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 автоматизованого реінжирінгу”**



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

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 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

*Правдюку Дмитру Валерійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи автоматизованого реінжирінгу*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
( 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 автоматизованого реінжирінгу*

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

**Правдюк Д.В. Дослідження та програмна реалізація системи автоматизованого реінжирінгу. 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 автоматизованого реінжирінгу.

Метою розробки є дослідження та програмна реалізація системи автоматизованого реінжирінгу.

Об'єктом дослідження є процес автоматизованого реінжирінгу.

Предметом дослідження є методи автоматизованого реінжирінгу.

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

Результат роботи – програмна реалізація системи автоматизованого реінжирінгу.

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 Rescueware Workbench, Delphi.

**Ключові слова**: комп'ютерна інженерія, реінжирінг

# ABSTRACT

**Pravdiuk D.V. Research and software implementation of the automated reengineering system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.**

In this final qualification work for the second (master's) level of higher education, software designed for the automated re-engineering system was developed.

The goal of the development is the research and software implementation of the automated reengineering system.

The object of research is the process of automated reengineering.

The subject of research is the methods of automated reengineering.

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

The result of the work is the software implementation of the automated reengineering system.

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 Rescueware Workbench, Delphi environment.

**Keywords:** computer engineering, reengineering

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DHCP– протокол, що використовується для динамічного розподілу IP-адрес.

DNS – розподілена служба Інтернет, використовувана для зіставлення логічних (доменних) імен і IP-адрес. DNS використовується для забезпечення можливості роботи зі зрозумілими й іменами, що легко запам'ятовуються, замість IP-адрес у числовому форматі

ICMP – протокол

IDS – система виявлення атак

IP – адресний протокол

LAN – локальна мережа

NAT – трансляція IP-адрес із одного адресного простору в IP-адреси іншого адресного простору

Proxy – програма-посередник, що транслює запити різних протоколів із локальної мережі в зовнішню мережу

TCP – протокол обміну даними на транспортному рівні

UDP – протокол

URL – уніфікований покажчик інформаційного ресурсу (стандартизований рядок символів, що вказує місцезнаходження документа в мережі Інтернет)

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

# ВСТУП

**Актуальність теми**. Система призначена для реінжинірингу коду застарілого програмного забезпечення, який був реалізований на мові Pascal, на сучасні мови програмування, у тому числі й на Delphi 10. Реінжиніринг програмного забезпечення (англ. software reengineering) – повторна реалізація успадкованої системи з метою підвищення зручності її використання, супроводження, розширення функціоналу чи впровадження оновлених технологій.

У сфері технологій, що постійно розвивається, програмне забезпечення панує безперечно . Але що станеться, коли ваше колись передове програмне забезпечення почне втрачати свій блиск? Навіть якщо ви розробили найкраще програмне забезпечення свого часу, завжди є місце для вдосконалення.

Ласкаво просимо до захоплюючої подорожі реінжинірингу програмного забезпечення – процесу, який дозволяє воскресити, вдосконалити та революціонізувати існуюче програмне забезпечення.

Реінжиніринг програмного забезпечення – це не просто розкіш; це необхідність у сучасному динамічному діловому середовищі. Оскільки технології розвиваються, очікування користувачів зростають, а вимоги ринку змінюються, випередження стає найважливішим. Застосовуючи реінжиніринг програмного забезпечення, ви можете підвищити продуктивність, масштабованість і ефективність, гарантуючи, що ваше програмне забезпечення буде конкурентоспроможним і відповідатиме мінливим галузевим стандартам.

**Мета й завдання дослідження.** Метою роботи є дослідження та програмна реалізація системи автоматизованого реінжирінгу.

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

– Огляд існуючих систем автоматизованого реінжирінгу.

– Дослідження системи автоматизованого реінжирінгу.

– Програмна реалізація системи автоматизованого реінжирінгу.

*Об'єктом дослідження* є процес автоматизованого реінжирінгу.

*Предметом дослідження* є методи автоматизованого реінжирінгу.

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

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

– Удосконалено метод автоматизованого реінжирінгу.

– Розроблено вітчизняний продукт автоматизованого реінжирінгу, який має більш широкі можливості, на відміну від існуючих аналогів.

**Практична цінність отриманих результатів** полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автоматизованого реінжирінгу.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи автоматизованого реінжирінгу, 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

Реінжиніринг програмного забезпечення – це трансформаційний процес, спрямований на вдосконалення та оновлення застарілих систем, щоб відповідати поточним і майбутнім вимогам. Це складна робота, яка включає аналіз, перепроектування та модифікацію програмного забезпечення для усунення обмежень, підвищення продуктивності та узгодження його з мінливими потребами бізнесу. Таким чином, розробники програмного забезпечення для цієї ролі повинні вимагати повного розуміння дизайну, архітектури та функціональності програмної системи.

Процес реінжинірингу програмного забезпечення охоплює такі ключові етапи, як зворотний інжиніринг, реструктуризація та прямий інжиніринг. Ці кроки дають змогу розробникам програмного забезпечення виявляти та вирішувати проблеми в рамках застарілої системи та оновлювати її відповідно до поточних і майбутніх вимог.

За своєю суттю проекти реінжинірингу програмного забезпечення – це практика відновлення та омолодження програмного забезпечення, яке застаріло або більше не відповідає поточним вимогам. Він передбачає всебічну оцінку архітектури програмного забезпечення, вихідного коду та функціональних можливостей для визначення областей для вдосконалення. За допомогою систематичного аналізу та модифікацій мета полягає в тому, щоб підвищити продуктивність програмного забезпечення, зручність обслуговування та масштабованість.

Реінжиніринг програмного забезпечення відрізняється від інших процесів розробки програмного забезпечення, таких як нова розробка або супровід. У той час як нова розробка передбачає створення програмного забезпечення з нуля, а

технічне обслуговування зосереджується на виправленні помилок і внесенні незначних оновлень, реінжиніринг зосереджується навколо трансформації та оптимізації існуючого програмного забезпечення. Реінжиніринг використовує основу вже функціональної системи, надаючи їй нове життя за допомогою стратегічних модифікацій і вдосконалень.

### 1.2 Область застосування

Проект реінжинірингу програмного забезпечення пропонує численні переваги, які безпосередньо впливають на операційну ефективність, конкурентоспроможність і зростання бізнесу:

– Покращена продуктивність: Реінжиніринг дозволяє підприємствам оптимізувати продуктивність програмного забезпечення, збільшивши час відгуку, зменшивши вузькі місця та підвищивши загальну ефективність. Це дозволяє програмам справлятися з великими робочими навантаженнями та ефективно масштабуватися.

– Покращена ремонтопридатність: застаріле програмне забезпечення часто стає складним для обслуговування та підтримки. Завдяки реінжинірингу підприємства можуть оптимізувати та спростити структуру програмного забезпечення, полегшуючи його розуміння, оновлення та підтримку з часом.

– Адаптація до мінливих потреб: бізнес-вимоги розвиваються, і програмне забезпечення має йти в ногу. Реінжиніринг дозволяє компаніям узгоджувати своє програмне забезпечення з поточними потребами, включаючи нові функції, технології та інтеграції, щоб відповідати мінливим вимогам ринку.

– Економія коштів: заміна всієї системи програмного забезпечення може коштувати дорого. Реінжиніринг програмного забезпечення пропонує економічно ефективну альтернативу завдяки використанню наявних інвестицій і поступовому вдосконаленню функціональності програмного забезпечення, зменшуючи потребу в повному перегляді.

– Конкурентні переваги: Оновлення та модернізація програмного забезпечення за допомогою реінжинірингу забезпечує підприємства конкурентною перевагою. Це дозволяє їм пропонувати покращений досвід користувача, покращену продуктивність та інноваційні функції, які відрізняють їх від конкурентів.

Реінжиніринг програмного забезпечення має потенціал для перетворення непродуктивного програмного забезпечення на цінний актив для бізнесу. Вдихаючи нове життя в існуючі системи, оптимізуючи продуктивність і адаптуючись до мінливих потреб, реінжиніринг дозволяє компаніям залишатися гнучкими, ефективними та конкурентоспроможними в сучасному технологічному середовищі.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи автоматизованого реінжирінгу, 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

**CASE-систем Rational Rose фірми Rational Software Corporation**

Системи, що входять до складу цього сімейства, призначаються для автоматизації етапів аналізу й проектування, а також для генерації кодів на різних мовах і випуску проектної документації. Всі CASE-засоби, що входять до складу Rational Rose, використовують методи Граді Буча (він є науковим керівником розробок) і OMT Джеймса Рамбо, підтримуючи єдиний графічний інтерфейс із користувачем. Розходження між цими системами мінімальні й визначаються мовами, на яких генеруються коди програм.

У складі Rational Rose можна виділити 6 основних структурних компонентів: репозиторій, графічний інтерфейс користувача (GUI), засоби перегляду проекту (browser), засоби контролю проекту, засоби збору статистики й генератор документів. Всі перераховані компоненти є загальними для всіх систем сімейства. До них варто додати генератор кодів – індивідуальний для кожної системи – і аналізатор – окремий програмний модуль, що забезпечує реінжиніринг.

Репозиторій являє собою об’єктно-орієнтовану базу даних. Засоби перегляду забезпечують "навігацію" по проекту, у тому числі, переміщення нагору/долілиць по ієрархіях класів і підсистем, перемикання з одного виду діаграм до іншого й т.д. Засобу контролю й збору статистики дають можливість знаходити й усувати помилки в міру розвитку проекту, а не після завершення його опису. Генератор звітів формує тексти вихідних документів на основі інформації, яка утримується в репозиторії.

Rational Rose включає засоби автоматичної генерації кодів програм на мовах третього й четвертого поколінь. Використовуючи інформацію, що утримується в логічній і фізичній моделях проекту, генератор кодів формує файли заголовків і файли описів класів і об'єктів. Створюваний у такий спосіб кістяк програми може бути уточнений шляхом прямого програмування.

Аналізатор кодів створює моделі проектів на основі інформації, що утримується в обумовлених користувачем вихідних текстах програм. У процесі роботи аналізатор здійснює контроль правильності вихідних текстів і діагностику помилок. Модель, отримана в результаті його роботи, може цілком або фрагментарно використовуватися в різних проектах. У такий спосіб забезпечується можливість повторного використання програмних компонентів.

В основі роботи Rational Rose лежить побудова різного роду діаграм і специфікацій, що визначають логічну й фізичну структури моделі, її статичні й динамічні аспекти. Діаграми класів показують класи і їхні ієрархії, відображаючи логікові побудови прикладної системи. Для великих систем звичайно будується не одна а кілька подібних діаграм. Діаграма класів визначає тільки статичні аспекти, що відносяться до класів. Динаміка їхнього поводження представляється на діаграмах станів. Діаграми сценаріїв показують існуючі об'єкти і їхня взаємодія в логічному проекті прикладної системи. Модульні діаграми визначають розподіл класів і об'єктів у модулях, що фізично реалізують проект. Одна модульна діаграма представляє всю або частину модульної архітектури системи. Для рішення завдання розподілу апаратних ресурсів в Rational Rose використовуються діаграми процесів.

Не вся необхідна інформація про проект може бути наочно відбита в діаграмах, тому Rational Rose містить засоби уведення специфікацій, які доповнюють набір діаграм. Специфікації створюються для класів, клас-утиліт, операцій, підсистем, об'єктів, модулів і т.д.

Основні властивості Rational Rose, що забезпечують її високу конкурентоспроможність на світовому ринку програмних засобів:

– охват всіх етапів життєвого циклу роботи над проектом з єдиною методикою й нотацією;

– можливість повторного використання програмних розробок користувачів за рахунок засобів реінжинірингу;

– наявність засобів автоматичного контролю, що дозволяють вести налагодження проекту в міру його розробки;

– можливість опису проекту на різних рівнях для різних категорій користувачів;

– зручний для користувача графічний інтерфейс;

– автоматична генерація кодів на мовах C++, Ada, Smalltalk, PowerBuilder, SQLWindows, VisualBasic;

– наявність засобів групової розробки;

– широкий спектр застосування системи – бази даних, банківські системи, телекомунікація, системи реального часу й т.д.;

– можливість використання різних платформ: IBM PC (у середовищі Windows), Sun SPARC Stations (Solaris, SunOS), Hewlett-Packard (HP UX), IBM RS/6000 (AIX).

Система Rational Rose/C++ була застосована для реінжинірингу банківської інформаційної системи. Це – складна система з повною передісторією, що містить системні бібліотеки й додатки. Реінжиніринг додатків, що містять понад 500 класів, виконувався близько 25 годин.

Окремо на тій же платформі був виконаний реінжиніринг бази дані системи за допомогою CASE-засобу SILVERRUN. За час порядку 5 мінут була побудована ER-модель бази даних, що містить близько 200 таблиць.

Результати можна сформулювати в такий спосіб:

1. Система Rational Rose/C++ забезпечує реінжиніринг складних інформаційних систем на базі досить обмежених ресурсів.

2. CASE-засіб SILVERRUN забезпечує реінжиніринг складних баз даних за мінімальний час.

3. Відновлено архітектуру системи у вигляді ієрархії категорій класів.

4. Отримано логічну модель системи у вигляді сукупності діаграм і специфікацій класів.

5. Побудовано фізичну модель системи у вигляді діаграм підсистем і модулів.

6. Аналіз результатів реінжинірингу дозволив визначити найбільш істотні недоліки існуючого комплексу програм і шляхи їхнього виправлення. От кілька прикладів:

– У рамках проекту визначені загальні класи, які окремими розроблювачами можуть перевизначатися. При цьому кожний розроблювач має право створювати новий клас із тим же ім'ям, що й загальний клас, у своєму каталозі. У результаті дублюється код, зміни, внесені в загальні класи, потрібно повторювати в каталозі розроблювача, виникає плутанина з іменами класів, що особливо незручно при побудові й аналізі моделі. Об’єктно-орієнтований підхід пропонує простий і природний шлях виправлення таких ситуацій – застосування механізму спадкування. Розроблювачі повинні не дублювати загальний клас, а заводити клас-спадкоємець (з оригінальним ім'ям).

– У системі представлені кілька різних проектів, причому в проекті можуть використовуватися класи з інших проектів шляхом вказівки доступу до відповідних програмних модулів. Це погано, тому що клас у проекті-власнику може бути змінений, а в інших це не потрібно, що може приводити до зовсім непередбачених помилок, які можуть виникати навіть без змін вихідних кодів, а просто при повторному конфігуруванні системи. Потрібно сформувати бібліотеку класів, що представляють типові проектні рішення, куди варто занести всі класи, які розділяються між різними проектами. Ці класи змінювати не слід, а при необхідності внесення змін або уточнень у проекті заводиться клас-спадкоємець.

– Реінжиніринг і аналіз окремих додатків показав, що є багато класів, які не зв'язані відносинами з іншими класами. Причина полягає в тому, що не витриманий об’єктно-орієнтований стиль програмування. У додатках є досить

складні функції на З (наприклад, main), які виконують ролі керуючих класів. Таке положення справ не відповідає завданню супроводу, оскільки велика кількість фактичних зв'язків, наявних у системі, у моделі буде відсутній. У результаті простежування й локалізація помилок буде утруднена. Вихід із цієї ситуації очевидний. Варто повністю використовувати об’єктно-орієнтовані можливості C++, створюючи керуючі класи й мінімізуючи використання функцій на С.

Побудовані діаграми класів і модулів є основою для розробки повної моделі системи шляхом доповнення їхніми діаграмами станів і взаємодій, які створюються за допомогою того ж CASE-засобу.

## 2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

**RescueWare Workbench**

Конвертор Rescueware Workbench є серцем інтегрованого середовища реінжинірингу Rescueware Workbench [11]. Для всіх підтримуваних типів компонентів він робить перевірку на наявність помилок і відсутніх модулів, збирає інформацію для користувальницької оболонки системи. Для компонентів програмної логіки й користувальницького інтерфейсу їм виконується переклад на нові мови програмування (зараз підтримуються Delphi, C++, Java, Visual Basic і ін.) Слід особливо зазначити таку його частину, як побудова програмних зрізів [7].

Ключовими ідеями в даному трансляторі є подання програми у вигляді дерева [10] і наявність проміжної мови (ПМ) для обробки програмної логіки. Для цього кожна використовувана мова має свій набір класів, а ПМ побудована на вимогу максимальної незалежності від вхідних мов [1]. Для додавання нової вхідної мови досить додати моделюючий її набір класів, синтаксичний аналізатор, що синтезує прохід і підтримку часу виконання, а інші частини вимагають мінімальних змін. До переваг даної моделі можна віднести й

можливість використання поліпшуючої програми переглядів, наприклад, видалення невикористовуваних змінних. Модульність архітектури конвертора виразилася у вигляді реалізації набора бібліотек динамічного компонування (DLL) по функціональному принципу.

