax - ValueMin);
      nrmSigmoid: LayersBP[SensorLayer].NeuronsBP[i].Output := 1/( 1 + exp(-
Alpha * AVector[i]));
    end;
  Propagate;
end;

procedure TNeuralNetExtended.DoOnBeforeTeach;
begin
  if InputNeuronCount <> InputFieldCount then
    raise ENeuronNotEqualFieldError.Create(SInFieldCount);
  if OutputNeuronCount <> OutputFieldCount then
    raise ENeuronNotEqualFieldError.Create(SOutFieldCount);
  if InputNeuronCount < 0 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  if OutputNeuronCount < 0 then
    raise EInOutDimensionError.Create(SOutNeuronCount);
  if (not FMaxTeachError) and (not FMaxTestError) and
     (not FMidTeachError) and (not FMidTestError) and (not FEpoch) then
    raise EBPStopCondition.Create(SBPStopCondition);
  inherited DoOnBeforeTeach;
end;

procedure TNeuralNetExtended.DoOnEpochPassed;
begin
  if MaxTeachError and (MaxTeachResidual < MaxTeachErrorValue) then
    StopTeach := True;
  if MidTeachError and (MidTeachResidual < MidTeachErrorValue) then
    StopTeach := True;
  if MaxTestError and (MaxTestResidual < MaxTestErrorValue) then
    StopTeach := True;
  if MidTestError and (MidTestResidual < MidTestErrorValue) then
    StopTeach := True;
  if TeachIdent and (Round((FRecognizedTeachCount * 100)/PatternCount) <
TeachIdentCount) then
```

```
    StopTeach := True;
  if TestIdent and (Round((FRecognizedTestCount * 100)/TestSetPatternCount) <
TestIdentCount) then
    StopTeach := True;
  inherited DoOnEpochPassed;
end;

procedure TNeuralNetExtended.LoadDataFrom;
var
  xTempStream: TFileStream;
  i, j: integer;
  xFieldCount: integer;
  xArray: TVectorFloat;
  xPatternsList: TStringList;
begin
  // створюється потік
  xTempStream := TFileStream.Create(FSourceFileName, fmOpenRead);
  // створюється список
  xPatternsList := TStringList.Create;
  xPatternsList.LoadFromStream(xTempStream);
  try
    if SettingsLoaded then
    begin
      xFieldCount := FNeuroDataSource.FieldCount(xPatternsList.Strings[0]);
      if AvailableFieldsCount <> xFieldCount then
        if MessageDlg('Кількість полів у файлі даних не відповідає значенню
AvailableFieldsCount'+ #13 + 'Установити нове значення AvailableFieldsCount = '+
IntToStr(xFieldCount),
                    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
          AvailableFieldsCount := xFieldCount;
    end
    else
      AvailableFieldsCount :=
FNeuroDataSource.FieldCount(xPatternsList.Strings[0]);
    FNeuroDataSource.ExtractHeaders(FFields, xPatternsList.Strings[0]);
    // установлюється розмірність часового масиву
    SetLength(xArray, FAvailableFieldsCount);
    // установлюється розмірність масиву даних
    if FUseForTeach = 100 then
      PatternCount := xPatternsList.Count - 1
    else
    begin
      PatternCount := Round((xPatternsList.Count - 1) * FUseForTeach / 100);
      TestSetPatternCount := xPatternsList.Count - PatternCount - 1;
    end;
    for i := 0 to FAvailableFieldsCount - 1 do
      FFields[i].DataInCount := xPatternsList.Count - 1;
    for j := 0 to xPatternsList.Count - 2 do
    begin
      FNeuroDataSource.ExtractValues(xArray, xPatternsList.Strings[j + 1]);
      for i := 0 to FAvailableFieldsCount - 1 do
        FFields[i].DataIn[j] := xArray[i]
    end;
  finally
    xTempStream.Free;
    xPatternsList.Free;
    SetLength(xArray, 0);
    xArray := nil;
  end;
end;

procedure TNeuralNetExtended.LoadPhase1;
begin
  FSourceFileName := FNnwFile.ReadString('Phase1', 'LearnSampleFileName', '');
  FNeuroDataSource.Name := FSourceFileName;
end;

procedure TNeuralNetExtended.LoadPhase2;
var
```

```
  i: integer;
begin
  AvailableFieldsCount := FNnwFile.ReadInteger('Phase2', 'AvailableFieldsCount',
1);
  for i := 0 to AvailableFieldsCount - 1 do
  with FFields[i] do
  begin
    Name := FNnwFile.ReadString('Phase2', 'FieldName_'+IntToStr(i), '');
    Kind := FNnwFile.ReadInteger('Phase2', 'FieldType_'+IntToStr(i), 0);
    NormType := FNnwFile.ReadInteger('Phase2', 'NormType_'+IntToStr(i), 0);
    ValueMax := FNnwFile.ReadFloat('Phase2', 'Max_'+IntToStr(i), 0);
    ValueMin := FNnwFile.ReadFloat('Phase2', 'Min_'+IntToStr(i), 0);
    ValueMid := FNnwFile.ReadFloat('Phase2', 'Mid_'+IntToStr(i), 0);
    Dispersion := FNnwFile.ReadFloat('Phase2', 'Disp_'+IntToStr(i), 0);
    Alpha := FNnwFile.ReadFloat('Phase2', 'Alpha_'+IntToStr(i), 0);
    Ind := FNnwFile.ReadBool('Phase2', 'Ind_'+IntToStr(i), False);
  end;
  SettingsLoaded := True;
end;

procedure TNeuralNetExtended.LoadPhase4;
begin
  UseForTeach := FNnwFile.ReadInteger('Phase4', 'UseForTeach',
DefaultUseForTeach);
  IdentError:= FNnwFile.ReadFloat('Phase4', 'IdentErr', DefaultErrorValue);
  TestAsValid := FNnwFile.ReadBool('Phase4', 'TestAsValid', False);
  Epoch:= FNnwFile.ReadBool('Phase4', 'Epoch', False);
  EpochCount:= FNnwFile.ReadInteger('Phase4', 'Epoch', DefaultEpochCount);
  MaxTeachError:= FNnwFile.ReadBool('Phase4', 'MaxTeachErr', False);
  MaxTeachErrorValue:= FNnwFile.ReadFloat('Phase4', 'MaxTeachErr',
DefaultErrorValue);
  MidTeachError:= FNnwFile.ReadBool('Phase4', 'MidTeachErr', False);
  MidTeachErrorValue:= FNnwFile.ReadFloat('Phase4', 'MidTeachErr',
DefaultErrorValue);
  TeachIdent:= FNnwFile.ReadBool('Phase4', 'TeachIdent', False);
  TeachIdentCount:= FNnwFile.ReadInteger('Phase4', 'TeachIdent',
DefaultTeachIdentCount);
  MaxTestError:= FNnwFile.ReadBool('Phase4', 'MaxTestErr', False);
  MaxTestErrorValue:= FNnwFile.ReadFloat('Phase4', 'MaxTestErr',
DefaultErrorValue);
  MidTestError:= FNnwFile.ReadBool('Phase4', 'MidTestErr', False);
  MidTestErrorValue:= FNnwFile.ReadFloat('Phase4', 'MidTestErr',
DefaultErrorValue);
  TestIdent:= FNnwFile.ReadBool('Phase4', 'TestIdent', False);
  TestIdentCount:= FNnwFile.ReadInteger('Phase4', 'TestIdent',
DefaultTestIdentCount);
end;