Для більш повного розуміння побудованої системи динамічної підтримки, коротко опишемо попередні етапи конвертора, що стосуються перекладу програмної логіки й компонент користувальницької взаємодії.

Переклад програмної логіки при використанні даної системи автоматизованого перекладу складається з наступних етапів. Спочатку програма подається на вхід синтаксичному аналізатору мови Pascal. Коли зустрічається вставка на іншій мові, то він робить перемикання на відповідний аналізатор. Результатом роботи цього етапу служить дерево, що складається зі специфічних для вхідних мов вузлів.

Докладно цей етап описаний у роботі [2]. Потім виробляється одержання дерева проміжного подання з результату попереднього проходу. Це відбувається в кілька стадій. Синтези SQL і Pascal роблять проміжне подання [2], у якому залишилися деякі конструкції мови Pascal. Наступна реструктуризація розбиває набір параграфів програми на безліч процедур, базуючись на керуючих операторах, що залишилися [4]. Слідом виробляється  етап повного видалення або заміщення нелокальних на локальні оператори переходу. Варіант виконуваної дії залежить від підтримки цільовою мовою оператора локального переходу [5]. Після завершення переглядів, що залежать від вхідної мови, у дереві програми присутні SQL і CICS вузли.

Після цього можуть застосовуватися етапи, що поліпшують програму, наприклад видалення невикористовуваних змінних, побудовувач класів і ін. Слідом виробляються перегляди для одержання програми вихідною мовою, такі як синтез цільової мови й SQL з наступною генерацією. Завданням синтезу вихідної мови також є обробка CICS вузлів. Перегляд, що генерує, в основному, відповідальний за печать тексту програми вихідною мовою. Тому завдання

сполучення переведеної програмної логіки з підтримкою часу виконання лежать на синтезі цільової мови й SQL.

Як цільова мова для перекладу програмної логіки зараз використовуються C++, Java і Visual Basic.

Опишемо процес перекладу для інтерфейсних компонент. Як ми вже відзначали, даний переклад досить простий і складається із трьох наступних стадій:

1. Перевірка коректності. Синтаксичний аналізатор перевіряє правильність опису інтерфейсної компоненти й у випадку успіху генерує файлове подання в єдиному проміжному форматі для можливості використання генерацією й візуальною оболонкою.

2. Необов'язкова участь користувача. Перегляд і модифікація зовнішнього вигляду інтерфейсних компонент за допомогою візуальних засобів, що входять в RescueWare Workbench. Можливість додавання різних графічних елементів, таких як кнопка, меню й 'гаряча' клавіша, із вказівкою дії, що відбуває при його використанні, наприклад перехід на іншу інтерфейсну компоненту.

3. Генерація у вихідну мову (Visual Basic, HTML, Java, C++). Відзначимо, що для деяких випадків (C++ і Visual Basic) немає підтримки часу виконання, тому що можна обійтися язиковими засобами й це незначно збільшує породжений код.

RescueWare являє собою конвертер, що забезпечує перенос застарілих програмних систем на рейки нових технологій. Внутрішню організацію конвертера можна розбити на дві частини: синтезуючу й ту, що генерує.

1. Синтезуюча частина системи ([2], [3]) переводить дану програму в проміжну мову, близьку по ідеології до мов третього покоління, зберігаючи вихідні ідентифікатори, функціональність, прив'язку до вихідного тексту, і т.п.

2. Далі програма проміжною мовою надходить у розпорядження частини, що генерує, конвертера, що для програм на Pascal генерує тексти програм на C++, Java і Visual Basic, а для екранних форм – файли опису ресурсів, HTML і ASP

файли. Крім цього генерація вирішує завдання по перекладу додатків на нові технології, наприклад, переклад додатка, що використовує алфавітно-цифровий екран, у додаток працюючий з GUI.

3. Для коректної роботи отриманої в результаті генерації програми потрібна бібліотека динамічної підтримки (Runtime support library), докладно описана в [5].

**Delphi 10**

Delphi 10 це – новітня версія популярного рішення для розробки додатків не тільки для Windows, OS X, iOS і Android, але й тепер для «Internet of Things». Даний реліз включає багато нового: нові можливості для VCL, бібліотека паралельних обчислень, поліпшена бібліотека FireMonkey для розробки кроссплатформених додатків, корпоративні мобільні сервіси (EMS) і засобу роботи з «Internet of Things» завдяки новій можливості взаємодії через Bluetooth.

Дизайнер єдиного інтерфейсу (Multi-Device Designer):

– Delphi 10 забезпечує можливість створювати на базі єдиного вихідного коду нативні додатки для Windows, OS X, iOS і Android.

– Новий революційний спосіб побудови єдиного користувальницького інтерфейсу, заснованого на візуальному дизайнері форм, що адаптується до різних типів пристроїв, форм-факторів і операційних систем.

– Можливість розробити один раз користувальницький інтерфейс, а потім переглядати й коректувати його для кожного типу пристроїв (мобільних, планшетів і настільних систем; iOS, Android, Windows і OS X).

– Можливість робити зміни в головному користувальницькому інтерфейсі, які поширяться на інші пристрої конкретних користувальницьких інтерфейсів; тонке настроювання конкретної форми не торкаючись інших форм.

– Можливість налаштовувати стилі для кожного пристрою за бажанням, при необхідності використовування успадковані стилі.

– Поставляється набір стандартних шаблонів під розміри й розмітки для різних популярних мобільних пристроїв, планшетів і настільних систем. Ви так само можете додавати свої шаблони.

– Використання всього лише одного екземпляра класу форм дозволяє розроблювачеві створювати єдиний вихідний файл по керуванню всіма основними подіями користувальницького інтерфейсу.

Поведінкові сервіси FireMonkey:

– Новий API сервіс, що дозволяє платформі й розроблювачеві запитувати інформацію про пристрій на якому працює додаток, що динамічно підбудовується під форм-фактор і платформу.

– Положення елементів користувальницького інтерфейсу автоматично підбудовується під кожний конкретний пристрій (наприклад, позиція вкладок на різних мобільних платформах).

Компонент MultiView:

– Новий компонент MultiView, унікальний компонент в FireMonkey який використовує сервіси поводження, для того щоб динамічно перемкнути показ залежно від форм-фактора й бажаного поводження платформи.

– MultiView відображає меню на телефоні у вигляді drawer, а на планшеті як popup або docked меню.

Корпоративні мобільні сервіси (Enterprise Mobility Services):

– Корпоративні мобільні сервіси – ключовий елемент платформи для корпоративних мобільних додатків (Mobile Enterprise Application Platform), заснований на REST технологіях стек проміжного ПЗ, що містить у собі хостинг API, розміщення й доступ до SQL баз даних для обслуговування клієнтів мобільних, настільних і web-додатків.

– Масштабоване, без збереження стану, засноване на REST керування даними на кожному із проміжних рівнів ПЗ.

– Створення спеціалізованих API для бізнес функціонала.

– Високоефективний доступ до корпоративних баз даних Oracle, DB2, Microsoft SQL Server Informix, SQL Server, і доступ до іншого функціонала з мобільних додатків через ПЗ проміжного рівня.

– Керування спеціалізованим мобільним API за допомогою REST/JSON за допомогою підзавантажуємих модулів.

– Компоненти доступу до даних у багаторівневій архітектурі за допомогою стандартних REST/JSON.

– Статистика роботи користувачів, сесій і викликів API, і генерація звітів за допомогою web-інтерфейсу.

– Сервіс безпечного зберігання даних як на серверах, так і на мобільних пристроях.

– Швидкий доступ з ваших додатків до корпоративних API, базам даних і сервісам на ваших серверах або в приватних хмарах.

– Гнучкі варіанти ліцензування для розроблювачів. Покупка ліцензій на поширення на основі кількості користувачів, які мають доступ до корпоративних мобільних сервісів (EMS).

З’єднання за допомогою App Tethering, REST і Bluetooth:

– За допомогою App Tethering, RAD Studio робить надзвичайно легким процес з пошуку й взаємодії з іншими додатками за допомогою локальної мережі, Wi-Fi або Bluetooth.

– Розширений функціонал існуючих VCL windows-додатків за допомогою мобільних додатків-компаньйонів або Bluetooth пристроїв простим додаванням пари компонентів у кожний з додатків.

– Використовування технології Bluetooth або Bluetooth LE для з'єднання з гаджетами, такими як, медичні датчики й пристрої, що носяться, для створення унікальних спеціалізованих рішень.

– Використовування простих REST API для з'єднання з будь-яким розташованим рядом пристроєм і взаємодії з ним.

– Спеціальні Bluetooth LE компоненти для роботи із пристроями чутливими до споживання енергії.

Бібліотека паралельних обчислень (Parallel Computing Library):

– Бібліотека паралельних обчислень дозволяє вам експоненційно збільшити продуктивність, спрощуючи написання багатопоточних додатків які повною мірою використовують переваги багатоногоядерних процесорів.

– Підвищена продуктивність існуючих VCL і FireMonkey додатків, за допомогою самонастроювального пула потоків.

Використовування нового модуля System.Threading дозволяє:

– Паралельно for-цикли.

– Планування завдань.

– Асинхронні завдання.

– Об'єднання декількох завдань у потоки.

– Проста інтеграція для істотного збільшення продуктивності.

IDE і інструменти розробки:

– Новий PAServer Manager дозволяє запускати й управляти декількома екземплярами PAServer із системного трея OS X (платформний помічник для розроблювачів під iOS і OS X).

– Поліпшена підтримка системи керування версіями Subversion, додана підтримка системи керування версіями Git.

– Новий інтегрований дизайн єдиного інтерфейсу.

– Середовище розробки підтримує нові Java класи в Android apk.

– У середовище розробки убудовані короткі керівництва для допомоги в ознайомленні з головними можливостями RAD Studio.

– Нова утиліта Java2Delphi для генерації натівних bridge-файлів.

Поліпшення в RTL і VCL:

– Нова бібліотека OmniXML для прискорення продуктивності, зокрема для мобільних платформ і більше простого вибору движка XML.

– Поліпшена низькорівнева обробка JSON.

– Оновлено бібліотеку STL для iOS і нові версії бібліотек Boost і Dinkumware для Win64.

– Удосконалено компонент роботи з панеллю завдань, що з'явився в XE6, для підтримки списку, що випадає, панелі завдань в Windows разом з новим VCL-компонентом TJumpLists.

– Новий модуль кодування й декодування для web.

– Безліч інших поліпшень якості й продуктивності.

Поліпшення мови Object Pascal:

– Удосконалено ініціалізацію динамічних масивів.

– Можливість змішувати й сполучити постійні й динамічні масиви.

– Аналогічні строковим операції на динамічних масивах, включаючи '+' оператор для конкатенації масивів, Insert і Delete.

– За допомогою конструктора масиву можна задати початкове значення глобальної змінної динамічного масиву.

– Так само оптимізовані Дженерики (generics).

Поліпшення в БД і FireDAC:

– БД IBLite, що вбудовується тепер доступна в Windows, OS X, Android і iOS.

– FireDAC підтримує поля blob і файлові потоки СУБД MSSQL.

– Натівна підтримка timeouts команд API СУБД.

– Параметри підключення FireDAC тепер відображаються в інспекторі об'єктів (Object Inspector) у вигляді запису.

– В удосконаленому Data Explorer тепер показуються первинні ключі і їхнього поля, так само, як і зовнішні ключі. Показуються об'єкти типу sequences/generators.

– Інші поліпшення в FireDAC, включаючи новий драйвер для IBLite для настільних і мобільних додатків.

Поліпшення в FireMonkey:

– Компоненти TEdit і TCalendar використовують новий підхід: у процесі роботи в iOS ви можете вибрати використання стандартного стилю або рідного подання.

– Підтримка декількох моніторів для настільних платформ.

– Компонент TBannerAd підтримує API Google Mobile Ads від Google Play Services.

– Повноекраний режим на Android KitKat.

– Відновлення змісту TListView за допомогою жесту ' pull-to-refresh' на iOS і Android.

Додаткові можливості Delphi 10:

– Розробка додатків для Windows, OS X, iOS, Android, використовуючи мову й інструменти, які ви знаєте і яким довіряєте (Object Pascal або C++), а також продовжуйте розвивати свої додатки для Windows з використанням оновленої бібліотеки VCL.

– Розробка додатків, підключених до «Internet of Things» за допомогою App Tethering, Bluetooth, і REST запитам на пристрої й гаджетах.

– Використання FireDAC для високопродуктивного доступу до баз даних масштабу підприємства легко й просто.

– Взаємодія з популярними постачальниками сервісів у хмарах за допомогою REST, такими як Baa, для push-повідомлень, автентифікації, зберігання й т.д.

– Зв'язування будь-якого елементу управління інтерфейсу з об'єктами або базами даних. За допомогою LiveBindings будь-який компонент може працювати з даними.

– Створення багатоланкових додатків DataSnap масштабу підприємства. Платформа DataSnap дозволяє транслювати дані з баз даних через проміжні сервера додатків на клієнтські пристрої.

## 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 автоматизованого реінжирінгу.

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

Розпізнавання ознак, які вказують на необхідність реінжинірингу програмного забезпечення, має вирішальне значення для того, щоб ваше програмне забезпечення залишалося ефективним, надійним і відповідало цілям вашої організації.

**Реінжиніринг застарілого програмного забезпечення або проблеми сумісності з існуючими системами**

Однією з явних ознак того, що програмна система може потребувати реінжинірингу, є те, що вона застаріла або стала несумісною з сучасними технологіями. Застаріле програмне забезпечення, створене на застарілих платформах або з використанням застарілих мов програмування , може важко інтегруватися з новими системами або мати незручний інтерфейс користувача. У таких випадках стає необхідною реінжиніринг програмного забезпечення для оновлення системи та забезпечення сумісності з новітніми технологіями.

**Обмеження продуктивності та масштабованості**

Якщо програмна система відчуває вузькі місця продуктивності, часті збої або повільний час відгуку, це вказує на необхідність реінжинірингу. У міру того, як бізнес зростає, а вимоги користувачів зростають, програмні системи повинні бути здатні обробляти більші робочі навантаження та ефективно масштабуватися. Якщо існуюча система не відповідає вимогам до продуктивності або не має можливості масштабування, реінжиніринг може оптимізувати архітектуру програмного забезпечення, покращити алгоритми та покращити керування ресурсами для забезпечення кращої продуктивності та масштабованості.

**Високі витрати на обслуговування та часті збої системи**

Застарілі системи часто вимагають значного обслуговування та спричиняють високі витрати через застарілу інфраструктуру, складну кодову базу та відсутність підтримки з боку постачальників. Якщо система програмного забезпечення потребує частих виправлень, виправлень та оновлень, а пов'язані з цим витрати на технічне обслуговування стають непідйомними, це може бути ознакою необхідності реінжинірингу. За допомогою реінжинірингу програмного забезпечення компанії можуть оптимізувати кодову базу, усунути технічну заборгованість і зменшити навантаження на поточне обслуговування, що призведе до економії коштів і підвищення стабільності системи.

**Подорож процесу реінжинірингу програмного забезпечення**

Процес реінжинірингу програмного забезпечення дійсно можна умовно розділити на три основні фази:

– зворотний інжиніринг;

– реструктуризація;

– прямий інжиніринг.

Ці етапи працюють разом для вдосконалення та модернізації існуючих програмних систем. Розглянемо кожну фазу більш детально.

**Фаза 1: зворотне проектування**

Реверсивне проектування передбачає вивчення та розуміння поточної архітектури програмного забезпечення. Він має на меті отримати уявлення про архітектуру, дизайн і функціональність системи. Ось деякі ключові аспекти зворотного проектування:

– Перевірка коду: розробники вивчають вихідний код, документацію та інші доступні ресурси, щоб зрозуміти, як працює програмне забезпечення. Вони аналізують структуру коду, алгоритми та шаблони, що використовуються в системі.

– Перегляд документації: Існуюча документація, наприклад специфікації системи, посібники користувача та проектні документи, переглядається для збору

інформації про передбачувану функціональність системи та конструктивні рішення.

– Аналіз системи: розробники аналізують поведінку системи, потік даних і взаємодію із зовнішніми компонентами. Вони можуть використовувати інструменти для візуалізації структури системи, визначення залежностей і відстеження потоку виконання.

Основною метою зворотного проектування є отримання повного розуміння існуючої програмної системи. Ці знання допомагають визначити сфери для вдосконалення та закладають основу для наступних етапів.

**Етап 2: Реструктуризація**

Фаза реструктуризації зосереджена на перепроектуванні та реорганізації системи програмного забезпечення для підвищення її продуктивності, зручності обслуговування та масштабованості. Ось основні аспекти етапу реструктуризації:

– Рефакторинг коду: розробники змінюють існуючу кодову базу, щоб покращити її структуру, читабельність і зручність обслуговування. Це може передбачати видалення повторюваного коду, вилучення багаторазових компонентів і застосування шаблонів проектування для підвищення модульності системи.

– Покращення архітектури: архітектуру системи можна переробити, щоб усунути недоліки архітектури, покращити масштабованість або включити нові технології. Це включає переоцінку взаємодії компонентів, впровадження рівнів або модулів і оптимізацію продуктивності системи.

– Видалення застарілого коду: застарілий або невикористаний код визначається та видаляється, зменшуючи складність і покращуючи продуктивність системи. Це розвантажує кодову базу та полегшує її розуміння та підтримку.

Фаза реструктуризації має на меті оптимізувати структуру програмної системи, зробити її більш ефективною, гнучкою та зручною для обслуговування. Він усуває виявлені недоліки та готує систему до майбутніх вдосконалень.

**Фаза 3: передова інженерія**

Фаза передового проектування передбачає використання знань, отриманих під час зворотного проектування, і вдосконалень, зроблених під час фази реструктуризації, для розробки нової та вдосконаленої версії програмного забезпечення. Ось ключові аспекти передового проектування:

– Реалізація оновлених компонентів: розробники впроваджують оновлені або оновлені компоненти на основі інформації, отриманої в результаті зворотного проектування та реструктуризації. Це може включати переписування коду, впровадження нових бібліотек або фреймворків та інтеграцію сучасних технологій.

– Введення нових функцій: оновлене програмне забезпечення може включати нові функції або функції, щоб відповідати оновленим вимогам бізнесу. Це передбачає проектування та розробку додаткових модулів або модулів із розширеними можливостями.

– Тестування та забезпечення якості: проводиться ретельне тестування , щоб переконатися, що оновлене програмне забезпечення відповідає визначеним вимогам і функціонує за призначенням. Це включає в себе різні методи тестування, такі як модульне тестування, інтеграційне тестування та тестування системи, щоб перевірити стабільність, надійність і продуктивність системи.

Фаза передового проектування завершується новою версією програмного забезпечення, яка включає в себе знання, отримані в результаті зворотного проектування, і вдосконалення, зроблені під час реструктуризації. Тепер оновлене програмне забезпечення готове до розгортання та використання.

Дотримуючись цих трьох етапів зворотного проектування, реструктуризації та передового проектування, процес реінжинірингу програмного забезпечення дозволяє підприємствам вдихнути нове життя у свої існуючі системи. Це дає їм змогу оптимізувати продуктивність, підвищити придатність до обслуговування та узгодити сучасні вимоги, гарантуючи, що система залишається надійною та адаптованою до мінливих потреб бізнесу.

### 3.2 Розробка структурної схеми

Реінжиніринг програмного забезпечення – це процес оновлення програмного забезпечення. Цей процес включає розробку додаткових функцій програмного забезпечення та додавання функцій для кращого та ефективнішого програмного забезпечення. Що стосується визначення, цей процес також передбачає, що програмний продукт матиме покращену ремонтопридатність.

Таким чином, реінжиніринг є кроком до постійного вдосконалення програмного забезпечення для кращого розвитку та клієнтського досвіду. Крім того, це спосіб зробити існуючі продукти покращеними.

Необхідність реінжинірингу програмного забезпечення стає невід'ємною частиною покращення якості ваших продуктів. Цей процес додає більше цінності вашому бізнесу, оскільки він не лише покращує ваші послуги, але й сприяє додатковому доходу.

Реінжиніринг програмного забезпечення є економічно ефективним методом розробки програмного забезпечення. чому Цей процес дозволяє виявити непотрібні елементи, реалізовані у вашому поточному програмному забезпеченні, і видалити їх із системи.

Роблячи це, ви мінімізуєте понесені витрати (часові, фінансові, прямі, непрямі тощо). Якщо клієнту потрібен продукт, який у вас уже є, але йому потрібні додаткові функції, вам, можливо, доведеться лише переробити наявний, щоб максимізувати ефективність розробки.

Що стосується економії, цей процес також полегшує обслуговування ваших продуктів. Можливість повторно перевірити програмне забезпечення може допомогти виявити кілька помилок у поточній реалізації.

Це дозволяє групі розробників приймати рішення про вдосконалення процедур розробки. Таким чином, тривалість обслуговування життєвого циклу розробки програмного забезпечення стає набагато легшою для виконання та підтримки.

Процес реінжинірингу програмного забезпечення в основному проходить три основні фази. Це (1) реверс-інжиніринг, (2) реструктуризація та (3) форвард-інжиніринг.

**1. Зворотне проектування**

Простий пошук у Google скаже нам цезворотне проектуванняє «відтворення продукту іншого виробника після детального вивчення його конструкції або складу». Однак цей процес не обмежується лише застосуванням цього процесу до продукту іншого виробника, а й до вашого власного.

Це досягається шляхом ретельного аналізу та перевірки специфікацій системи та розуміння існуючих процесів. Систематично, реверсування життєвого циклу розробки програмного забезпечення для реалізації програмного забезпечення найкраще підходить для цієї процедури, оскільки вона зазвичай розгадує кожен рівень від вищого рівня до нижчого рівня представлення системи.

**2. Реструктуризація**

Після завершення зворотного проектування та визначення відповідних специфікацій виконується реструктуризація . Реструктуризація пов’язана з перевпорядкуванням або реконструкцією вихідного коду та вирішенням питання про збереження чи зміну умов програмування.

Однак це не повинно вплинути на наявні функції програмного забезпечення. Натомість цей процес покращує їхню надійність і придатність до обслуговування.

Іншою частиною цієї процедури є видалення або реконструкція частин вихідного коду, які часто викликають помилки в програмному забезпеченні (також може бути налагодження).

Окрім цього, усунення застарілих або старіших версій певних частин системи (таких як програмна реалізація та апаратні компоненти) має підтримувати оновлення системи.

**3. Передова техніка**

Потік закінчується спередова техніка. Це процес інтеграції останніх специфікацій на основі результатів оцінки зворотного проектування та реструктуризації.

Стосовно процесу в цілому, це визначається відносно зворотного проектування, коли є спроба побудувати назад, від закодованого набору до моделі, або порушити процес інтеграції програмного забезпечення.

Немає конкретної моделі SDLC, якої слід дотримуватися при реінжинірингу програмного забезпечення. Модель завжди залежатиме від того, що найкраще підходить до навколишнього середовища та реалізації вашого продукту.

Однак, як і розробка програмного забезпечення, це систематична розробка, яка включає процеси всередині процесів і вимагає ретельної перевірки для бездоганних результатів.

Зараз багато IT-компаній осягнули концепцію реінжинірингу програмного забезпечення, оскільки вони займаються розробкою програмного забезпечення. З боку клієнта, багато малих і середніх підприємств вдаються до офшорного процесу в інші країни з набагато нижчими витратами та оплатою.

Офшоринг – це розширення діяльності вашої компанії (або частини вашої компанії) в іншій країні. Це зроблено з наміром заощадити витрати, мінімізувати навчання, отримати більше прибутку та отримати доступ до глобального ринку IT-індустрії з меншим наглядом, але повним контролем.

Відомим фактом є інформація про те. що зараз є велика кількість вихідного коду на старих мовах програмування, а саме таких як Pascal, Fortran, Modula.

Мови програмування застаріли та їх перестали використовувати та залишилась велика кількість вихідного коду на цих мовах. Цей код не має розвинутого інтерфейсу чи багатофункціональність але в цих кодах є реалізація математичних алгоритмів, високопродуктивних та новаторських підходів у реалізації математичних розв’язків рівнянь.

Незважаючи на те, що ці програми складалися досить давно, ідеї реалізації математичних теорій ніколи не старіють, особливо багато алгоритмічно-математичного вихідного коду використалося у вищих навчальних закладах.

Всі ці наробітки й рішення були заблоковані при старінні мови програмування.

Завдяки цьому розробка напрямку перекладу коду із застарілої мови програмування в більш новий завжди актуальна, код котрий надалі буде більш детально розглядатися та перероблятися під новий лад. Можливо також таке, що код після перетворення не буде у повній мірі робити але сама структура алгоритму залишиться.

У розробленій магістерській програмі використається переклад частин вихідного коду з мови програмування Pascal у Delphi 10.

На рисунку 3.1 зображена структурна схема системи де розглянута будова транслятора. При розробці схеми транслятора був проведений аналіз з теорії трансляторів [1-10].

З рисунку добре видно що транслятор розбито на декілька блоків а саме – блок лексичного аналізатора який взаємодіє з набором лексем мови програмування Pascal, проміжного машино-незалежного коду, синтаксичного аналізатора з синтаксисом мови Delphi 10 та генератора коду.

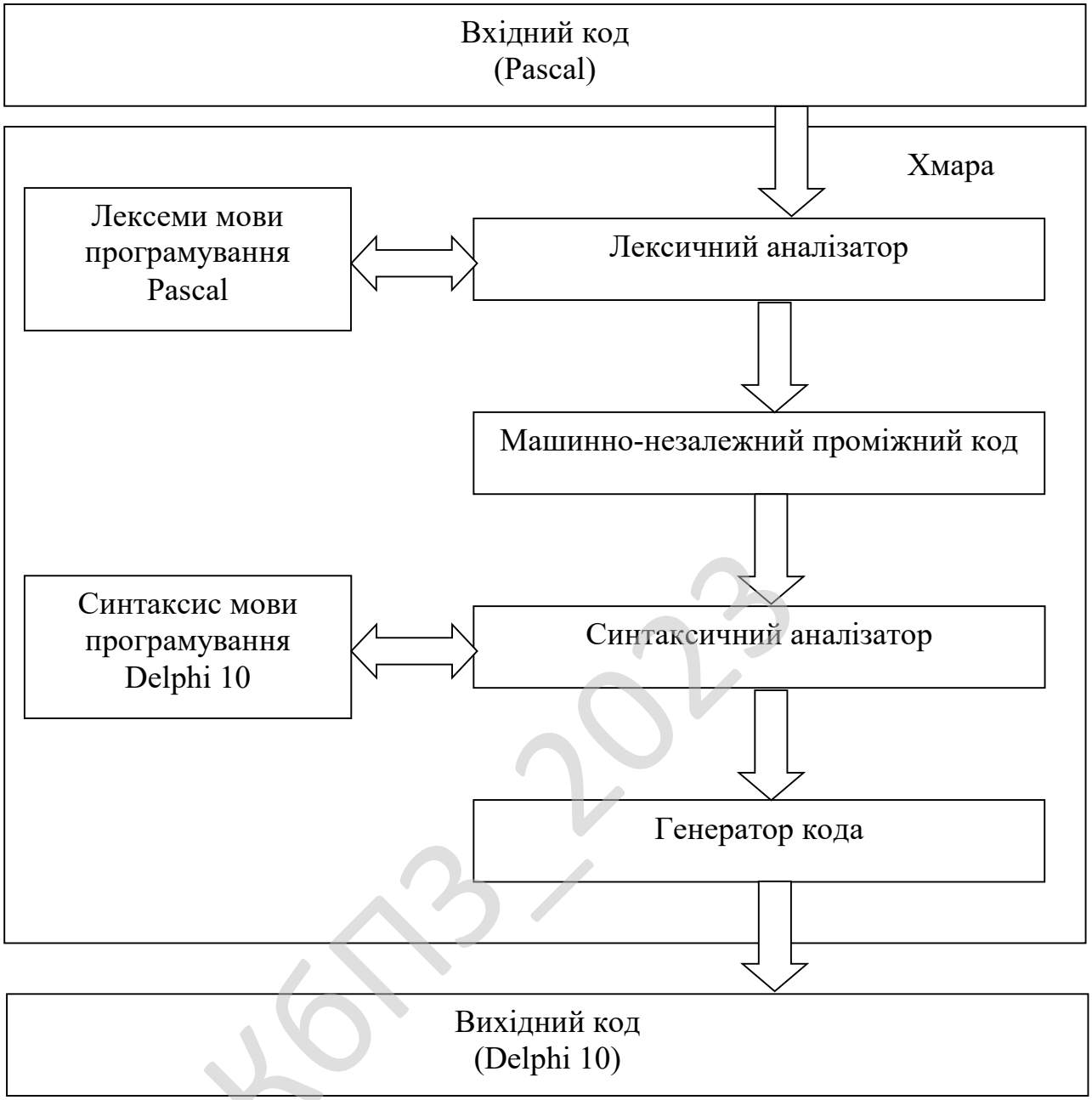




Рисунок 3.1 – Структурна схема системи

Детальний розгляд роботи цих компонентів розглянуто на функціональній схемі. На вхід транслятора поступає код Pascal, перед тим як запустити його у лексичний аналізатор (почати його оброблювати) проходить перевірка коду, що це дійсно є код Pascal. Ця дія виконується простою підстановкою шаблонного коду початку програми Pascal.

### 3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Після початку обробки коду Pascal проходить перевірка коду на відповідність (докладніше розглянуто у магістерських блок-схемах). Далі у лексичний аналізатор подається код та проводиться запуск лексичного аналізатора, обробка та одержання лексем, створення послідовності із всіх лексем коду та одержання машинно-незалежного проміжного коду. Далі відбувається запуск синтаксичного аналізатора котрий обробляє одержані фрагменти об'єктної моделі та створює послідовності із всіх фрагментів об'єктної моделі. Після проведення цих дій генератор коду проводить створення та виведення коду на екран.

Лексичний аналізатор виконує розпізнавання лексем мови і заміну їхніми відповідними кодами.Під лексемами розуміються елементарні одиниці, що входять у структуру пропозиції мови, такі як ключові слова, константи, імена і т.п. Правильність завдання структури речення мови на фазі лексичного аналізу не виконується. Синтаксичний аналізатор виконує перевірку правильності завдання пропозицій мови відповідно до граматики мови. Тільки при відсутності помилок можлива робота наступних модулів компілятора.

На фазі трансляції в проміжну форму виконується перетворення програми в матрицю проміжного кодові, де кожен рядок матриці містить у загальному випадку трійку – код операції і два операнда. Проміжні коди не мають прямих аналогів у системі машинних команд, тому дану форму представлення програми називають машинно-незалежною.

Фаза оптимізації призначена для зменшення надмірності програми по витратах годин. У залежності від критеріїв проектування транслятора дана фаза обробки програми може виключатися з циклові обробки програми.

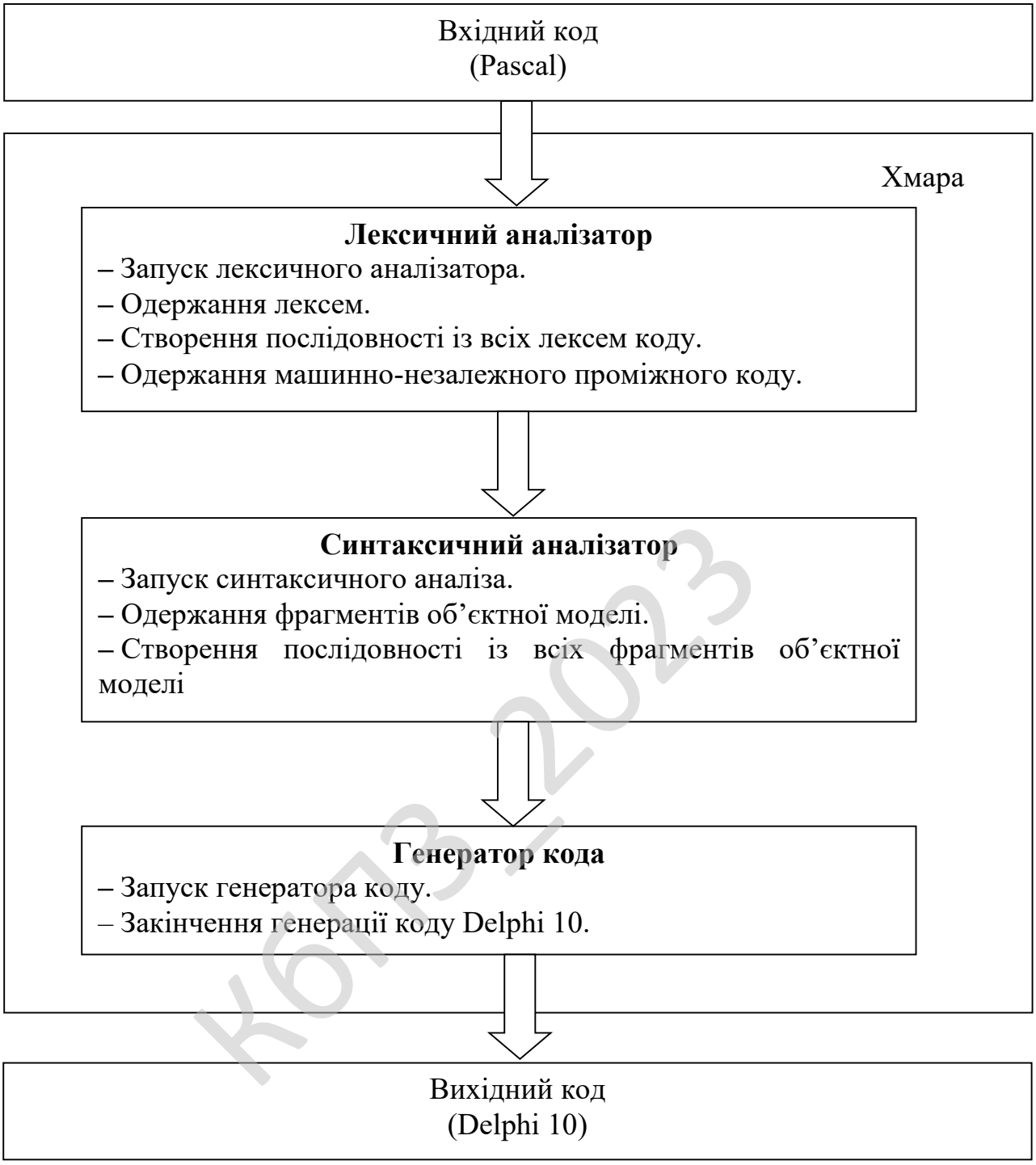


Рисунок 3.2 – Функціональна схема системи

На фазі розподілу пам'яті визначаються обсяги пам'яті, використовуваних у даній програмі, і вносяться коди резервування пам'яті в матрицю проміжного кодові. Кожна програма використовує принаймні два види пам'яті: статичну – для розміщення даних і кодів програми, і стекову – для звертання до підпрограм.