procedure TNeuralNetExtended.LoadNetwork;
var
  i,j,k: integer;
  xLayerCount: integer;
begin
  // знищується поточна конфігурація нейромережі
  ResetLayers;
  Alpha := FNnwFile.ReadFloat('Network', 'Alpha', DefaultAlpha);
  Momentum := FNnwFile.ReadFloat('Network', 'Miu', DefaultMomentum);
  TeachRate := FNnwFile.ReadFloat('Network', 'TeachSpeed', DefaultTeachRate);
  EpochCount := FNnwFile.ReadInteger('Network', 'Epoch', DefaultEpochCount);
  xLayerCount := FNnwFile.ReadInteger('Network','CountLayers',
DefaultLayerCount);
  // задається кількість нейронів у шарах
  AutoInit := False;
  for i := 0 to xLayerCount - 1 do
    AddLayer(FNnwFile.ReadInteger('Network','Layer_'+IntToStr(i),
DefaultNeuronCount));
  AutoInit := True;
  // ініціалізація нової конфігурації нейромережі
  Init;
```

```
  // завантаження вагових коефіцієнтів і зсуву
  for i:= 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
    begin
      for k := 0 to LayersBP[ i-1].NeuronCount - 1 do
        LayersBP[i].NeuronsBP[j].Weights[k] := FNnwFile.ReadFloat('Network',
                              'W_'+IntToStr( i-
1)+'_'+IntToStr(k)+'_'+IntToStr(j), 0);
      LayersBP[i].NeuronsBP[j].Weights[LayersBP[ i-1].NeuronCount] :=
FNnwFile.ReadFloat('Network',
       'WT_'+IntToStr( i-1)+'_'+IntToStr(j), 0);
    end;
end;

procedure TNeuralNetExtended.SavePhase1;
begin
  FNnwFile.WriteString('Phase1', 'LearnSampleFileName', FNeuroDataSource.Name);
end;

procedure TNeuralNetExtended.SavePhase2;
var
  i: integer;
begin
  FNnwFile.WriteInteger('Phase2', 'AvailableFieldsCount',
FAvailableFieldsCount);
  for i := 0 to AvailableFieldsCount - 1 do
  with Fields[i] do
  begin
    FNnwFile.WriteString('Phase2', 'FieldName_'+IntToStr(i), Name);
    FNnwFile.WriteInteger('Phase2', 'FieldType_'+IntToStr(i), Kind);
    FNnwFile.WriteInteger('Phase2', 'NormType_'+IntToStr(i), NormType);
    FNnwFile.WriteFloat('Phase2', 'Max_'+IntToStr(i), ValueMax);
    FNnwFile.WriteFloat('Phase2', 'Min_'+IntToStr(i), ValueMin);
    FNnwFile.WriteFloat('Phase2', 'Mid_'+IntToStr(i), ValueMid);
    FNnwFile.WriteFloat('Phase2', 'Disp_'+IntToStr(i), Dispersion);
    FNnwFile.WriteFloat('Phase2', 'Alpha_'+IntToStr(i), Alpha);
    FNnwFile.WriteBool('Phase2', 'Ind_'+IntToStr(i), Ind);
  end;
end;

procedure TNeuralNetExtended.SavePhase4;
begin
  FNnwFile.WriteBool('Phase4', 'Epoch', Epoch);
  FNnwFile.WriteInteger('Phase4', 'Epoch', EpochCount);
  FNnwFile.WriteFloat('Phase4', 'IdentErr', IdentError);
  FNnwFile.WriteBool('Phase4', 'MaxTeachErr', MaxTeachError);
  FNnwFile.WriteFloat('Phase4', 'MaxTeachErr', MaxTeachErrorValue);
  FNnwFile.WriteBool('Phase4', 'MaxTestErr', MaxTestError);
  FNnwFile.WriteFloat('Phase4', 'MaxTestErr', MaxTestErrorValue);
  FNnwFile.WriteBool('Phase4', 'MidTeachErr', MidTeachError);
  FNnwFile.WriteFloat('Phase4', 'MidTeachErr', MidTeachErrorValue);
  FNnwFile.WriteBool('Phase4', 'MidTestErr', MidTestError);
  FNnwFile.WriteFloat('Phase4', 'MidTestErr', MidTestErrorValue);
  FNnwFile.WriteFloat('Phase4', 'Miu', Momentum);
  FNnwFile.WriteBool('Phase4', 'TeachIdent', TeachIdent);
  FNnwFile.WriteFloat('Phase4', 'TeachSpeed', TeachRate);
  FNnwFile.WriteInteger('Phase4', 'TeachIdent', TeachIdentCount);
  FNnwFile.WriteBool('Phase4', 'TestAsValid', TestAsValid);
  FNnwFile.WriteBool('Phase4', 'TestIdent', TestIdent);
  FNnwFile.WriteInteger('Phase4', 'TestIdent', TestIdentCount);
  FNnwFile.WriteInteger('Phase4', 'UseForTeach', UseForTeach);
end;

procedure TNeuralNetExtended.SaveNetwork;
var
  i, j, k: integer;
begin
  FNnwFile.WriteFloat('Network', 'TeachSpeed', TeachRate);
  FNnwFile.WriteFloat('Network', 'Miu', Momentum);
```

```
  FNnwFile.WriteFloat('Network', 'Alpha', Alpha);
  FNnwFile.WriteInteger('Network', 'Epoch', EpochCount);
  FNnwFile.WriteInteger('Network','CountLayers', LayerCount);
  // задається кількість нейронів у шарах
  for i := 0 to LayerCount - 1 do
    FNnwFile.WriteInteger('Network','Layer_'+IntToStr(i),
StrToInt(NeuronsInLayer[i]));
  // завантаження вагових коефіцієнтів і зсуву
  for i:= 1 to LayerCount - 1 do
    for j := 0 to StrToInt(NeuronsInLayer[i]) - 1 do
    begin
      for k := 0 to StrToInt(NeuronsInLayer[ i-1]) do
        FNnwFile.WriteFloat('Network','W_'+IntToStr( i-1)+'_'+IntToStr(k)+
                            '_'+IntToStr(j),
LayersBP[i].NeuronsBP[j].Weights[k]);
      FNnwFile.WriteFloat('Network','WT_'+IntToStr( i-1)+'_'+IntToStr(j),
         LayersBP[i].NeuronsBP[j].Weights[StrToInt(NeuronsInLayer[j])]);
   end;
end;

procedure TNeuralNetExtended.NormalizeData;
var
  i: integer;
begin
  // нормалізація вхідних і вихідних значень
  for i := 0 to FAvailableFieldsCount - 1 do
  begin
    FFields[i].FindMinMax;
    FFields[i].Normalize;
  end;
end;

procedure TNeuralNetExtended.Train;
var
  i, j, k: integer;
begin
  if FUseForTeach = 100 then
  begin
    PatternCount := FFields[0].DataInCount;
    TestSetPatternCount := 0;
  end
  else
  begin
    PatternCount := Round((FFields[0].DataInCount - 1) * FUseForTeach / 100);
    TestSetPatternCount := FFields[0].DataInCount - PatternCount;
  end;
  if not TeachStopped then
    NormalizeData;
  // формування вхідних значень навчальної множини
  RealOutputIndexCount := OutputFieldCount;
  RealInputIndexCount := InputFieldCount;
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdInput then
    begin
      for j := 0 to PatternCount - 1 do
        FPatternsInput[j, k] := FFields[i].DataIn[j];
      // запам'ятовує індекс поля
      RealInputIndex[k] := i;
      Inc(k);
    end;
  // формування вихідних значень навчальної множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := 0 to PatternCount - 1 do
        FPatternsOutput[j, k] := FFields[i].DataIn[j];
      // запам'ятовує індекс поля
```