На фазі генерації кодові виконується підстановка кодових зразків вихідною мовою, що відповідають проміжним кодам програми. Крім того, на цій фазі обробки програми виконується також машинно-залежна оптимізація програми, що полягає в обліку особливостей архітектури даного комп'ютера.

Використання загальних регістрів процесора для збереження результатів проміжних обчислень дозволяє майже в два рази зменшити годино виконання програми.

### 3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

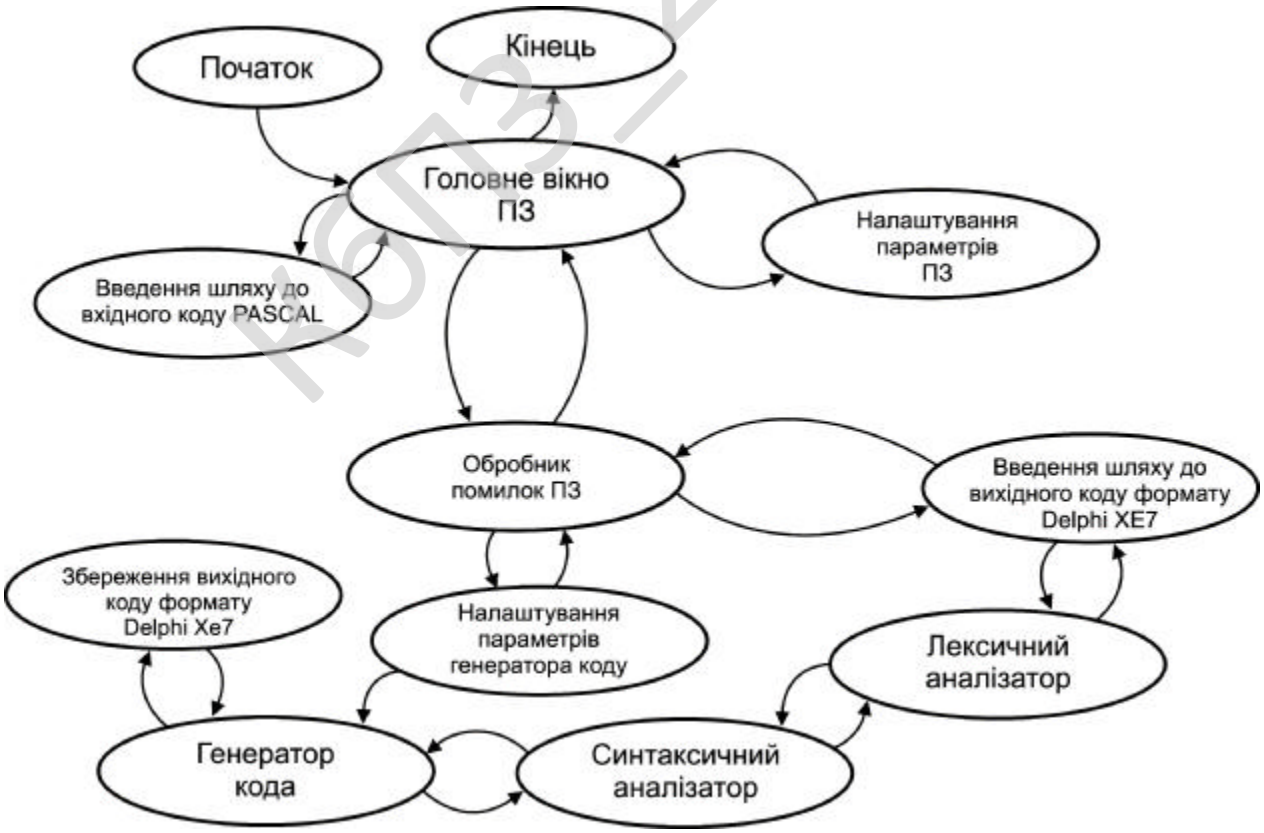

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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 ПРОЕКТУ. 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 зображено роботу підпрограми. З якої 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 підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

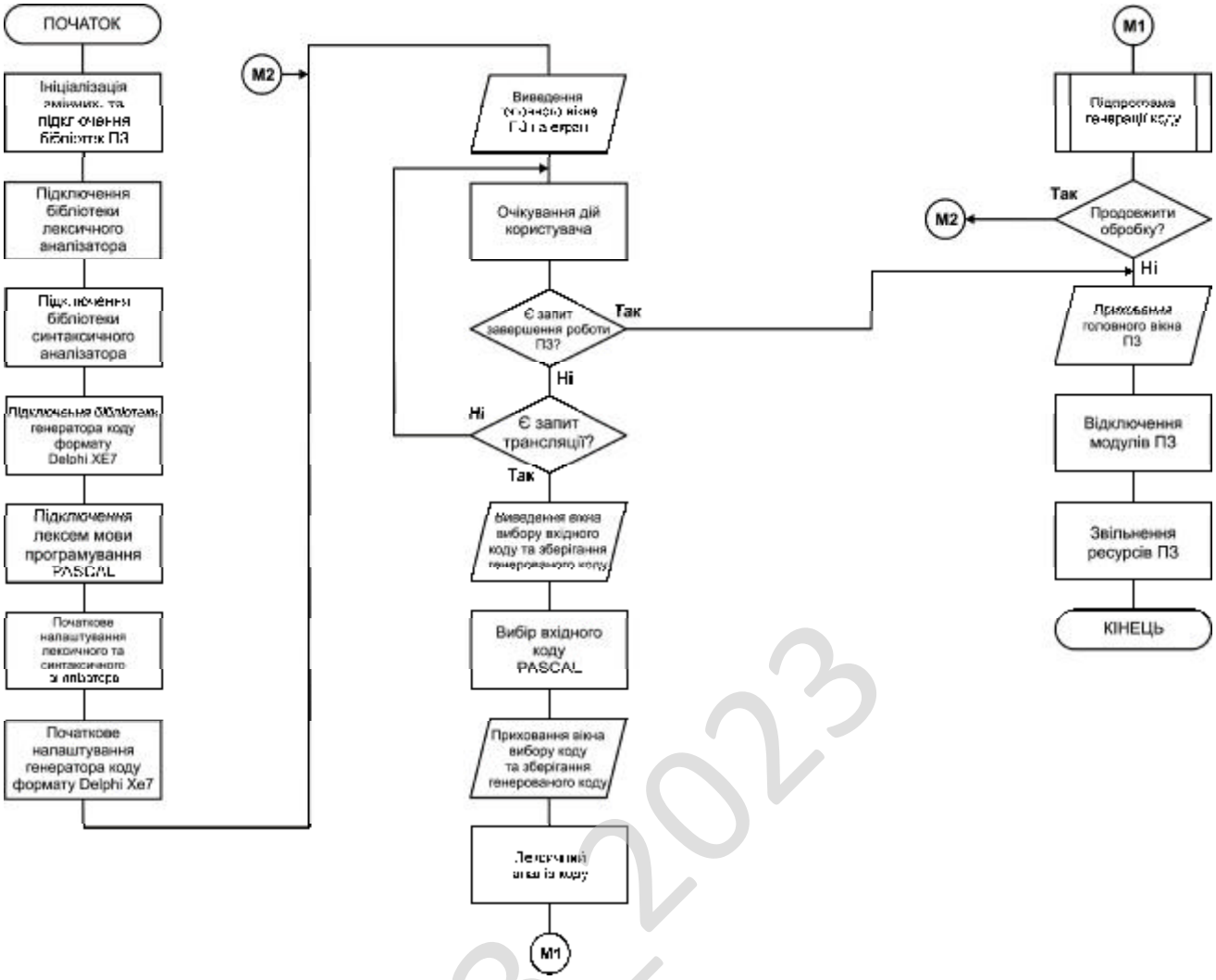


Рисунок 4.1 – Блок схема основної програми



UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

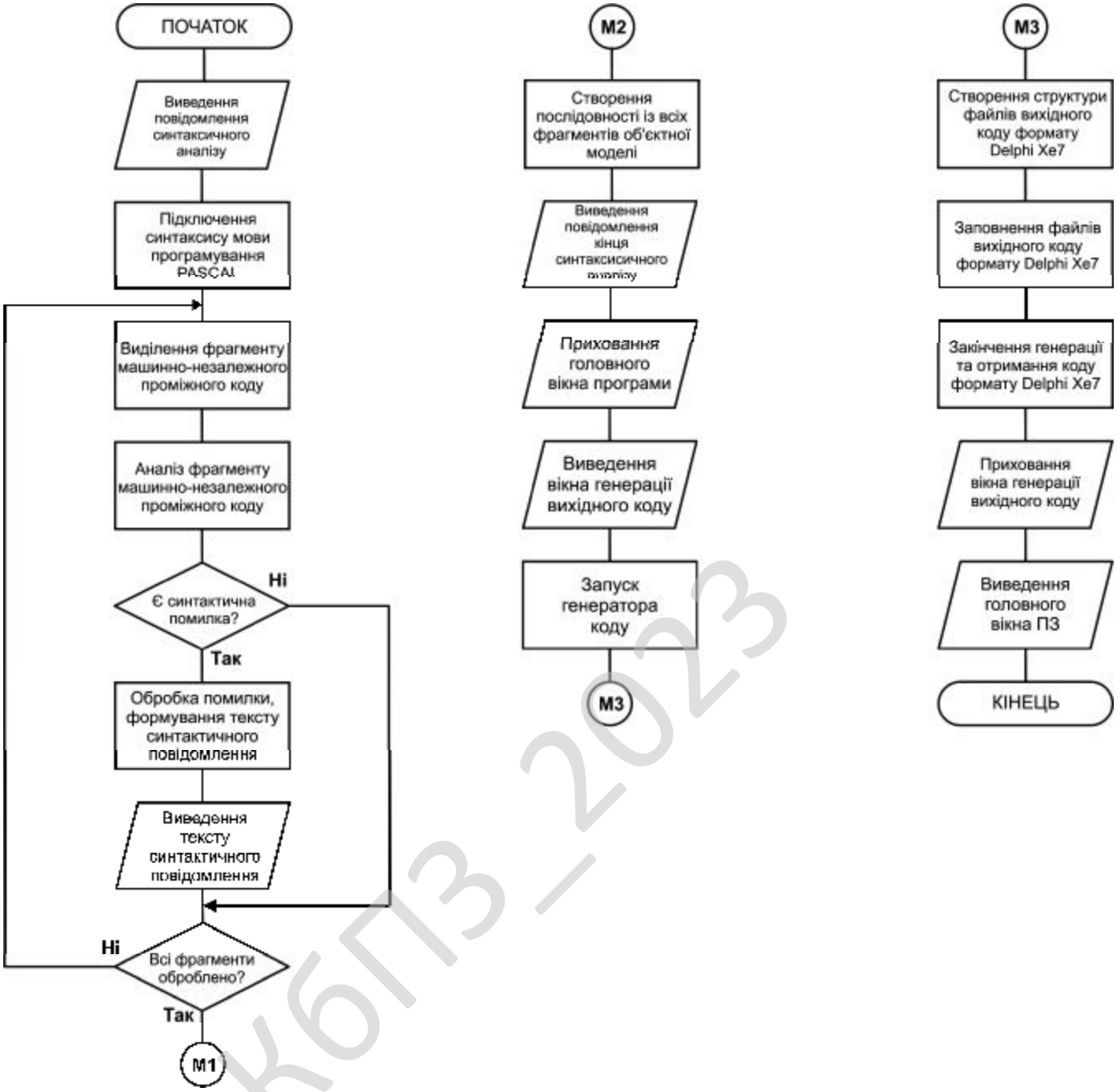




Рисунок 4.2 – Блок схема підпрограми генерації коду

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще

більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

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 та чорної скриньки.

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.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Першою основною задачею системи є розбивка вхідного рядка на лексеми.

Друга – заповнення відповідних таблиць. Найбільш відповідальною і важкою задачею є виділення лексем із вхідного рядка. Пропозиція вихідної програми складаються з лексичних одиниць, розділених символами-роздільниками.

У форматі DELPHI 10 символами-роздільниками є: , . ' ; : знаки операцій (), [].

Розпізнавання таких об'єктів ,як ідентифікатори і літерали виконуються на основі правил мови.

Після виділення лексеми з вхідного рядка спочатку виконується пошук у таблиці терміналів символів.

Якщо лексема належить МТС, то у вихідний рядок заноситися пари чисел:номер рядка в ММС і 0.

Наведемо приклад.

```
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. SUM-OF-PRICES.
3 AUTHOR. T-PRATT.
4 ENVIRONMENT DIVISION.
5 CONFIGURATION SECTION.
6 SOURCE-COMPUTER. SUN.
7 OBJECT-COMPUTER. SUN.
8 INPUT-OUTPUT SECTION.
9 FILE-CONTROL.
10 SELECT INP-DATA ASSIGN TO INPUT.
11 SELECT RESULT-FILE ASSIGN TO OUTPUT.
12 DATA DIVISION.
13 FILE SECTION.
14 FD INP-DATA LABEL RECORD IS OMITTED.
15 01 ITEM-PRICE.
16 02 ITEM PICTURE X(30)
17 03 PRICE PICTURE 9999V99
18 WORKING-STORAGE SECTION.
19 77 TOT PICTURE 9999V99. VALUE 0. USAGE IS COMPUTATIONAL.
20 01 SUM-LINE.
21 02 FILLER VALUE ' SUM -'PICTURE X(12).
22 02 SUM-OUT PICTURE $$.$$$.$$9.99.
23 03 COUNT-OUT PICTURE ZZZ9.
```

Якщо лексема не виявлена в ММС, те вона класифікується як можливий ідентифікатор.

Після того, як лексема класифікується як можливий ідентифікатор, виконується пошук у таблиці ідентифікаторів. Якщо така лексема відсутня в МІ,те створений новий елемент МІ для цього ідентифікатора і номер рядка стані специфікатором у коді лексеми.

Якщо є, то новий рядок не утвориться, а формується код лексеми.

25 PROCEDURE DIVISION.

26 START.

27 OPEN INPUT INP-DATA AND OUTPUT RESULT-FILE.

28 READ-DATA.

29 READ INP-DATA AT END GO TO PRINT-LINE.

30 ADD PRICE TO TOT.

31 ADD 1 TO COUNT.

32 MOVE PRICE TO PRICE-OUT.

33 MOVE ITEM TO ITEM-OUT.

34 WRITE RESULT-LINE FROM ITEM-LINE.

35 GO TO READ-DATA.

36 PRINT-LINE.

37 MOVE TOT TO SUM-OUT.

Лексичний аналіз виробляється або за один повний перегляд вихідної програми, або шляхом виклику лексичного аналізатора щораз, коли для синтаксичного аналізу необхідне розпізнавання черговий лексичної одиниці.