```
      RealOutputIndex[k] := i;
      Inc(k);
    end;
  // формування вхідних значень тестової множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdInput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatterns[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
  // формування вихідних значень тестової множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatternsOut[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
  // навчання або донавчання мережі
  TeachOffLine;
end;

procedure TNeuralNetExtended.SetAvailableFieldsCount(Value : integer);
var
  i: integer;
begin
  FAvailableFieldsCount := Value;
  // встановлюється кількість полів
  SetLength(FFields, Value);
  for i := 0 to FAvailableFieldsCount - 1 do
    FFields[i] := TNeuroField.Create;
end;

procedure TNeuralNetExtended.SetFields(Index: integer; Value: TNeuroField);
begin
  try
    FFields[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SFieldIndexRange, [Index])
  end;
end;

procedure TNeuralNetExtended.SetDefaultProperties;
begin
  // параметри встановлювані за замовчуванням
  Epoch := False;
  IdentError:= DefaultErrorValue;
  MaxTeachError := False;
  MaxTeachErrorValue := DefaultErrorValue;
  MaxTestError:= False;
  MaxTestErrorValue:= DefaultErrorValue;
  MidTestError:= False;
  MidTestErrorValue:= DefaultErrorValue;
  MidTeachError := False;
  MidTeachErrorValue :=  DefaultErrorValue;
  SettingsLoaded := False;
  TeachIdent := False;
  TeachIdentCount:= DefaultTeachIdentCount;
  TestAsValid := False;
  TestIdent:= False;
  TestIdentCount:= DefaultTestIdentCount;
  UseForTeach := DefaultUseForTeach;
end;

procedure TNeuralNetExtended.SetFileName(Value: TFilename);
```

```
begin
  if Assigned(FNnwFile) then
    FNnwFile.Free;
  try
    FNnwFile := TIniFile.Create(Value);
    FFileName := Value;
  except
    on E: EInOutError do
      raise E.CreateFmt(SWrongFileName, [Value]);
  end;
  FNeuroDataSource := TNeuroDataSource.Create;
  LoadPhase1;
  LoadPhase2;
  LoadPhase4;
  LoadNetwork;
  LoadDataFrom;
end;

procedure TNeuralNetExtended.SetTeachIdentCount(const Value: integer);
begin
  if (Value <= 0) or (Value > 100) then
    FTeachIdentCount := DefaultTeachIdentCount
  else
    FTeachIdentCount := Value;
end;

procedure TNeuralNetExtended.SetUseForTeach(const Value: integer);
begin
  if (Value <= 0) or (Value > 100) then
    FUseForTeach := DefaultUseForTeach
  else
    FUseForTeach := Value;
end;

procedure TNeuralNetExtended.SetRealOutputIndex(Index: integer; const Value:
integer);
begin
  FRealOutputIndex[Index] := Value;
end;

procedure TNeuralNetExtended.SetRealOutputIndexCount(const Value: integer);
begin
  SetLength(FRealOutputIndex, Value)
end;
procedure TNeuralNetExtended.SetRealInputIndex(Index: integer; const Value:
integer);
begin
  FRealInputIndex[Index] := Value;
end;
procedure TNeuralNetExtended.SetRealInputIndexCount(const Value: integer);
begin
  SetLength(FRealInputIndex, Value)
end;
procedure Register;
begin
  RegisterComponents('Neural_network_', [TNeuralNetHopf, TNeuralNetBP,
TNeuralNetExtended]);
end;
end.
```

Pumpdata.pas – формування бази знань

```
unit PumpData;

interface

uses
  SysUtils, IniFiles, Classes, Neural_network_Types;

type

  EFieldNormError = class(Exception);
  EFieldKindError = class(Exception);

  TNeuroField = class;
  TNeuroFields = array of TNeuroField;

  TNeuroField = class(TObject)
  private
    FAlpha: double;
    FDataIn: TVectorFloat;
    FDispersion: double;
    FInd: boolean;
    FKind: byte;
    FName: string;
    FNormType: byte;
    FValueMax: double;
    FValueMid: double;
    FValueMin: double;
    function GetDataIn(Index: integer): double;
    function GetKindName: TNeuroFieldType;
    function GetNormTypeName: TNormalize;
    function GetDataInCount: integer;
    procedure SetDataIn(Index: integer; Value: double);
    procedure SetKind(Value: byte);
    procedure SetNormType(Value: byte);
    procedure SetDataInCount(Value: integer);
  public
    procedure FindMinMax;
    procedure CalcMid;
    procedure CalcDispersion;
    procedure Normalize;
    procedure DeNormalize;
    property Alpha: double read FAlpha write FAlpha;
    property DataIn[Index: integer]: double read GetDataIn write SetDataIn;
    property DataInCount: integer read GetDataInCount write SetDataInCount;
    property Dispersion: double read FDispersion write FDispersion;
    property Ind: boolean read FInd write FInd;
    property Kind: byte read FKind write SetKind;
    property KindName: TNeuroFieldType read GetKindName;
    property Name: string read FName write FName;
    property NormType: byte read FNormType write SetNormType;
    property NormTypeName: TNormalize read GetNormTypeName;
    property ValueMax: double read FValueMax write FValueMax;
    property ValueMin: double read FValueMin write FValueMin;
    property ValueMid: double read FValueMid write FValueMid;
  end;

  TNeuroDataSource = class(TObject)
  private
    FName: TFileName;
    function IsHeaderChar(AValue: char): boolean;
  public
    function FieldCount(AHeader: string): integer;
    procedure ExtractHeaders(const AFields: TNeuroFields; AHeader: string);
    procedure ExtractValues(const AVector:TVectorFloat; AHeader: string);
    property Name: TFileName read FName write FName;
  end;
```