39 CLOSE INP-DATA AND RESULT-FILE.

40 STOP RUN.

### 4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 9041:2020. Його повна назва: ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

Цей алгоритм призначений для шифрування коротких (до 616 біт) повідомлень для будь-яких алгоритмів шифрування, в тому числі визначених національними стандартами України.

Як і стандарт цифрового підпису ДСТУ 4145:2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейєрштрасса найновітніші розробки у галузі еліптичної криптографії – криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази.Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів. Так, нижня межа стійкості криптосистем у цьому стандарті дорівнює 2127 (≈1042) (це більш ніж у півтора рази вище, ніж у ДСТУ 4145) і можуть бути обрані інші рівні, такі як 2255 (≈1085), 2383 (≈10127) та 2767 (≈10255); крім того, строго обґрунтована його стійкість як до атак на відновлення відкритого тексту, так і до розрізнюючих атак.

Проект алгоритму шифрування, що ліг в основу цього стандарту, пройшов апробацію як в Україні, так і за її межами (Центрально-Європейська конференція з криптографії (червень 2020 року) – форум ведучих криптологів з усього світу).

Стандарт ДСТУ-9041 узгоджений з усіма діючими в Україні національними стандартами. Новиною стандарту є його сфера застосування – інкапсуляція ключів, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145, використовує виключно національні криптографічні алгоритми національних стандартів та не містить

посилань на відповідні пост-радянські стандарти, термін дії яких вже практично вичерпався.

Новий стандарт не належить до так званих постквантових стандартів. Але його стійкість буде під загрозою лише тоді, коли з’являться квантові комп’ютери з 700 і більше кубітами (на даний час кількість "робочих" кубітів, які вдалося створити, – близько 50). Його перевагою перед постквантовими алгоритмами є відносно невелика довжина ключа (у десятки або навіть у сотні разів менша, ніж у постквантових алгоритмах).

# 5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Блок меню.

– Блок вибору директорій пошуку коду на мові Pascal.

– Блок вибору директорій зберігання конвертованого коду у формат Delphi 10.

– Блоків перегляду таблиць перетворення та налаштування хмарних сервісів.

– Поле виведення знайденого коду чи ручного введення коду на мові Pascal.

– Поле виведення конвертованого коду у формат Delphi 10.

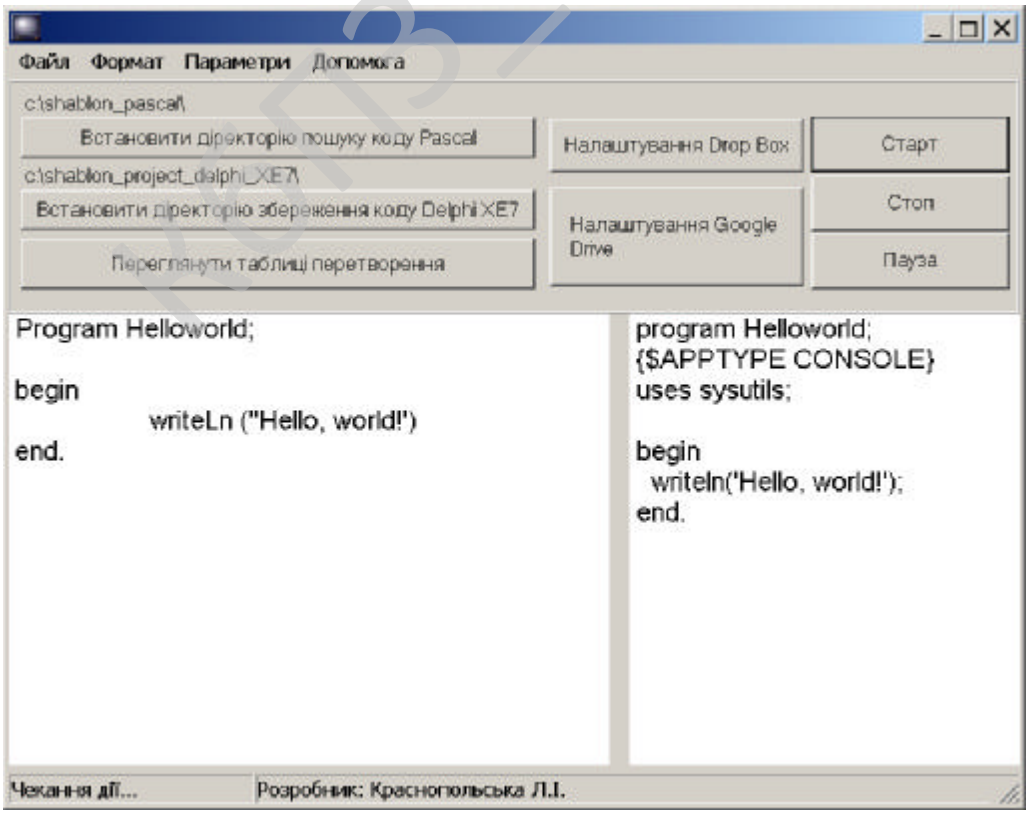


Рисунок 5.1 – Головне вікно ПЗ

Навігаційне меню, складається з розділів: Файл; Формат; Параметри; Допомога. Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

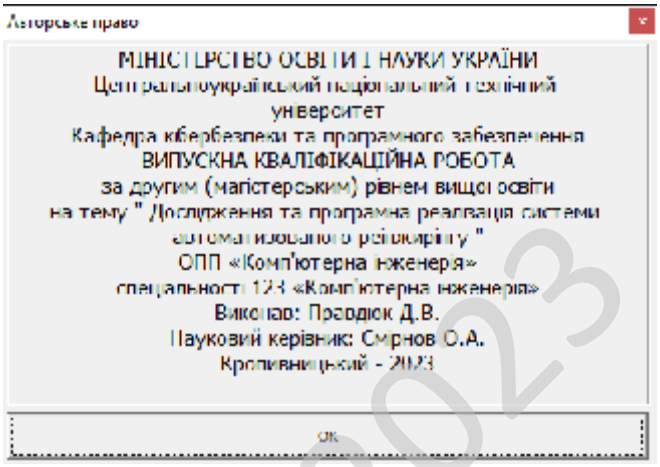


Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Безплатне програмне забезпечення можна безоплатно встановлювати та, в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

# 6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи автоматизованого реінжирінгу.

*Метою розробки* є дослідження та програмна реалізація системи автоматизованого реінжирінгу.

*Об'єктом дослідження* є процес автоматизованого реінжирінгу.

*Предметом дослідження* є методи автоматизованого реінжирінгу.

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

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

– Удосконалено метод автоматизованого реінжирінгу.

– Розроблено вітчизняний продукт автоматизованого реінжирінгу, 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 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи автоматизованого реінжирінгу.

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 | 57 |
| 3. Запланований термін розробки, днів | Fpq | 48 (2 місяці) |
| 4. Група задачі підсистеми управління (1-6) | – | 1 |
| 5. Ступінь новизни задачі (А, Б, В, Г) | – | Б |
| 6. Складність алгоритму (1, 2, 3) | – | 2 |
| 7. Кількість макетів вхідної інформації | – | 3 |

Продовження таблиці 7.1

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

Продовження 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 | – | 60000 |
| 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 А – коефіцієнт Боема, А=2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

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

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

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

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

$$T_{уточн}=T_{ном}\, ПV_j, \tag{7.3}$$

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

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

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

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

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

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

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot65=109 \text{ люд/день}$$

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 | 109 | Ф 7.1-7.4 |
| Впровадження | 13 | Д13 |
| Всього | 150 | – |

### 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,

$$Ч = \frac{150 \cdot 1}{48\text{-}5} = 3,5 \text{ ставки}$$

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 | 6 | 540 | 9 |
| Монітор | 60 | 6 | 360 | 6 |
| Клавіатура | 30 | 6 | 180 | 3 |
| Маніпулятор «мишка» | 30 | 6 | 180 | 3 |
| 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 | 1 | 30 | 0,5 |
| Кабельні господарства ЛОМ на 1 м.п. | 2,5 | 140 | 350 | 5,83 |
| Копіювальний апарат | 140 | 1 | 140 | 2,33 |
| Усього за рік: | | | $З_ч$ | 32,99 |

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{33 \cdot 3}{1,2} = 82,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=82,5/(48·8)=0,2 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 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi | 0,8 | 0,2 |
| | Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS) | 0,2 | |
| | Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ | 0,2 | |
| | Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет | 0,4 | |
| Всього | | 1,6 | |

Продовження таблиці 7.4

| Посада | Вид роботи | Час | Кількість штатних одиниць |
|---|---|---|---|
| Продакт-менеджер | Презентації нової продукції, пошук каналів збуту | 2 | 0,5 |
| | Підтримка постійних клієнтів | 1 | |
| | Оформлення договорів, ведення тендерів | 0,5 | |
| | Контроль взаєморозрахунків з постачальниками | 0,5 | |
| Всього | | 4 | |
| Дизайнер WEB | Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію | 0,5 | 0,2 |
| | Створення графічних і стилістичних елементів сайту | 0,5 | |
| | Оформлення банерів і промо-сторінок | 0,3 | |
| | Розміщення графіки і контенту на Інтернет сторінках | 0,3 | |
| Всього | | 1,6 | |
| Інженер верстальник | Розробка та верстка макетів рекламної продукції та технічної документації | 1 | 0,2 |
| | Верстка друкованих видань | 0,2 | |
| | Додрукова підготовка макетів | 0,2 | |
| | Розміщення графіки і контенту на Інтернет сторінках | 0,2 | |
| Всього | | 1,6 | |

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

| Посада | Кількість ставок | Середньо-місячний оклад, грн. | Всього за період роз-робки, грн. |
|---|---|---|---|
| Керівник (IT-менеджер) | 1 | 19103 | 38206 |
| Продакт-менеджер | 0,5 | 17000 | 17000 |
| Інженер-програміст | 3,5 | 20000 | 140000 |
| Інженер-електронщик | 0,2 | 15000 | 6000 |
| Інженер-системотехнік | 0,2 | 15000 | 6000 |
| Адміністратор мережі | 0,2 | 11000 | 4400 |
| Системний програміст | 0,2 | 9000 | 3600 |
| Дизайнер WEB | 0,2 | 9000 | 3600 |
| Інженер-верстальник | 0,2 | 9000 | 3600 |
| Бухгалтер-економіст | 0,2 | 10000 | 4000 |
| Всього за період розробки | $R_{сп}$=6,4 | - | $Ф_{роб}$=226406 |

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{226406}{6,4 \cdot 48} = 737 \; грн.$$

## 7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць.

$S_у$ – питома площа на одне робоче місце, $м^2$,

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

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

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

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

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

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

де Цм – 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 комерційною пропозицією фірми Brain за 4.11.23 – джерело http://brain.com.ua/

Таблиця 7.6 – Специфікація

| Найменування комплектуючої або обладнання | Тип | Оптова ціна |
|---|---|---|
| Персональний комп’ютер | | 11771 |
| Системний блок | | 7771 |
| Процесор | INTEL Core™ i3 10105F (BX8070110105F 1200, 4 ядра, 8 потоків, 3.7 GHz, 4.4 GHz TDP – 65 Вт, 14nm | - |
| Системна плата | GIGABYTE H510M K V2 сокет – 1200 чипсет – Intel H470, DDR4 | - |
| Жорсткий диск | HDD 500 Gb SAMSUNG Barracud HD502HJ (3.5", 500ГБ, 16МБ, SATA II-300) | - |
| Оперативна пам’ять | DDR4 8GB 3200 MHz Dat (DT8G4DLDND32) DDR4, 8 ГБ, В наборі 1 | - |
| Відеоадаптер | Radeon R7 350 2Gb Afox PCI-Express 3.0, ГБ, GDDR5, 128 Bit | - |
| Корпус | Logicpower 8702 – 550w 12cm | - |
| інше | Клавіатура, мишка | - |
| Монітор | Монітор BenQ GL2450HM Black | 2600 |
| Принтер лазерний | Canon i-SENSYS LBP6030W | 2700 |
| Принтер струминний | Epson Stylus Photo P50 (C11CA45341) + USI cable | 5500 |
| Сканер | Epson Perfection V37 | 2800 |
| Копіювальний апарат | Canon i-SENSYS MF217W with Wi-Fi | 5965 |
| Пристрій безперебійног 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 | - | - |
| 5. Транспортні засоби | 143000 | - | |
| 6. Господарський інвентар | 28000 | - | - |
| Всього по групі | 176190 | 20 | 35238 |
| 7. Нематеріальні активи | 60000 | 10 | 6000 |
| Разом | $К_р$= 1769406 | | $А_р$= 174246 |

Примітка: вартість автомобіля Chevrolet Aveo 2008 взята за даними сайту «Авто-Ріа», джерело https://auto.ria.com/uk/auto_chevrolet_aveo_32795647.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.

$$З_о = 737 \cdot 150/57 = 1940 \text{ грн}$$

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, %

$$З_д = 1940 \cdot 10 \cdot 0{,}01 = 194 \text{ грн}$$

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

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

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(1940+194) = 470 \text{ грн}$$

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

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1940 \cdot 15 \cdot 0{,}01 = 291 \text{ грн}$$

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

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

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

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

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):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,6 грн./шт., DVD-R box – 46,6 грн./шт.

$$З_{М2} = 46,6 \cdot 10 = 466 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої становить:

$$З_{М3} = \sum Ц_з ., \quad (7.18)$$

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.

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

$$З_М = (80 + 466 + 1702)/57 = 39 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0,01, \quad (7.19)$$

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

$$О_п = 1940 \cdot 15 \cdot 0,01 = 291 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програ ($N_е$ = 57 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

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

$$А_м = 174246 \cdot 2/(57 \cdot 12) = 509 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтям калькуляції

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

$$C_п = 1940+194+470+291+39+291+509=3734 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де $Р_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 3734 = 1867 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмног забезпечення задачі

| Найменування статей витрат | Позначення | Величина, грн. |
|---|---|---|
| 1 | 2 | 3 |
| 1. Основна зарплата виконавців | $З_о$ | 1940 |
| 2. Додаткова зарплата виконавців | $З_д$ | 194 |
| 3. Відрахування на соціальні потреби | $С_{оц}$ | 470 |
| 4. Загальногосподарські витрати | $Г_{осп}$ | 291 |
| 5. Витрати на матеріали | $З_М$ | 39 |
| 6. Освоєння нових операційних систем, мов програмування | $О_п$ | 291 |
| 7. Амортизація основних фондів | $А_м$ | 509 |
| 8. Повна собівартість програмного забезпечення | $С_п$ | 3734 |
| 9. Плановий прибуток | $П_р$ | 1867 |
| 10.Ціна підприємства $Ц_п=С_п+П_р$ | $Ц_п$ | 5601 |
| 11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$ | $ПДВ$ | 1120,2 |
| 12.Відпускна ціна програмної продукції Ц = Цп + ПДВ | $Ц$ | 6721,2 |

## 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 | – | 6721 |
| Всього капітальних витрат | – | 6721 |

## 7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

| Найменування статей витрат | Позна-чення | Сума витрат за варіантами, грн. | |
|---|---|---|---|
| | | Базовий | Новий |
| 1. Витрати на технічне обслуговування | $З_р$ | 33550 | 26840 |
| 2. Витрати на електроенергію | $З_{ел}$ | 4608 | 4224 |
| 3. Витрати на амортизацію | $З_{ам}$ | 0 | 3361 |
| Всього витрат за рік | $І$ | 38158 | 34425 |

Витрати на технічне обслуговування:

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

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*,

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 250 годин на рік до 200 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 250 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 33550\ грн.$$

до

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

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

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

Зел баз=0,2·7200·3,2=4608 грн

Зел нов=0,2·6600·3,2=4224 грн

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

| Групи основних фондів | Норма амортизації % | Балансова вартість, грн., за варіантами | | Сума відрахувань, грн за варіантами | |
|---|---|---|---|---|---|
| | | Базовий | Новий | Базовий | Новий |
| Програмна продукція | 50 | – | 6721 | – | 3360,5 |
| Всього відрахувань | - | – | 6721 | – | 3360,5 |

## 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.

$$Ев = (5601 - 3734) \cdot 57 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 77378 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: $K_р^*$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1769406}{(5601 - 3734) \cdot 40 \cdot 12 / 2} = 3,9 \text{ років}$$

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

$$E_{сп} = (38158\text{-}34425)\text{-}0,5 \cdot 6721 = 373 \text{ грн.}$$

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

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

$$T_{сп} = \frac{6721}{38158 - 34425} = 1{,}8 \text{ років}$$

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. | 57 |
| 2. Повна собівартість розробленої програми | Грн. | 3774 |
| 3. Ціна розробленої програми | Грн. | 5601 |
| 4. Плановий прибуток від реалізації розробленої програми | Грн. | 1827 |
| 5. Рентабельність програмної продукції | % | 50 |
| 6. Об'єм додаткових капітальних вкладень у виробника програмної продукції | Грн. | 1769406 |
| 7. Загальний прибуток від реалізації програмної продукції | Грн. | 104139 |
| 8. Величина економічного ефекту при виготовлені програмної продукції | Грн. | 77378 |
| 9. Період окупності додаткових капітальних вкладень у виробника програмної продукції | Років | 3,9 |
| 10.Об'єм додаткових капітальних вкладень у споживача програмної продукції | Грн. | 6721 |
| 11.Величина економічного ефекту у користувача програмної продукції | Грн. | 373 |
| 12.Період окупності додаткових капітальних вкладень у користувача програмної продукції | Років | 1,8 |

## 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

Техніка безпеки – це система правил і заходів, які допомагають запобігти травмам, хворобам і аваріям на робочому місці або в повсякденному житті. Знання техніки безпеки дуже важливе, бо воно рятує життя і здоров'я людей. Наприклад, якщо людина працює на шахті, то вона повинена знати правила безпеки у вугільних шахтах, щоб не потрапити під обвал або не підірватися на міні. Або якщо людина хоче навчитися програмувати, то вона повинена дотримуватися гігієнічних вимог і не сидіти за комп'ютером занадто довго, щоб не зашкодити своїм очам і спині.