```
implementation

{ Клас TNeuroField }

function TNeuroField.GetDataIn(Index: integer): double;
begin
  Result := FDataIn[Index];
end;

function TNeuroField.GetDataInCount: integer;
begin
  Result := High(FDataIn) + 1;
end;

function TNeuroField.GetKindName: TNeuroFieldType;
begin
  case FKind of
    0 : Result := fdInput;
    1 : Result := fdOutput;
    2 : Result := fdNone;
  end;
end;

function TNeuroField.GetNormTypeName: TNormalize;
begin
  case FNormType of
    0 : if KindName = fdInput then
          Result := nrmLinear
        else if KindName = fdOutput then
          Result := nrmLinearOut;
    1 : Result := nrmSigmoid;
    2 : Result := nrmAuto;
    3 : Result := nrmNone;
  end;
end;

procedure TNeuroField.CalcMid;
var
  i: integer;
begin
  FValueMid := 0;
  for i := Low(FDataIn) to High(FDataIn) do
    FValueMid := FValueMid + FDataIn[i];
  FValueMid := FValueMid/(High(FDataIn) + 1);
end;

procedure TNeuroField.CalcDispersion;
var
  i: integer;
begin
  if High(FDataIn) > 1 then
  begin
    FDispersion := 0;
    for i := Low(FDataIn) to High(FDataIn) do
      FDispersion := FDispersion + sqr(FDataIn[i] - ValueMid);
    FDispersion := sqrt(FDispersion/High(FDataIn));
  end
  else
    FDispersion := 0;
end;

(*procedure TNeuroField.DeNormalize;
var
  i: integer;
  xTmp: double;
begin
  case NormTypeName of
    nrmLinear: for i := Low(FDataIn) to High(FDataIn) do
```

```
                FDataIn[i] := (FDataIn[i] + 1)*(FValueMax - FValueMin)/2 +
FValueMin;
    nrmLinearOut: for i := Low(FDataIn) to High(FDataIn) do
                FDataIn[i] := FDataIn[i]*(FValueMax - FValueMin) + FValueMin;
    nrmSigmoid: for i := Low(FDataIn) to High(FDataIn) do
                FDataIn[i] := - Ln(1/FDataIn[i] - 1)/Alpha;
  end;
end;*)

procedure TNeuroField.FindMinMax;
var
  i: integer;
begin
  FValueMax:= FDataIn[0];
  FValueMin:= FDataIn[0];
  for i := 1 to High(FDataIn) do
  begin
    if FValueMin > FDataIn[i] then
      FValueMin := FDataIn[i];
    if FValueMax < FDataIn[i] then
      FValueMax := FDataIn[i]
  end;
end;

procedure TNeuroField.Normalize;
var
  i: integer;
  xTmp: double;
begin
  case NormTypeName of
    nrmAuto:  begin
                CalcMid;
                CalcDispersion;
                for i := Low(FDataIn) to High(FDataIn) do
                begin
                  xTmp := (FDataIn[i] - FValueMid)/FDispersion;
                  FDataIn[i] := 1/(1 + exp(-xTmp));
                end;
              end;
    nrmLinear: for i := Low(FDataIn) to High(FDataIn) do
               FDataIn[i] := 2*(FDataIn[i] - FValueMin)/(FValueMax - FValueMin)-
1;
    nrmLinearOut: for i := Low(FDataIn) to High(FDataIn) do
               FDataIn[i] := (FDataIn[i] - FValueMin)/(FValueMax - FValueMin);
    nrmSigmoid: for i := Low(FDataIn) to High(FDataIn) do
               FDataIn[i] := 1/( 1 + exp(-Alpha * FDataIn[i]));
  end;
end;

procedure TNeuroField.SetNormType(Value: byte);
begin
  if (Value < 0) or (Value > 3) then
    raise EFieldNormError.CreateFmt(SFieldNorm, [Value])
  else
    FNormType := Value;
end;

procedure TNeuroField.SetKind(Value: byte);
begin
  if (Value < 0) or (Value > 2) then
    raise Exception.CreateFmt(SFieldKind, [Value])
  else
    FKind := Value;
end;

procedure TNeuroField.SetDataIn(Index: integer; Value: double);
begin
  FDataIn[Index] := Value;
end;
```

```
procedure TNeuroField.SetDataInCount(Value: integer);
begin
  SetLength(FDataIn, Value)
end;

{ Клас TNeuroDataSource }

function TNeuroDataSource.IsHeaderChar(AValue: char): boolean;
begin
  if (AValue in Letters) or (AValue in Capitals) or (AValue in DigitChars) then
    Result := True
  else
    Result := False;
end;

procedure TNeuroDataSource.ExtractValues(const AVector:TVectorFloat; AHeader:
string);
var
  s: string;
  i, xCurPos: integer;
begin
  i := 0;
  AHeader := Trim(AHeader);
  xCurPos := Pos(SpaceChar, AHeader);
  try
    while xCurPos > 0 do
    begin
      s := Copy(AHeader, 1, xCurPos - 1);
      AVector[i] := StrToFloat(s);
      Inc(i);
      Delete(AHeader, 1, xCurPos - 1);
      AHeader := Trim(AHeader);
      xCurPos := Pos(SpaceChar, AHeader);
    end;
    s := AHeader;
    AVector[i] := StrToFloat(s);
  except
    on EConvertError do
      EConvertError.CreateFmt(SCannotBeNumber, [s])
  end;
end;

procedure TNeuroDataSource.ExtractHeaders(const AFields: TNeuroFields; AHeader:
string);
var
  s: string;
  xFieldCount, j, xCurPos: integer;
begin
  { виділяє заголовки з файлу }
  xFieldCount := 0;
  AHeader := Trim(AHeader);
  xCurPos := Pos(SpaceChar, AHeader);
  while xCurPos > 0 do
  begin
    s := Copy(AHeader, 1, xCurPos - 1);
    AFields[xFieldCount].FName := '';
    for j := 1 to Length(s) do
      if isHeaderChar(s[j]) then
        AFields[xFieldCount].FName := AFields[xFieldCount].FName + s[j];
    Inc(xFieldCount);
    Delete(AHeader, 1, xCurPos - 1);
    AHeader := Trim(AHeader);
    xCurPos := Pos(SpaceChar, AHeader);
  end;
  AFields[xFieldCount].FName := '';
  for j := 1 to Length(AHeader) do
    if isHeaderChar(AHeader[j]) then
      AFields[xFieldCount].FName := AFields[xFieldCount].FName + AHeader[j];
```