Охорона праці та здоров'я у сфері ІТ – це комплекс заходів, які спрямовані на забезпечення безпечних і здорових умов праці для працівників, які використовують інформаційні технології, а також на запобігання травматизму, професійним захворюванням і стресу.

Охорона праці та здоров'я у сфері ІТ включає такі напрями, як:

– Ергономіка – це наука про адаптацію робочого середовища до фізичних і психологічних особливостей людини[2]. Ергономіка вимагає врахування таких факторів, як розміри, форми, кольори, освітлення, шум, температура, вологість, вентиляція, постава, рухи, пози, втома, навантаження на очі та ін. Егономіка допомагає покращити комфорт, продуктивність і задоволення працівників.

– Комп'ютерна безпека – це захист комп'ютерних систем і даних від несанкціонованого доступу, зміни, знищення або блокування. Комп'ютерна безпека вимагає використання антивірусних програм, фаєрволів, паролей, шифрування, резервного копіювання та інших технологій. Комп'ютерна безпека допомагає запобігти крадіжці, шпигунству, шантажу, саботажу та іншим загрозам.

– Соціальна взаємодія – це процес спілкування між людьми у робочому колективі або через мережеві сервіси. Соціальна взаємодія вимагає дотримання

правил етикету, поваги, толерантності, співробітництва та конструктивного діалогу. Соціальна взаємодія допомагає покращити настрій, мотивацію, комунікацію та творчий потенціал працівників.

Правила охорони праці і здоров'я для програмістів:

– Регулярно роби перерви в роботі. Вставай із-за столу і розминай м'язи.

– Налаштуй яскравість і контрастність монітору так, щоб не напружувати очі.

– Використовуй ергономічну мишку і клавіатуру, які зручно лягають у руку і не виклкають болю.

– Слідкуй за своєю поставою. Сиди прямо і не нахиляйся до екрану.

– Захищай свій комп'ютер від вірусів, шпигунських програм і хакерів. Оновлюй ативірусне програмне забезпечення і не відкривай підозрілі файли і посилання.

– Не забувай про соціальну взаємодію. Спілкуйся з колегами, друзями і родиною. – Відвідуй заходи, які тебе цікавлять. Не ізолюй себе від світу.

– Люби свою професію, але не забувай про інші сфери життя. Розвивай свої захопленя, хоббі і таланти. Знаходь рівновагу між роботою і відпочинком.

Закон України “Про охорону праці” визначає основні принципи, завдання, права і обов’язки суб’єктів відносин з охорони праці, а також організаційні та правові основи державного управління і контролю за дотриманням законодавства про охорну праці.

Згідно з цим законом, ІТ компанії повинні впроваджувати такі заходи з охорни праці:

– Створювати на підприємстві службу охорони праці або призначати відповідальних осіб, які забезпечують розроблення, реалізацію та контроль за дотриманням заходів з охорони праці.

– Забезпечувати безпечні і нешкідливі умови праці для працівників, використвючи сучасні засоби техніки безпеки, санітарно-гігієнічні умови, засоби клектинго та індивідуального захисту, оптимальні режими праці та відпочинку.

– Проводити атестацію робочих місць на відповідність нормативно-правовим атам з охорони праці та аудит з охорони праці.

– Проводити навчання та інструктаж з питань охорони праці, з надання першої медичної допомоги потерпілим від нещасних випадків і правил поведінки у разі виникнення аварії[5].

– Забезпечувати лікувально-профілактичне обслуговування працюючих, санітарно-побутове обслуговування, пільги і компенсації для працівників, які працюють у важких і шкідливих умовах.

– Нести відповідальність за порушення законодавства про охорону праці та зподіяння шкоди життю і здоров'ю працівників.

## 8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Це дуже важливі питання, адже робота з комп'ютером може впливати на здров'я людини. За ДСанПіН 3.3.2.007-98, шкідливі і небезпечні фактори при роботі з комп'ютером можуть бути такі:

– Електромагнітне випромінювання – це випромінювання, яке створюється комп'ютером і його периферійними пристроями, такими як монітор, пртер, сканер тощо. Це випромінювання може викликати головний біль, запаморчення, розлади сну, зниження імунітету та інші негативні ефекти.

– Електростатичне поле – це поле, яке утворюється внаслідок накопичення електричних зарядів на поверхні комп'ютера і його частин. Це поле може спричиняти сухість шкіри, свербіж, подразнення очей, алергічні реакції та інші проблеми.

– Нервово-емоційне напруження – це напруження, яке виникає внаслідок трвлої концентрації уваги, високої вимогливості до результату роботи, нестабільності програмного забезпечення, конфліктних ситуацій тощо. Це напруження може призводити до стресу, депресії, роздратування, погіршення пам'яті та інших порушень.

– Навантаження на органи зору – це навантаження, яке виникає внаслідок догго перебування перед монітором, низької якості зображення, недостнго освітлення приміщення, поганої ергономіки робочого місця тощо. Це навантаження може спричиняти зниження гостроти зору, появу блимавок і кругів перед очима, синдром сухого ока та інші захворювання.

– Дрібні стереостатичні рухи кінцівок – це рухи, які пов’язані з керуванням клавіатурою і мишею. Ці рухи можуть призводити до перевантаження і запалення суглобів і сухожиль рук і пальців, а також до тендовагінітів і синдрому зап’ясткового каналу.

Таким чином, робота з комп’ютером не така безпечна, як може здатися на перший погляд. Тому дуже важливо дотримуватися правил охорони праці і гігієни при роботі з комп’ютером.

### 8.3 Аналіз умов праці на робочому місці фахівця

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 4,4 м×6,2 м×2,9 м. Одна з її більших стін має шість двостулкових вікон, розмірами 2 м×1,8 м, які виходять на північний захід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита леноліумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8 м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1255 мм×845 мм. Висота столів 760 мм. Висота стільців від рівня підлоги становить 430 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 [4] 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 | 22 – 24°С |
| | Відносна вологість | 40 – 60% |
| | Швидкість руху повітря | до 0,1 м/с |
| Теплий | Температура повітря в приміщенні | 23 – 25°С |
| | Відносна вологість | 40 ... 60% |
| | Швидкість руху повітря | 0,1 ... 0,2 м / с |

**Розробка заходів з умов поліпшення охорони праці**

Провівши аналіз умов праці в розглянутому вище приміщенні, було отримано наступні результати:

– значення мікроклімату в приміщенні не перевищує норму;

– розрахунки розміру робочого місця на одного працівника відповідають нормі;

– рівень шуму в приміщенні не становить вище норми.

З вище перечислених результатів можна зробити висновок, що основний вплив на продуктивність ІТ-спеціалістів є його психологічний стан. Тому є доцільним зменшити рівень стресу на робочому місці.

Рекомендовані наступні заходи: За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м. Конструкція робочого місця користувача персонального комп'ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм). Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим [7].

### 8.3 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *80·80·8 мм*., (згідно з ДСТУ 2251:2018 «Кутки сталеві гарячекатані рівнополичні. Сортамент») довжиною *L=1,6 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,5 м*. Відстань між вертикальними заземлювачами (електродами) *A=1,6 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:

$$T=t+L/2=0,6+1,6/2=1,4 \text{ м.}$$

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 [6];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [7].

Еквівалентний діаметр вертикальньного електрода (кутка) [7]:

$$Dв=0,95 \cdot K = 0,95 \cdot 80 = 76,85 \text{ мм.} = 0,076 \text{ м.}$$

де *K= 80 мм.* – розмір полиці металевого кутка.

Відношення $A/L=1,6/1,6=1$

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [7]:

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

$$=0,366(54,5/1,6)[(lg(2\cdot1,6/0,076)+(1/2\ )lg((4\cdot1,4+1,6)/(4\cdot1,4-1,6))]=$$

$$=21,8\ Ом.$$

Визначаємо коефіцієнт екранування вертикальньних електродів $Кев=0,8$ при орієтовній кількості вертикальньних електродів, яке дорівнює 5 [7].

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

$$N=R_O\ /(Кев\ R_{ЗН})= 21,8\ /\ (0,8\cdot4)= 6,8 \approx 7\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot1,6\cdot7=12,2 \approx 12\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [7]:

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

$$=0,366(40\cdot5/12)\cdot lg((2\cdot12^{\,2}\ )/(0,06\cdot0,6\ ))=20,16\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідной климатичной зони для з'єднуючої полоси [7]:

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

Загальньний опір розтіканню електричного струму заземлювача [7]:

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

$$=(21,8\cdot\ 20,16\ )/(21,8\cdot\ 0,75+\ 9\cdot\ 20,16\ \cdot\ 0,8)=3,46\ Ом.$$

*де* $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [7].

Умова $R \le R_{ЗН}$ виконується $(3,46 \le 4)$.

Так як при *7* вертикальних електродах $R$ суттево менше $R_{ЗН}$, зменшимо кількість вертикальних електродів $N$ до *6* і виконаємо перерахунок. У результаті остаточно отримали: $R =3,96\ Ом.$ при кількості вертикальних електродів $N=6$.

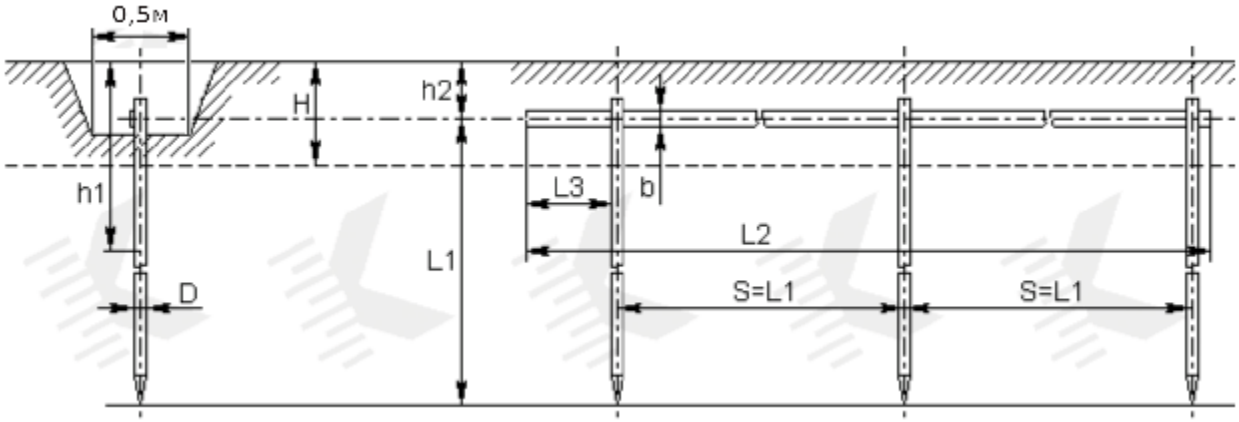


Рисунок 8.1 – Схема штучного заземлення.

### 8.5 Висновки до розділу

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

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.

**Список використаних джерел інформації**

1. Державні будівельні норми України ДБН В.2.5 – 28 – 2006: Інженерне обладнання будинків і споруд. Природне і штучне освітлення. – *Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0168667-06#Text* (дата звернення 19.10.22).

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – *Режим доступу до ресурсу:* https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення 19.10.22).

3. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – *Режим доступу до ресурсу:* https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення 19.10.22).

4. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – *Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99* (дата звернення 19.09.22).

5. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.22).

6. Охорона праці. Ч. 1. Захисне заземлення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд. ; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград : КІСМ, 1997. – 20 с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення 19.09.22).

7. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : "Колос", 1973. – 238 с.

# 9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи автоматизованого реінжирінгу.

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 автоматизованого реінжирінгу.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем автоматизованого реінжирінгу.

– Досліджена система автоматизованого реінжирінгу.

– На основі отриманих результатів досліджень створена програмна реалізація системи автоматизованого реінжирінгу.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання автоматизованого реінжирінгу.

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 Rescueware Workbench, Delphi. 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 9041:2020.

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 373 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,8 роки.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Правдюк Д.В. Дослідження та програмна реалізація системи автоматизованого реінжирінгу // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Adi-Tabatai A., Ciernak M., Lueh G., et al. Fast, Effective Code Generation in a Just-In-Time Java Compiler // Proceedings of the ACM SIGPLAN'98 Conference on Programming Language Design and Implementation, P. 280-290.

3. Aho A., Sethi R., Ullman J. Compilers Principles, Techniques, and Tools //Addison-Wesley, 1988 31

4. Davis T. Build your own ObjectPool in Java to boost app speed. // In JavaWorld, June 1997. см. http://www.javaworld.com/javaworld/jw-06-1998/jw-06-object-pool.html

5. Jacobson I. Re-engineering of old systems to an object-oriented architecture // In:OOPSLA'91, ACM Press, 1991, P. 340-350

6. Lowe D. The CICS Programmer's Desk Reference // Mike Murach & Associates, Inc,1992

7. Muchnic S. Advanced Compiler Design and Implementation // Morgan Kaufmann Publisher, Inc. 1997.

8. Glass R.L. Pascal-A Contradiction and an Enigma // Communication of the ACM. – 1997. -V.40, № 9. -P. 11-13.

9. Gosling J., Joy B., Steele G. The Java Language Specification // Addison-Wesley Pub Co, 1996

10. Graham J. Migrating To Object Technology //Addison-Wesley, 1995

11. Holub A. Programming Java Threads in real world, part I. // In JavaWorld, Sept.1997. см. http://www.javaworld.com/javaworld/jw-09-1998/jw-09-threads.html

12. Markosin L., Newcomb P., Brand R., et al. Using an Enabling Technology to Reengineer Legacy Systems // Communications of the ACM.- 1994.- V.37,№5.- P. 58-70.

13. Műller H., Wong K., Tilley S. Understanding Software Systems Using Reverse Engineering Technology // In Proc: Colloquium on Object Orientation in Databases and Software Engineering; The 62nd Congress of ACFAS, 1994

14. Olsem M., Sittenauer C. Reengineering Technology Report V.1. – P. 20-23 // Software Technology Support Center, August 1993 – см. http://www.stsc.hill.af.mil/RENG/defin.html

15. Ragget D., Le Hors A., Jabocs I. –editors. HTML 4.0 Specification // W3C Recommendation, 1998, см. http://www.w3.org/TR/REC-html40/

16. Tilley S.R., Smith D.B. Legacy System Reengineering // 8th International Workshop on Software Technology and Engineering Practice (STEP97: July 15, 1997; London, UK) – см. http://www.sei.cmu.edu/reengineering/pubs/step97/tutorial/

17. Quilici A. Reverse Engineering: A Path Towards Success // ACM Communications, ICSE'95 P. 333-336

18. 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.

19. 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.

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

21. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools».

Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

22. 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.

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

24. 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.

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

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

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

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

33. 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.

34. 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.

35. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС,

важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

36. 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.

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

38. 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.

39. 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.

40. 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.

41. 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.

42. 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).

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

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

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

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

**Технічне завдання**

**Зміст**

## 1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи автоматизованого реінжирінгу.

## 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 автоматизованого реінжирінгу.

## 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 автоматизованого реінжирінгу;

– 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 Rescueware Workbench, Delphi.