```
  { повертає кількість полів }
end;

function TNeuroDataSource.FieldCount(AHeader: string): integer;
var
  xFieldCount, xCurPos: integer;
begin
  { виділяє заголовки з файлу }
  xFieldCount := 0;
  AHeader := Trim(AHeader);
  xCurPos := Pos(SpaceChar, AHeader);
  while xCurPos > 0 do
  begin
    Inc(xFieldCount);
    Delete(AHeader, 1, xCurPos - 1);
    AHeader := Trim(AHeader);
    xCurPos := Pos(SpaceChar, AHeader);
  end;
  { повертає кількість полів }
  Result := xFieldCount + 1;
end;


end.
```

NeuralNetExtend.pas – навчання мережі

```
unit NeuralNetExtend;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, Neural_network_Comp, ExtCtrls, StdCtrls, Spin, Grids,
Neural_network_Types,
  IniFiles;

const
  FormCaption = 'Навчання мережі';

type
  TfrmNeuralNetExtend = class(TForm)
    PageControl: TPageControl;
    pnlNavigation: TPanel;
    Tab1: TTabSheet;
    btnBack: TButton;
    rgrFileType: TRadioGroup;
    btnNext: TButton;
    btnCancel: TButton;
    Tab2: TTabSheet;
    lblFileName: TLabel;
    btnOpenFile: TButton;
    edtFileName: TEdit;
    OpenDialog: TOpenDialog;
    Tab3: TTabSheet;
    ltbFieldName: TListBox;
    Label2: TLabel;
    rdgFieldType: TRadioGroup;
    rdgNormType: TRadioGroup;
    GroupBox1: TGroupBox;
    Label3: TLabel;
    edtMin: TEdit;
    Label4: TLabel;
    edtMax: TEdit;
    Label5: TLabel;
    edt: TEdit;
    Tab4: TTabSheet;
    speLayers: TSpinEdit;
    Label6: TLabel;
    stgNeuronsInLayer: TStringGrid;
    Label7: TLabel;
    Tab5: TTabSheet;
    Label8: TLabel;
    tbrAlpha: TTrackBar;
    sttAlpha: TStaticText;
    Label9: TLabel;
    edtMomentum: TEdit;
    Label10: TLabel;
    edtTeachRate: TEdit;
    Tab6: TTabSheet;
    Label11: TLabel;
    Label12: TLabel;
    btnContinueTeach: TButton;
    sttMaxTeachError: TStaticText;
    sttEpochCount: TStaticText;
    GroupBox2: TGroupBox;
    Label13: TLabel;
    edtIdentError: TEdit;
    speEpochCount: TSpinEdit;
    cbxEpoch: TCheckBox;
    cbxMaxTeachError: TCheckBox;
    edtMaxTeachErrorValue: TEdit;
```

```
    cbxMidTeachError: TCheckBox;
    edtMidTeachErrorValue: TEdit;
    cbxTeachIdent: TCheckBox;
    speTeachIdentValue: TSpinEdit;
    btnBeginTeach: TButton;
    cbxMaxTestError: TCheckBox;
    cbxMidTestError: TCheckBox;
    cbxTestIdent: TCheckBox;
    edtMaxTestErrorValue: TEdit;
    edtMidTestErrorValue: TEdit;
    speTestIdentValue: TSpinEdit;
    Tab7: TTabSheet;
    Label14: TLabel;
    stgInput: TStringGrid;
    Label15: TLabel;
    stgOutput: TStringGrid;
    btnCompute: TButton;
    Memo1: TMemo;
    Label16: TLabel;
    Label17: TLabel;
    Memo2: TMemo;
    Bevel1: TBevel;
    Memo3: TMemo;
    Label1: TLabel;
    sttMaxTestError: TStaticText;
    Label18: TLabel;
    sttMidTeachError: TStaticText;
    Label19: TLabel;
    sttMidTestError: TStaticText;
    SaveDialog: TSaveDialog;
    edtUseForTeach: TEdit;
    Label20: TLabel;
    btnSave: TButton;
    procedure btnCancelClick(Sender: TObject);
    procedure btnNextClick(Sender: TObject);
    procedure btnBackClick(Sender: TObject);
    procedure btnOpenFileClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure ltbFieldNameClick(Sender: TObject);
    procedure rdgFieldTypeClick(Sender: TObject);
    procedure rdgNormTypeClick(Sender: TObject);
    procedure edtAChange(Sender: TObject);
    procedure tbrAlphaChange(Sender: TObject);
    procedure edtMomentumChange(Sender: TObject);
    procedure edtTeachRateChange(Sender: TObject);
    procedure NeuralNetExtendedEpochPassed(Sender: TObject);
    procedure btnContinueTeachClick(Sender: TObject);
    procedure edtIdentErrorChange(Sender: TObject);
    procedure cbxEpochClick(Sender: TObject);
    procedure speEpochCountChange(Sender: TObject);
    procedure cbxMaxTeachErrorClick(Sender: TObject);
    procedure edtMaxTeachErrorValueChange(Sender: TObject);
    procedure cbxMidTeachErrorClick(Sender: TObject);
    procedure edtMidTeachErrorValueChange(Sender: TObject);
    procedure cbxTeachIdentClick(Sender: TObject);
    procedure speTeachIdentValueChange(Sender: TObject);
    procedure cbxMaxTestErrorClick(Sender: TObject);
    procedure edtMaxTestErrorValueChange(Sender: TObject);
    procedure cbxMidTestErrorClick(Sender: TObject);
    procedure edtMidTestErrorValueChange(Sender: TObject);
    procedure cbxTestIdentClick(Sender: TObject);
    procedure speTestIdentValueChange(Sender: TObject);
    procedure NeuralNetExtendedAfterTeach(Sender: TObject);
    procedure btnBeginTeachClick(Sender: TObject);
    procedure btnComputeClick(Sender: TObject);
    procedure speLayersChange(Sender: TObject);
    procedure btnSaveClick(Sender: TObject);
    procedure edtUseForTeachChange(Sender: TObject);
  private
```

```
    { Private declarations }
    Teach: boolean;
    NotSaved: boolean;
    procedure ChangePage;
    procedure OpenFile;
    procedure LoadFile;
    procedure Tune;
    procedure CreateNet;
    procedure RunTeach;
    procedure TestNet;
  public
    { Public declarations }
  end;

var
  frmNeuralNetExtend: TfrmNeuralNetExtend;
implementation

{$R *.DFM}

// Запуск програми
procedure TfrmNeuralNetExtend.FormActivate(Sender: TObject);
begin
  Caption := FormCaption;
  PageControl.ActivePage := PageControl.Pages[0];
  Teach := false;
  NotSaved := false;
end;

// Вихід із програми
procedure TfrmNeuralNetExtend.btnCancelClick(Sender: TObject);
begin
  if NotSaved then
    if MessageDlg('Є незбережені дані. Зберегти?',mtConfirmation, [mbYes, mbNo],
0) = mrYes then
      btnSave.Click;
  Close;
end;

// Натискання кнопки "Вперед"
procedure TfrmNeuralNetExtend.btnNextClick(Sender: TObject);
begin
  with PageControl do
  begin
    ActivePage := FindNextPage(ActivePage, true, false);
    ChangePage;
    if ActivePage.PageIndex = PageCount - 1 then
      btnNext.Enabled := false
    else
      btnNext.Enabled := true;
    btnBack.Enabled := true;
  end;
end;

// Натискання кнопки "Назад"
procedure TfrmNeuralNetExtend.btnBackClick(Sender: TObject);
begin
  with PageControl do
  begin
    ActivePage := FindNextPage(ActivePage, false, false);
    if ActivePage.PageIndex = 0 then
      btnBack.Enabled := false
    else
      btnBack.Enabled := true;
    btnNext.Enabled := true;
  end;
end;

// Дія на зміну сторінки
```

```
procedure TfrmNeuralNetExtend.ChangePage;
begin
  case PageControl.ActivePage.PageIndex of
    1: OpenFile;
    2: LoadFile;
    3: Tune;
    4: CreateNet;
    6: TestNet;
  end;
end;

// Вибрати файл - джерело даних
procedure TfrmNeuralNetExtend.OpenFile;
begin
  if rgrFileType.ItemIndex = 0 then
  begin
    OpenDialog.Filter := 'NNW files (*.nnw)|*.nnw';
    lblFileName.Caption := 'Виберіть nnw-файл';
  end
  else
  begin
    OpenDialog.Filter := 'Text files (*.txt)|*.txt';
    lblFileName.Caption := 'Виберіть txt-файл';
  end;
end;

// Вибір файлу в діалоговому вікні
procedure TfrmNeuralNetExtend.btnOpenFileClick(Sender: TObject);
begin
  OpenDialog.Execute;
  Caption := FormCaption + ' - ' + ExtractFileName(OpenDialog.FileName);
  edtFileName.Text := OpenDialog.FileName;
end;

// Завантажити в компонент обраний файл
procedure TfrmNeuralNetExtend.LoadFile;
var
  i: integer;
begin
  try
    if rgrFileType.ItemIndex = 0 then
      NeuralNetExtended.FileName := edtFileName.Text   // nnw-файл
    else
    begin
      NeuralNetExtended.SourceFileName := edtFileName.Text; // текстовий файл
      NeuralNetExtended.LoadDataFrom; // завантажує дані з текстового файлу
      // конфігурація нейронної мережі за замовчуванням
      NeuralNetExtended.AddLayer(2);
      NeuralNetExtended.AddLayer(3);
      NeuralNetExtended.AddLayer(1);
    end;
  except
    raise Exception.Create('Помилка при відкритті файлу');
  end;
  NeuralNetExtended.Init;  // Ініціалізація мережі
  // Формування списку полів для StringList-A
  ltbFieldName.Clear;
  for i := 0 to NeuralNetExtended.AvailableFieldsCount - 1 do
    ltbFieldName.Items.Add(NeuralNetExtended.Fields[i].Name);
  ltbFieldName.ItemIndex := 0;
  ltbFieldNameClick(Self);
end;

// Настроювання параметрів мережі
procedure TfrmNeuralNetExtend.Tune;
var
  i: integer;
begin
  speLayers.Value := NeuralNetExtended.LayerCount - 2;
```

```
  stgNeuronsInLayer.RowCount := speLayers.Value + 1;
  stgNeuronsInLayer.Cells[0, 0] := '№ шаруючи';
  stgNeuronsInLayer.Cells[1, 0] := 'Нейронів';
  for i := 0 to speLayers.Value - 1 do
  begin
    stgNeuronsInLayer.Cells[0, i + 1] := IntToStr(i);
    stgNeuronsInLayer.Cells[1, i + 1] := IntToStr(NeuralNetExtended.Layers[i +
1].NeuronCount);
  end;

  tbrAlpha.Position := trunc(NeuralNetExtended.Alpha * 100);
  edtMomentum.Text := FloatToStr(NeuralNetExtended.Momentum);
  edtTeachRate.Text := FloatToStr(NeuralNetExtended.TeachRate);
  edtIdentError.Text := FloatToStr(NeuralNetExtended.IdentError);
  edtUseForTeach.Text := FloatToStr(NeuralNetExtended.UseForTeach);

  cbxEpoch.Checked := NeuralNetExtended.Epoch;
  speEpochCount.Text := IntToStr(NeuralNetExtended.EpochCount);

  cbxMaxTeachError.Checked := NeuralNetExtended.MaxTeachError;
  edtMaxTeachErrorValue.Text :=
FloatToStr(NeuralNetExtended.MaxTeachErrorValue);

  cbxMidTeachError.Checked := NeuralNetExtended.MidTeachError;
  edtMidTeachErrorValue.Text :=
FloatToStr(NeuralNetExtended.MidTeachErrorValue);

  cbxTeachIdent.Checked := NeuralNetExtended.TeachIdent;
  speTeachIdentValue.Value := NeuralNetExtended.TeachIdentCount;

  cbxMaxTestError.Checked := NeuralNetExtended.MaxTestError;
  edtMaxTestErrorValue.Text := FloatToStr(NeuralNetExtended.MaxTestErrorValue);

  cbxMidTestError.Checked := NeuralNetExtended.MidTestError;
  edtMidTestErrorValue.Text := FloatToStr(NeuralNetExtended.MidTestErrorValue);

  cbxTestIdent.Checked := NeuralNetExtended.TestIdent;
  speTestIdentValue.Value := NeuralNetExtended.TeachIdentCount;
end;

// Відображення інформації про обране поле (тип поля, нормалізація та інше)
procedure TfrmNeuralNetExtend.ltbFieldNameClick(Sender: TObject);
begin
  // Тип поля - вхідне, вихідне, не використовувати
  rdgFieldType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind;
  // Тип нормалізації
  rdgNormType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType;
  // Параметр нормалізації
  edt.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha);
  // Мінінум та максимум (для лінійної нормалізації)
  edtMin.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMin);
  edtMax.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMax);
end;

// Змінити тип поля
procedure TfrmNeuralNetExtend.rdgFieldTypeClick(Sender: TObject);
begin
  NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind :=
rdgFieldType.ItemIndex
end;

// Змінити тип нормалізації
procedure TfrmNeuralNetExtend.rdgNormTypeClick(Sender: TObject);
begin
```

```
  NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType :=
rdgNormType.ItemIndex
end;

// Змінити параметр нормалізації
procedure TfrmNeuralNetExtend.edtAChange(Sender: TObject);
begin
 NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha := StrToFloat(edt.Text);
end;

// Змінити параметр Alpha мережі
procedure TfrmNeuralNetExtend.tbrAlphaChange(Sender: TObject);
begin
  sttAlpha.Caption := FloatToStr(tbrAlpha.Position / 100);
end;

// Зміна кількості схованих шарів
procedure TfrmNeuralNetExtend.speLayersChange(Sender: TObject);
begin
  stgNeuronsInLayer.RowCount := speLayers.Value + 1;
end;

// Створення мережі обраної топології
procedure TfrmNeuralNetExtend.CreateNet;
var
  i: integer;
  xInput, xOutput: integer;
begin
  // Змінюється тільки кількість нейронів у схованих шарах,
  // Кількість нейронів у вхідному й вихідному шарі залежить від
  // типів полів
  with NeuralNetExtended do
  begin
    xInput := InputFieldCount;
    xOutput := OutputFieldCount;
    ResetLayers;
    AddLayer(xInput);
    for i := 0 to speLayers.Value - 1 do
      AddLayer(StrToInt(stgNeuronsInLayer.Cells[1, i + 1]));
    AddLayer(xOutput);
  end;
end;

// Змінити момент мережі
procedure TfrmNeuralNetExtend.edtMomentumChange(Sender: TObject);
begin
  NeuralNetExtended.Momentum := StrToFloat(edtMomentum.Text);
end;

// Змінити швидкість навчання мережі
procedure TfrmNeuralNetExtend.edtTeachRateChange(Sender: TObject);
begin
  NeuralNetExtended.TeachRate := StrToFloat(edtTeachRate.Text);
end;