### 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 повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

## 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 – 91 аркуш.

## 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

_____ Смірнов О.А.



***Дослідження та програмна реалізація системи автоматизованого реінжирінгу***

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 40

Літера: РП

Кропивницький – 2023 року

**Файл проекту DPR**

```
program Cloud_Network_Reingeniring;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2};
  Unit3 in 'Unit3.pas' {Form3};
  Unit4 in 'Unit4.pas' {Form4};
  Unit5 in 'Unit5.pas' {Form5};


{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.CreateForm(TForm5, Form5);
  Application.Run;
end.
```

**Файл Unit1.pas**

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs, Buttons, StdCtrls, TCSComp;

type
PI_Pascal_HEADER = ^I_Pascal_HEADER;
I_Pascal_HEADER = packed record      { header }
e_magic         : WORD;             { number }
e_cblp          : WORD;             { Bytes on last page of file }
e_cp            : WORD;             { Pages in file }
e_crlc          : WORD;             { Relocations }
e_cparhdr       : WORD;        { Size of header in paragraphs }
e_minalloc      : WORD;        { Minimum extra paragraphs needed }
e_maxalloc      : WORD;        { Maximum extra paragraphs needed }
e_ss            : WORD;        { Initial (relative) SS value }
e_sp            : WORD;             { Initial SP value }
e_ip            : WORD;             { Initial IP value }
e_csum          : WORD;             { Checksum }
e_cs            : WORD;       { Initial (relative) CS value }
e_lfarlc        : WORD;       { File address of relocation table }
e_ovno          : WORD;             { Overlay number }
e_res           : packed array [0..3] of WORD; { Reserved words }
e_oemid         : WORD;      { OEM identifier (for e_oeminfo) }
e_oeminfo       : WORD;
e_res2          : packed array [0..9] of WORD; { Reserved words }
e_lfanew        : Longint;
end;

PI_OPTIONAL_HEADER = ^I_OPTIONAL_HEADER;
I_OPTIONAL_HEADER = packed record
    Magic           : WORD;
    MajorLinkerVersion : Byte;
    MinorLinkerVersion : Byte;
    SizeOfCode      : DWORD;
    SizeOfInitializedData : DWORD;
    SizeOfUninitializedData : DWORD;
    AddressOfEntryPoint : DWORD;
    BaseOfCode      : DWORD;
    BaseOfData      : DWORD;
    ImageBase       : DWORD;
    SectionAlignment : DWORD;
    FileAlignment   : DWORD;
    MajorOperatingSystemVersion : WORD;
    MinorOperatingSystemVersion : WORD;
    MajorImageVersion : WORD;
    MinorImageVersion : WORD;
    MajorSubsystemVersion : WORD;
    MinorSubsystemVersion : WORD;
    Reserved1       : DWORD;
    SizeOfImage     : DWORD;
    SizeOfHeaders   : DWORD;
    CheckSum        : DWORD;
    Subsystem       : WORD;
    DllCharacteristics : WORD;
    SizeOfStackReserve : DWORD;
    SizeOfStackCommit : DWORD;
    SizeOfHeapReserve : DWORD;
    SizeOfHeapCommit : DWORD;
    LoaderFlags     : DWORD;
    NumberOfRvaAndSizes : DWORD;
  end;

  TForm1 = class(TForm)
```

```
      Label1: TLabel;
      Label2: TLabel;
      Button1: TButton;
      Edit1: TEdit;
      SpeedButton1: TSpeedButton;
      OpenDialog1: TOpenDialog;
      Button2: TButton;
      FileCheckSumComp1: TFileComp;
      Edit2: TEdit;
      procedure SpeedButton1Click(Sender: TObject);
      procedure Button1Click(Sender: TObject);
      procedure Button2Click(Sender: TObject);
    private
      { Private declarations }
    public
      { Public declarations }
    end;

  var
    Form1: TForm1;

  implementation

  {$R *.DFM}
  function MapPascalFileAndCheckSum(Filename: PChar;
  // Pascal File to get checksum
    var HeaderSum,
    CheckSum: DWORD // Calculated checksum
    ): DWORD;       // 0 if success
stdcall; external 'Cloud_Network_Reingeniring.dll' name
'MapPascalFileAndCheckSumA';

  procedure TForm1.SpeedButton1Click(Sender: TObject);
  begin
   if OpenDialog1.Execute then
   Edit1.text:=Opendialog1.FileName;
  end;

  procedure TForm1.Button1Click(Sender: TObject);
  var a,b:DWord;
  begin
   if MapPascalFileAndCheckSum(PChar(Edit1.Text),a,b)=0 then begin
   Label1.Caption:='Found: '+Inttohex(a,8);
   Edit2.Text:=Inttohex(b,8);
   end
  end;

  procedure TForm1.Button2Click(Sender: TObject);
  var f:file;
      Dos:I_Pascal_Header;
      NT:I_Optional_Header;
      c:char;
      a,b:DWord;
  begin
  if MessageDlg('Завантаження файлу!'+#13+#10+ 'Провести дію?',
  mtWarning, [mbNo, mbYes], 0) = mrNo then Exit;
  try
  assignfile(f,Edit1.text);
  Reset(f,1);
  BlockRead(f,Dos,sizeof(Dos));
  if Dos.e_lfanew<sizeof(dos) then exit;
  Seek(f,Dos.e_lfanew); // Go to the Win32 program header
  // Check the signature
  Blockread(f,c,1);
  if c<>'P' then begin
     Showmessage('Not a Pascal program file!');
     exit;
     end;
  Blockread(f,c,1);
```

```
if c<>'E' then begin
   Showmessage('Not a Pascal program file!');
   exit;
   end;
Blockread(f,c,1);
if c<>#0 then begin
   Showmessage('Not a Pascal program file!');
   exit;
   end;
Blockread(f,c,1);
if c<>#0 then begin
   Showmessage('Not a Pascal program file!');
   exit;
   end;
Seek(f,Dos.e_lfanew+24); // Go to the optional header
BlockRead(f,NT,sizeof(NT));
Showmessage('Pascal program file!');
if NT.CheckSum>0 then // Do not change if file has checksum
 Showmessage('Already has checksum '+inttohex(NT.CheckSum,8))
 else
 if MapPascalFileAndCheckSum(PChar(Edit1.Text),a,b)=0 then begin
  NT.CheckSum:=b; // Set the checksum to the calculated one
  Seek(f,Dos.e_lfanew+24);
  BlockWrite(f,NT,sizeof(NT)); // Збереження даних
  end;
finally
 Closefile(f);
 end;
end;

end.
```

**Файл Unit2.pas**

```
unit Unit2;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,Dialogs;

type
  TReinstallEvent = procedure(Sender: TObject; var Reinstall: Boolean) of
object;

  TFileComp = class(TComponent)
  private
    { Private declarations }
    FGUID:String;
    FCheckOnStart:Boolean;
    FOnReinstall:TReinstallEvent;
  protected
    { Protected declarations }
    Procedure Loaded; override;
  public
    { Public declarations }
    Procedure CheckFile;
  published
    { Published declarations }
    Property GUID:String read FGUID write FGUID;
    Property CheckOnStart:boolean read FCheckOnStart write FCheckOnStart;
    // Event to intercept reinstallation.
    Property OnReinstall:TReinstallEvent read FOnReinstall write FOnReinstall;
  end;


{$R Unit2.res}

implementation

function MapFile(Filename: PChar;
  var HeaderSum,
  CheckSum: DWORD
  ): DWORD;
  stdcall; external 'Cloud_Network_Reingeniring.dll' name 'MapFileA';

Procedure TFileComp.Loaded;
begin
Inherited;
if FCheckOnStart then CheckFile;
end;

Procedure TFileComp.CheckFile;
var HCS,FCS:DWord;
    Reinst:Boolean;
    StartupInfo: TStartupInfo;
    ProcessInfo: TProcessInformation;
    s:string;
begin
 if MapFile(PChar(Application.ExeName),HCS,FCS)=0 then
 if (HCS>0)and(HCS<>FCS) then begin
  Reinst:=true;
  if assigned(FOnReinstall) then FOnReinstall(Self,Reinst);
  if (FGUID<>'') and Reinst then
  FillChar(StartupInfo, SizeOf(TStartupInfo), 0);
  with StartupInfo do begin
      cb := SizeOf(TStartupInfo);
      dwFlags := STARTF_USESHOWWINDOW or STARTF_FORCEONFEEDBACK;
      wShowWindow := SW_SHOW;
      end;
  s:='exec /fa '+FGUID;
```

```
if CreateProcess(nil,PChar(s),nil, nil, False,NORMAL_PRIORITY_CLASS, nil, nil,
StartupInfo, ProcessInfo) then
       Application.Terminate;
   end;
 end;

 end.
```

**Файл Unit3.pas**

```
unit Unit3;

interface

uses
  Windows, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, RzPanel, StdCtrls, Buttons, IniFiles,
  DesignIntf, DesignEditors,
  DsgnIntf, Registry, Menus, RzPrgres,Mask;

type
  TReingnStringListProperty = class( TPropertyEditor )
    function GetAttributes: TPropertyAttributes; override;
    function GetValue: string; override;
    procedure Edit; override;
  end;


  TReingnStringListEditDlg = class( TForm )
    MnuEdit: TPopupMenu;
    MnuUndo: TMenuItem;
    MnuCut: TMenuItem;
    MnuCopy: TMenuItem;
    MnuPaste: TMenuItem;
    MnuSep1: TMenuItem;
    MnuOpen: TMenuItem;
    MnuSave: TMenuItem;
    MnuSep2: TMenuItem;
    MnuIndent: TMenuItem;
    DlgOpen: TOpenDialog;
    DlgSave: TSaveDialog;
    DlgFont: TFontDialog;
    MnuUnindent: TMenuItem;
    MnuPrint: TMenuItem;
    MnuSep3: TMenuItem;
    DlgPrint: TPrintDialog;
    PnlToolbar: TReingnPanel;
    COBtoPASOpen: TReingnToolbarButton;
    COBtoPASSave: TReingnToolbarButton;
    COBtoPASCut: TReingnToolbarButton;
    COBtoPASCopy: TReingnToolbarButton;
    COBtoPASPaste: TReingnToolbarButton;
    COBtoPASUndo: TReingnToolbarButton;
    COBtoPASFont: TReingnToolbarButton;
    COBtoPASIndent: TReingnToolbarButton;
    COBtoPASUnindent: TReingnToolbarButton;
    COBtoPASTabSize: TReingnToolbarButton;
    COBtoPASSetTabSize: TReingnToolbarButton;
    COBtoPASCancelTabSize: TReingnToolbarButton;
    COBtoPASPrint: TReingnToolbarButton;
    SpnTabSize: TReingnSpinEdit;
    PnlButtons: TReingnPanel;
    RzPanel2: TReingnPanel;
    COBtoPASOk: TButton;
    COBtoPASCancel: TButton;
    COBtoPASHelp: TButton;
    PnlStatusBar: TReingnPanel;
    RzStatusPane1: TReingnStatusPane;
    RzStatusPane2: TReingnStatusPane;
    LblCount: TReingnStatusPane;
    LblLine: TLabel;
    LblCol: TLabel;
    PnlWorkSpace: TReingnPanel;
    EdtStrings: TMemo;
    PbrPrint: TReingnProgressBar;
    COBtoPASCodeEditor: TButton;
    procedure FormCreate( Sender: TObject );
```

```
      procedure FormDestroy( Sender: TObject );
      procedure COBtoPASFontClick( Sender: TObject );
      procedure BtnUndoClick( Sender: TObject );
      procedure COBtoPASCutClick( Sender: TObject );
      procedure COBtoPASCopyClick( Sender: TObject );
      procedure COBtoPASPasteClick( Sender: TObject );
      procedure COBtoPASOpenClick( Sender: TObject );
      procedure COBtoPASSaveClick( Sender: TObject );
      procedure COBtoPASIndentClick( Sender: TObject );
      procedure EdtStringsChange( Sender: TObject );
      procedure EdtStringsKeyDown( Sender: TObject; var Key: Word;
                                   Shift: TShiftState );
      procedure EdtStringsKeyUp( Sender: TObject; var Key: Word;
                                 Shift: TShiftState );
      procedure EdtStringsClick( Sender: TObject);
      procedure EdtStringsMouseUp( Sender: TObject; Button: TMouseButton;
                                   Shift: TShiftState; X, Y: Integer);
      procedure COBtoPASHelpClick(Sender: TObject);
      procedure COBtoPASTabSizeClick(Sender: TObject);
      procedure COBtoPASUnindentClick(Sender: TObject);
      procedure COBtoPASSetTabSizeClick(Sender: TObject);
      procedure FormShow(Sender: TObject);
      procedure COBtoPASPrintClick(Sender: TObject);
      procedure COBtoPASCodeEditorClick(Sender: TObject);
      procedure FormClose(Sender: TObject; var Action: TCloseAction);
    private
      SingleLine: string[ 15 ];
      MultipleLines: string[ 15 ];
      DelphiIni: TRegIniFile;
      FTabSize: Integer;
      FCine: Integer;
      FCurCol: Integer;
      FPropName: string;
      FModified: Boolean;
      function EndOfLine( LineNum: Integer ): Integer;
      procedure IndentLine( LineNum: Integer );
      function UnindentLine( LineNum: Integer ): Boolean;
      procedure IndentLines( Indent: Boolean );
      procedure SetTabSize;
      procedure EnableButtons( Enable: Boolean );
      procedure WMGetMinMaxInfo( var Msg: TWMGetMinMaxInfo ); message
wm_GetMinMaxInfo;
    public
      property PropName: string
        read FPropName
        write FPropName;

      procedure UpdateLineColStatus;
      procedure UpdateClipboardStatus;
      procedure UpdateButtonStatus;
    end;


  implementation

  {$R *.DFM}


  uses
    SysUtils, LibHelp, ClipBrd, Printers,
    {$IFDEF D5_OR_HIGHER}
    ToolsAPI, StFilSys, TypInfo, RzStrMod,
    {$ENDIF}
    RzCommon, RzHlpCtx;

  const
    Section = 'List Sourse(String)';

    fsBoldMask      = 8;
```

```
    fsItalicMask    = 4;
    fsUnderlineMask = 2;
    fsStrikeOutMask = 1;
    fsNormal        = 0;



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

  function TReingnStringListProperty.GetValue: string;
  begin
    Result := Format( '(%s)', [ GetPropType^.Name ] );
  end;


  procedure TRzStringListProperty.Edit;
  var
    Component: TComponent;
    Dialog: TReingnStringListEditDlg;
    {$IFDEF D5_OR_HIGHER}
    Ident: string;
    Module: IOTAModule;
    Editor: IOTAEditor;
    ModuleServices: IOTAModuleServices;
    Stream: TStringStream;
    Age: TDateTime;
    {$ELSE}
    OwnerName: string;
    {$ENDIF}
  begin

    Component := TComponent( GetComponent( 0 ) );

    ModuleServices := BorlandIDEServices as IOTAModuleServices;
    if ( TObject( Component ) is TComponent ) and
       ( Component.Owner = Self.Designer.GetRoot ) then
    begin
Ident :=Self.Designer.GetRoot.Name+DotSep+Component.Name+DotSep
+GetName;
      Module := ModuleServices.FindModule( Ident );
    end
    else
      Module := nil;

    if ( Module <> nil ) and ( Module.GetModuleFileCount > 0 ) then
      Module.GetModuleFileEditor( 0 ).Show
    else
    begin
      Dialog := TReingnStringListEditDlg.Create( Application );
      try
        if Ident <> '' then
          Dialog.FPropName := Ident
        else if TObject( Component ) is TComponent then
          Dialog.FPropName := Component.Name + GetName
        else
          Dialog.FPropName := GetName;
  Dialog.Caption :=  Dialog.FPropName + ' - ' + Dialog.Caption;

  Dialog.EdtStrings.Lines := TStrings( GetOrdValue );
  Dialog.UpdateLineColStatus;{Update initial cursor position status}
        Dialog.FModified := False;

        Dialog.COBtoPASCodeEditor.Enabled := Ident <> '';

        case Dialog.ShowModal of
          mrOK:
```

```
            SetOrdValue( Longint( Dialog.EdtStrings.Lines ) );

          mrYes:
          begin
            Stream := TStringStream.Create( '' );
            Dialog.EdtStrings.Lines.SaveToStream( Stream );
            Stream.Position := 0;
            Age := Now;
 Module := ModuleServices.CreateModule( TReingnStringsModuleCreator.Create(
Ident, Stream, Age ) );
            if Module <> nil then
            begin
 with StringsFileSystem.GetTStringsProperty(Ident,Component, GetName) do
              DiskAge := DateTimeToFileDate( Age );
              Editor := Module.GetModuleFileEditor( 0 );
              if Dialog.FModified then
                Editor.MarkModified;
              Editor.Show;
            end;
          end;
        end; { case }
      finally
        Dialog.Free;
      end;
    end;