// Дія по проходженню однієї епохи навчання
procedure TfrmNeuralNetExtend.NeuralNetExtendedEpochPassed(Sender: TObject);
begin
  sttMaxTestError.Caption := '';
  sttMidTestError.Caption := '';
  with NeuralNetExtended do
  begin
    sttMaxTeachError.Caption := FloatToStr(MaxTeachResidual); // Показати макс.
помилку на навчальній множині
    sttMidTeachError.Caption := FloatToStr(MidTeachResidual); // Показати серед.
помилку на навчальній множині
    if NeuralNetExtended.UseForTeach <> 100 then
    begin
```

```
      sttMaxTestError.Caption := FloatToStr(MaxTestResidual); // Показати макс.
помилку на тестовій множині
      sttMidTestError.Caption := FloatToStr(MidTestResidual); // Показати сред.
помилку на тестовій множині
    end;
    sttEpochCount.Caption := FloatToStr(EpochCurrent); // Показати номер
поточної епохи
  end;
  Application.ProcessMessages; // Дати можливість Windows перемалювати форму
end;

// Натискання кнопки "Продовжити навчання"
procedure TfrmNeuralNetExtend.btnContinueTeachClick(Sender: TObject);
begin
  NeuralNetExtended.ContinueTeach := true; // Скинути прапор - "Продовжити
навчання"
  RunTeach;  // Запуск
end;

// Натискання кнопки "Навчити"
procedure TfrmNeuralNetExtend.btnBeginTeachClick(Sender: TObject);
begin
  NeuralNetExtended.ContinueTeach := false; // Скинути прапор - "Почати навчання
знову"
  RunTeach;
end;

procedure TfrmNeuralNetExtend.edtIdentErrorChange(Sender: TObject);
begin
  NeuralNetExtended.IdentError := StrToFloat(edtIdentError.Text);
end;

procedure TfrmNeuralNetExtend.edtUseForTeachChange(Sender: TObject);
begin
  NeuralNetExtended.UseForTeach := StrToInt(edtUseForTeach.Text);
end;

procedure TfrmNeuralNetExtend.cbxEpochClick(Sender: TObject);
begin
  NeuralNetExtended.Epoch := cbxEpoch.Checked;
end;

procedure TfrmNeuralNetExtend.speEpochCountChange(Sender: TObject);
begin
  NeuralNetExtended.EpochCount := StrToInt(speEpochCount.Text);
end;

procedure TfrmNeuralNetExtend.cbxMaxTeachErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MaxTeachError := cbxMaxTeachError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMaxTeachErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MaxTeachErrorValue :=
StrToFloat(edtMaxTeachErrorValue.Text);
end;

procedure TfrmNeuralNetExtend.cbxMidTeachErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MidTeachError := cbxMidTeachError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMidTeachErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MidTeachErrorValue :=
StrToFloat(edtMidTeachErrorValue.Text);
end;
```

```
procedure TfrmNeuralNetExtend.cbxTeachIdentClick(Sender: TObject);
begin
  NeuralNetExtended.TeachIdent := cbxTeachIdent.Checked;
end;

procedure TfrmNeuralNetExtend.speTeachIdentValueChange(Sender: TObject);
begin
  NeuralNetExtended.TeachIdentCount := speTeachIdentValue.Value;
end;

procedure TfrmNeuralNetExtend.cbxMaxTestErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MaxTestError := cbxMaxTestError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMaxTestErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MaxTestErrorValue := StrToFloat(edtMaxTestErrorValue.Text);
end;

procedure TfrmNeuralNetExtend.cbxMidTestErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MidTestError := cbxMidTestError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMidTestErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MidTestErrorValue := StrToFloat(edtMidTestErrorValue.Text);
end;

procedure TfrmNeuralNetExtend.cbxTestIdentClick(Sender: TObject);
begin
  NeuralNetExtended.TestIdent := cbxTestIdent.Checked;
end;

procedure TfrmNeuralNetExtend.speTestIdentValueChange(Sender: TObject);
begin
  NeuralNetExtended.TeachIdentCount := speTestIdentValue.Value;
end;

// Зупинка навчання
procedure TfrmNeuralNetExtend.NeuralNetExtendedAfterTeach(Sender: TObject);
begin
  btnBack.Enabled := true;
  btnNext.Enabled := true;
  btnCancel.Enabled := true;
  btnContinueTeach.Caption := 'Навчити';
  NeuralNetExtended.StopTeach := true;
end;

procedure TfrmNeuralNetExtend.RunTeach;
begin
  Teach := not Teach;  // Перемикач стану "вчимося/не вчимося"
  if Teach then
  begin
    btnBeginTeach.Enabled := false;
    btnBack.Enabled := false;
    btnNext.Enabled := false;
    btnCancel.Enabled := false;
    NeuralNetExtended.StopTeach := false;
    btnContinueTeach.Caption := 'Зупинити навчання';
    NotSaved := true;
    NeuralNetExtended.Train;  // Запуск нейромережі на навчання
    btnCompute.Enabled := true;
    btnSave.Enabled := true;
  end
  else
  begin
    btnBeginTeach.Enabled := true;
```

```
    btnBack.Enabled := true;
    btnNext.Enabled := true;
    btnCancel.Enabled := true;
    btnContinueTeach.Caption := 'Продовжити навчання';
    NeuralNetExtended.StopTeach := true; // Зупинити навчання
  end;
end;

// Відкриття сторінки - тестування навченої нейромережі
procedure TfrmNeuralNetExtend.TestNet;
var
  i, j: integer;
begin
  stgInput.RowCount := NeuralNetExtended.InputFieldCount + 1;
  stgInput.Cells[0, 0] := 'Поле';
  stgInput.Cells[1, 0] := 'Значення';

  // Проставити імена вхідних полів
  j := 0;
  for i := 0 to NeuralNetExtended.AvailableFieldsCount - 1 do
    if (NeuralNetExtended.Fields[i].KindName = fdInput) then // Ознака того, що
поле вхідне
    begin
      Inc(j);
      stgInput.Cells[0, j] := NeuralNetExtended.Fields[i].Name;
    end;
  stgOutput.RowCount := NeuralNetExtended.OutputFieldCount + 1;
  stgOutput.Cells[0, 0] := 'Поле';
  stgOutput.Cells[1, 0] := 'Значення';

  // Проставити імена вихідних полів
  j := 0;
  for i := 0 to NeuralNetExtended.AvailableFieldsCount - 1 do
    if (NeuralNetExtended.Fields[i].KindName = fdOutput) then  // Ознака того,
що поле вихідне
    begin
      Inc(j);
      stgOutput.Cells[0, j] := NeuralNetExtended.Fields[i].Name;
    end;
end;

// Натискання кнопки "Обчислити"
procedure TfrmNeuralNetExtend.btnComputeClick(Sender: TObject);
var
  xVectorFloat: TVectorFloat;
  i: integer;
begin
  // Створити вектор, що будемо подавати на вхід
  // довжиною, рівною кількості нейронів на вхідному шарі
  SetLength(xVectorFloat, NeuralNetExtended.InputFieldCount);
  // Заповнити значення елементів вектора
  for i := 0 to NeuralNetExtended.InputFieldCount - 1 do
    xVectorFloat[i] := StrToFloat(stgInput.Cells[1, i + 1]);
  // Подати на вхід нейромережі. Результати будуть у вихідному шарі нейромережі
  NeuralNetExtended.ComputeUnPrepData(xVectorFloat);
  // Відобразити отримані результати
  for i := 0 to NeuralNetExtended.OutputFieldCount - 1 do
    stgOutput.Cells[1, i + 1] := FloatToStr(NeuralNetExtended.Output[i]);
  // Знищити вектор
  SetLength(xVectorFloat, 0);
  xVectorFloat := nil;
end;

// Зберегти навчену нейромережу
procedure TfrmNeuralNetExtend.btnSaveClick(Sender: TObject);
begin
  SaveDialog.InitialDir := ExtractFilePath(NeuralNetExtended.FileName);
  SaveDialog.FileName := ExtractFileName(NeuralNetExtended.FileName);
  if SaveDialog.Execute then
```

```
  begin
    NeuralNetExtended.NnwFile := TIniFile.Create(SaveDialog.FileName);
    NeuralNetExtended.SavePhase1;
    NeuralNetExtended.SavePhase2;
    NeuralNetExtended.SavePhase4;
    NeuralNetExtended.SaveNetwork;
    NotSaved := false;
  end;
end;

end.
```

Hopf.pas - Блок оптимізації структури інформаційного ресурсу

```
unit Hopf;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, Neural_network_Comp, Neural_network_Types, Db, DBTables, ExtCtrls,
DBCtrls, StdCtrls,
  ToolWin, ComCtrls;

type
  TForm1 = class(TForm)
    Table: TTable;
    btnExecute: TButton;
    DBNavigator: TDBNavigator;
    DataSource: TDataSource;
    btnEdit: TButton;
    stgDatabase: TStringGrid;
    stgInput: TStringGrid;
    stgOutput: TStringGrid;
    NeuralNetHopf: TNeuralNetHopf;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    Bevel: TBevel;
    TableLETTERS: TStringField;
    dbMain: TDatabase;
    procedure DataSourceDataChange(Sender: TObject; Field: TField);
    procedure FormActivate(Sender: TObject);
    procedure GridClick(Sender: TObject);
    procedure btnEditClick(Sender: TObject);
    procedure btnExecuteClick(Sender: TObject);
    procedure GridDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure FormCreate(Sender: TObject);
  public
    { Public declarations }
    procedure AddPattern(Value: string);
    procedure Clear(Grid: TStringGrid);
    procedure Init;
    procedure ShowMatrix(Grid: TStringGrid; Value: string);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

// Показати вектор у вигляді сітки
procedure TForm1.ShowMatrix(Grid: TStringGrid; Value: string);
var
  i, j: integer;
begin
  Clear(Grid);
  for i := 0 to Grid.ColCount - 1 do
    for j := 0 to Grid.RowCount - 1 do
    begin
      try
        if Value[i * Grid.RowCount + j + 1] = '1' then
          Grid.Cells[i, j] := '1'
        else
          Grid.Cells[i, j] := ' '
      except
        Grid.Cells[i, j] := ' '
```

```
      end;
    end;
end;

// Очистити сітку
procedure TForm1.Clear(Grid: TStringGrid);
var
  i, j: integer;
begin
  for i := 0 to Grid.ColCount - 1 do
    for j := 0 to Grid.RowCount - 1 do
      Grid.Cells[i, j] := ' ';
end;

// Показати символ з таблиці
procedure TForm1.DataSourceDataChange(Sender: TObject; Field: TField);
begin
  ShowMatrix(stgDatabase, TableLETTERS.Value);
end;

// Ініціалізація мережі значеннями з таблиці
procedure TForm1.Init;
begin
  // Очистити мережу від зразків
  NeuralNetHopf.ResetPatterns;

  // Додати зразки з таблиці до мережі
  Table.First;
  while not Table.Eof do
  begin
    AddPattern(TableLETTERS.AsString);
    Table.Next;
  end;
  Table.First;

  // Ініціалізувати ваги
  NeuralNetHopf.InitWeights;
  // Мережа підготовлена до розпізнавання
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  Clear(stgDatabase);
  Clear(stgInput);
  Clear(stgOutput);
  Init;
end;

procedure TForm1.GridClick(Sender: TObject);
begin
  with Sender as TStringGrid do
    if Cells[Col, Row] = '1' then
      Cells[Col, Row] := ' '
    else
      Cells[Col, Row] := '1'
end;

// Додавання нового образа до мережі
procedure TForm1.AddPattern(Value: string);
var
  i: integer;
  xVector: TVectorInt;
begin
  SetLength(xVector, stgDatabase.RowCount * stgDatabase.ColCount);

  // Перетворення символьного рядка у вектор
  for i := 1 to stgDatabase.RowCount * stgDatabase.ColCount do
    try
      if TableLETTERS.AsString[i] = '1' then
```

```
        xVector[i - 1] := 1
      else
        xVector[i - 1] := -1;
    except
      xVector[i - 1] := -1;
    end;


  NeuralNetHopf.AddPattern(xVector);
end;

procedure TForm1.btnEditClick(Sender: TObject);
var
  i, j: integer;
  xString: string;
begin
  xString := '';
  for i := 0 to stgDatabase.ColCount - 1 do
    for j := 0 to stgDatabase.RowCount - 1 do
      if stgDatabase.Cells[i, j] = '1' then
        xString := xString + '1'
      else
        xString := xString + ' ';
  Table.Edit;
  TableLETTERS.AsString := xString;
  Table.Post;
end;

// Розпізнавання символу
procedure TForm1.btnExecuteClick(Sender: TObject);
var
  i, j: integer;
  xString: string;
begin
  // Подаємо сигнали на вихід мережі
  for i := 0 to stgInput.ColCount - 1 do
    for j := 0 to stgInput.RowCount - 1 do
      if stgInput.Cells[i, j] = '1' then
        NeuralNetHopf.Layers[1].Neurons[i * stgInput.RowCount + j].Output := 1
      else
        NeuralNetHopf.Layers[1].Neurons[i * stgInput.RowCount + j].Output := -1;

  // Запуск процесу розпізнавання
  NeuralNetHopf.Calc;

  // Перетворення виходів мережі до рядка
  xString := '';
  for i := 1 to stgOutput.RowCount * stgOutput.ColCount do
    if NeuralNetHopf.Layers[1].Neurons[i - 1].Output = 1 then
      xString := xString + '1'
    else
      xString := xString + ' ';

  // Відобразити результат
  ShowMatrix(stgOutput, xString);
end;

procedure TForm1.GridDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  with Sender as TStringGrid do
  begin
    Canvas.Brush.Color := clBlack;
    if Cells[ACol,ARow] <> ' ' then
      Canvas.FillRect(Rect)
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
  dbMain.Params.Values['Path'] := ExtractFilePath(Application.ExeName);
  dbMain.Open;
  Table.Open;

end;

end.
```