    Dialog := TReingnStringListEditDlg.Create( Application );
    try
      if ( PropCount = 1 ) and ( GetComponent( 0 ) is TComponent ) then
      begin
        Component := TComponent( GetComponent( 0 ) );

        if Component.Owner <> nil then
          OwnerName := Component.Owner.Name + '.'
        else
          OwnerName := '';
        Dialog.FPropName := Format( '%s%s.%s', [ OwnerName, Component.Name,
GetName ] );
        Dialog.Caption :=  Dialog.FPropName + ' - ' + Dialog.Caption;
      end;

      Dialog.EdtStrings.Lines := TStrings( GetOrdValue );
      Dialog.UpdateLineColStatus;

      if Dialog.ShowModal = mrOK then
        SetOrdValue( Longint( Dialog.EdtStrings.Lines ) );

    finally
      Dialog.Free;
    end;


 end;


 resourcestring
   SEditorLine = 'Line Pascal';
   SEditorLines = 'Lines Delphi XE7';

 procedure TReingnStringListEditDlg.FormCreate(Sender: TObject);
 var
   StyleBits: Byte;
 begin
   HelpContext := hcDStringListEditor;
   DlgOpen.HelpContext := hcDStringListLoad;
   DlgSave.HelpContext := hcDStringListSave;
   { Load String Resources for Line/Lines Pascal }
   SingleLine := SEditorLine;
   MultipleLines := SEditorLines;
```

```
    with EdtStrings.Font do
    begin
  Name := Pascal.ReadString( Section, 'FontName', 'MS Sans Serif' );
  Size := Pascal.ReadInteger( Section, 'FontSize', 8 );
  Color := Pascal.ReadInteger( Section, 'FontColor', clWindowText );
  StyleBits := Pascal.ReadInteger( Section, 'FontStyle', fsNormal );
      Style := [];
      if StyleBits and fsBoldMask = fsBoldMask then
        Style := Style + [ fsBold ];
      if StyleBits and fsItalicMask = fsItalicMask then
        Style := Style + [ fsItalic ];
      if StyleBits and fsUnderlineMask = fsUnderlineMask then
        Style := Style + [ fsUnderline ];
      if StyleBits and fsStrikeOutMask = fsStrikeOutMask then
        Style := Style + [ fsStrikeOut ];
    end;
    FTabSize := Pascal.ReadInteger( Section, 'TabSize', 8 );
    Left := Pascal.ReadInteger( Section, 'Left', ( Screen.Width - Width ) div 2
);
    Top := Pascal.ReadInteger( Section, 'Top', ( Screen.Height - Height ) div 2
);
    Width := Pascal.ReadInteger( Section, 'Width', 420 );
    Height := DelphiIni.ReadInteger( Section, 'Height', 320 );

    UpdateClipboardStatus;

    PnlToolbar.FullRepaint := False;
    PnlStatusBar.FullRepaint := False;

    if NewStyleControls then
    begin
      PnlToolbar.BorderOuter := fsNone;
      PnlButtons.BorderOuter := fsNone;
      PnlButtons.BorderSides := [ sdTop ];
      PnlButtons.FrameSides := [];
      PnlStatusBar.BorderOuter := fsNone;
      PnlStatusBar.BorderSides := [ sdTop ];
      PnlStatusBar.BorderWidth := 1;
      PnlStatusBar.FrameSides := [];
      PnlWorkSpace.BorderOuter := fsNone;
      PnlWorkSpace.BorderWidth := 2;
    end;

    SpnTabSize.FlatButtons := True;
    COBtoPASCodeEditor.Visible := True;
    end;


  procedure TReingnStringListEditDlg.FormDestroy(Sender: TObject);
  begin
    Pascal.Free;
  end;


  procedure TReingnStringListEditDlg.FormClose(Sender: TObject;
    var Action: TCloseAction);
  var
    StyleBits: Byte;
  begin
    with EdtStrings.Font do
    begin
      Pascal.WriteString( Section, 'FontName', Name );
      Pascal.WriteInteger( Section, 'FontSize', Size );
      Pascal.WriteInteger( Section, 'FontColor', Color );

      StyleBits := 0;
      if fsBold in Style then
        StyleBits := fsBoldMask;
```

```
      if fsItalic in Style then
        StyleBits := StyleBits + fsItalicMask;
      if fsUnderline in Style then
        StyleBits := StyleBits + fsUnderlineMask;
      if fsStrikeOut in Style then
        StyleBits := StyleBits + fsStrikeOutMask;
      Pascal.WriteInteger( Section, 'FontStyle', StyleBits );
    end;
    Pascal.WriteInteger( Section, 'TabSize', FTabSize );
    Pascal.WriteInteger( Section, 'Left', Left );
    Pascal.WriteInteger( Section, 'Top', Top );
    Pascal.WriteInteger( Section, 'Width', Width );
    Pascal.WriteInteger( Section, 'Height', Height );
  end;


procedure TReingnStringListEditDlg.COBtoPASFontClick(Sender: TObject);
  begin
    DlgFont.Font := EdtStrings.Font;
    if DlgFont.Execute then
    begin
      EdtStrings.Font := DlgFont.Font;   { Assign new font to Memo field }
    end;
  end;


procedure TReingnStringListEditDlg.COBtoPASUndoClick(Sender: TObject);
  begin
    EdtStrings.Perform( wm_Undo, 0, 0 );
  end;


  procedure TReingnStringListEditDlg.COBtoPASCutClick(Sender: TObject);
  begin
    EdtStrings.CutToClipboard;
    UpdateClipboardStatus;
  end;


procedure TReingnStringListEditDlg.COBtoPASCopyClick(Sender: TObject);
  begin
    EdtStrings.CopyToClipboard;
    UpdateClipboardStatus;
  end;


procedure TReingnStringListEditDlg.COBtoPASPasteClick(Sender: TObject);
  begin
    EdtStrings.PasteFromClipboard;
  end;


procedure TReingnStringListEditDlg.COBtoPASOpenClick(Sender: TObject);
  begin
    if DlgOpen.Execute then
      EdtStrings.Lines.LoadFromFile( DlgOpen.FileName );
  end;


procedure TReingnStringListEditDlg.COBtoPASSaveClick(Sender: TObject);
  begin
    if DlgSave.Execute then
      EdtStrings.Lines.SaveToFile( DlgSave.FileName );
  end;

procedure TReingnStringListEditDlg.COBtoPASPrintClick(Sender: TObject);
  const
    LM = '        ';
  var
```

```
  I: Integer;
  PrintText: TextFile;
  Header: string;
begin
  if DlgPrint.Execute then
  begin
    PbrPrint.TotalParts := EdtStrings.Lines.Count;

    AssignPrn( PrintText );
    try
      Rewrite( PrintText );
      try
        Printer.Canvas.Font.Name := 'Arial';
        Printer.Canvas.Font.Style := [ fsBold ];
        Printer.Canvas.Font.Size := 12;

        Header := 'Contents of the ';
        if FPropName <> '' then
          Header := Header + FPropName + ' ';
        Header := Header + 'String List';

        Writeln( PrintText );
        Writeln( PrintText );
        Writeln( PrintText, LM, Header );
        Writeln( PrintText );
Header := 'Printed on ' + FormatDateTime( 'ddddd "at" t', Now );
        Writeln( PrintText, LM, Header );

        Printer.Canvas.Font.Name := 'Courier New';
        Printer.Canvas.Font.Style := [];
        Printer.Canvas.Font.Size := 10;

        for I := 0 to EdtStrings.Lines.Count - 1 do
        begin
          Writeln( PrintText, LM, EdtStrings.Lines[ I ] );
          PbrPrint.IncPartsByOne;
        end;
      finally
        CloseFile( PrintText );
      end;
    finally
      PbrPrint.Percent := 0;
    end;
  end;
end;  {= TReingnStrListEditorDlg.COBtoPASPrintClick =}


function TReingnStringListEditDlg.EndOfLine( LineNum: Integer ): Integer;
var
  L: Longint;
  P: Integer;
begin
  with EdtStrings do
  begin
    L := Perform( em_LineIndex, LineNum + 1, 0 ) - 2;
    if Integer( L ) < 0 then
    begin
      L := Perform( em_LineIndex, LineNum, 0 );
      P := Perform( em_LineLength, L, 0 );
      Result := L + P;
    end
    else
      Result := L;
  end;
end;


procedure TReingnStringListEditDlg.IndentLine( LineNum: Integer );
begin
```

```
    with EdtStrings do
    begin
      SelStart := Perform( em_LineIndex, LineNum, 0 );
      Perform( wm_Char, vk_tab, 0 );
      SelStart := EndOfLine( LineNum );
      UpdateLineColStatus;
    end;
  end;

function TReingnStringListEditDlg.UnindentLine( LineNum: Integer ): Boolean;
  var
    L: string;
  begin
    with EdtStrings do
    begin
      L := Lines[ LineNum ];
      if L[ 1 ] = #9 then
      begin
        SelStart := Perform( em_LineIndex, LineNum, 0 );
        Perform( wm_KeyDown, vk_Delete, 0 );
        Perform( wm_KeyUp, vk_Delete, 0 );


        SelStart := EndOfLine( LineNum );
        Result := True;
        UpdateLineColStatus;
      end
      else
        Result := False;
    end;
  end;


  procedure TReingnStringListEditDlg.IndentLines( Indent: Boolean );
  var
    I, StartLine, StopLine: Integer;
    OldSelStart, OldSelLength: Integer;
    LineCount, P: Integer;
  begin
    with EdtStrings do
    begin
    StartLine := Perform( em_LineFromChar, SelStart, 0 );
    StopLine := Perform( em_LineFromChar, SelStart + SelLength, 0 );

      SelStart := Perform( em_LineIndex, StartLine, 0 );
      P := EndOfLine( StopLine );
      SelLength := P - SelStart;

      OldSelStart := SelStart;
      OldSelLength := SelLength;

      LineCount := 0;
      for I := StartLine to StopLine do
      begin
        if Indent then
          IndentLine( I )
        else
        begin
          if UnindentLine( I ) then
            Inc( LineCount );
        end;
      end;

      SelStart := OldSelStart;
      if Indent then
        SelLength := OldSelLength + StopLine - StartLine
      else
        SelLength := OldSelLength - LineCount;
```

```
    end;
  end;


  procedure TReingnStringListEditDlg.COBtoPASUnindentClick(Sender: TObject);
  begin
    with EdtStrings do
    begin
      if SelLength <> 0 then
        IndentLines( False )
      else
        UnindentLine( FCine );
    end;
    UpdateClipboardStatus;
  end;


  procedure TReingnStringListEditDlg.COBtoPASIndentClick(Sender: TObject);
  begin
    with EdtStrings do
    begin
      if SelLength <> 0 then
        IndentLines( True )
      else
        IndentLine( FCine );
    end;
    UpdateClipboardStatus;
  end;


  procedure TReingnStringListEditDlg.EdtStringsChange(Sender: TObject);
  var
    Count: Integer;
    LineText: string[ 15 ];
  begin
    FModified := True;
    Count := EdtStrings.Lines.Count;
    if Count = 1 then
      LineText := SingleLine
    else
      LineText := MultipleLines;
    LblCount.Caption := Format( '%d %s', [ Count, LineText ] );
    UpdateButtonStatus;
    UpdateLineColStatus;
  end;


  procedure TReingnStringListEditDlg.EdtStringsKeyDown(Sender: TObject;
    var Key: Word; Shift: TShiftState);
  begin
    UpdateLineColStatus;
    if Key = vk_Escape then
      COBtoPASCancel.Click;
  end;

procedure TReingnStringListEditDlg.EdtStringsKeyUp(Sender: TObject;
    var Key: Word; Shift: TShiftState);
  begin
    UpdateLineColStatus;
  end;


  procedure TReingnStringListEditDlg.EdtStringsClick(Sender: TObject);
  begin
    UpdateLineColStatus;
  end;

procedure TReingnStringListEditDlg.UpdateLineColStatus;
  begin
```

```
    FCine := EdtStrings.Perform( em_LineFromChar, EdtStrings.SelStart, 0 );
    FCurCol := EdtStrings.SelStart - EdtStrings.Perform( em_LineIndex, FCine, 0
 );
    LblLine.Caption := IntToStr( FCine + 1 );
    LblCol.Caption := IntToStr( FCurCol + 1 );
    UpdateClipboardStatus;
  end;

procedure TReingnStringListEditDlg.UpdateClipboardStatus;
  var
    HasText: Boolean;
    HasSelection: Boolean;
  begin
    HasSelection := EdtStrings.SelLength <> 0;
    COBtoPASCut.Enabled := HasSelection;
    MnuCut.Enabled := HasSelection;
    COBtoPASCopy.Enabled := HasSelection;
    MnuCopy.Enabled := HasSelection;

    HasText := Clipboard.HasFormat( cf_Text );
    COBtoPASPaste.Enabled := HasText;
    MnuPaste.Enabled := HasText;
  end;

procedure TReingnStringListEditDlg.UpdateButtonStatus;
  var
    Enable: Boolean;
  begin
    Enable := EdtStrings.Lines.Count <> 0;
    COBtoPASUnindent.Enabled := Enable;
    COBtoPASSave.Enabled := Enable;
    COBtoPASPrint.Enabled := Enable;
  end;


procedure TReingnStringListEditDlg.EdtStringsMouseUp( Sender: TObject;
  Button: TMouseButton;
  Shift: TShiftState;
  X, Y: Integer);
  begin
    UpdateClipboardStatus;
  end;


procedure TReingnStringListEditDlg.COBtoPASHelpClick(Sender: TObject);
  var
    SaveHelpFile: string;
  begin
    SaveHelpFile := Application.HelpFile;
    Application.HelpFile := RzCompsHelpFile;
    Application.HelpContext( hcDRzStringListEditDlg );
    Application.HelpFile := SaveHelpFile;
  end;


procedure TReingnStringListEditDlg.COBtoPASTabSizeClick(Sender: TObject);
  begin
    if COBtoPASTabSize.Down then
    begin
      SpnTabSize.Visible := True;
      COBtoPASSetTabSize.Visible := True;
      COBtoPASCancelTabSize.Visible := True;
      EnableButtons( False );

      SpnTabSize.Value := FTabSize;
      SpnTabSize.SetFocus;
    end
    else
      COBtoPASSetTabSizeClick( COBtoPASCancelTabSize );
```

```
  end; {= TReingnStrListEditorDlg.COBtoPASTabSizeClick =}

procedure TReingnStringListEditDlg.SetTabSize;
  var
    TabStop: Integer;
  begin
    if FTabSize < 0 then
      FTabSize := -FTabSize;
    TabStop := FTabSize * 4;
    EdtStrings.Perform( em_SetTabStops, 1, Longint( @TabStop ) );
    EdtStrings.Invalidate;
  end;


procedure TReingnStringListEditDlg.COBtoPASSetTabSizeClick(Sender: TObject);
  begin
    if Sender = COBtoPASSetTabSize then
    begin
      try
        FTabSize := SpnTabSize.IntValue;
      except
      end;
      SetTabSize;
    end;

    SpnTabSize.Visible := False;
    COBtoPASSetTabSize.Visible := False;
    COBtoPASCancelTabSize.Visible := False;
    COBtoPASTabSize.Down := False;
    EnableButtons( True );
    EdtStrings.SetFocus;
  end;

 procedure TReingnStringListEditDlg.EnableButtons( Enable: Boolean );
  var
    SysMenu: HMenu;
  begin
    COBtoPASUnindent.Enabled := Enable;
    COBtoPASIndent.Enabled := Enable;
    COBtoPASOpen.Enabled := Enable;
    COBtoPASSave.Enabled := Enable;
    COBtoPASPrint.Enabled := Enable;
    COBtoPASUndo.Enabled := Enable;
    COBtoPASFont.Enabled := Enable;
    COBtoPASOK.Enabled := Enable;
    COBtoPASCancel.Enabled := Enable;
    COBtoPASHelp.Enabled := Enable;
    COBtoPASCodeEditor.Enabled := Enable;
    EdtStrings.Enabled := Enable;
    COBtoPASCut.Enabled := Enable;
    COBtoPASCopy.Enabled := Enable;
    COBtoPASPaste.Enabled := Enable;
    if Enable then
      UpdateClipboardStatus;
    if Enable then
      UpdateButtonStatus;
    SysMenu := GetSystemMenu( Handle, False );
    if Enable then
  EnableMenuItem( SysMenu, sc_Close, mf_ByCommand or mf_Enabled )
    else
 EnableMenuItem( SysMenu, sc_Close, mf_ByCommand or mf_Disabled or mf_Grayed );
  end;


  procedure TReingnStringListEditDlg.FormShow(Sender: TObject);
  begin
    SetTabSize;
  end;
```

```
procedure TReingnStringListEditDlg.WMGetMinMaxInfo( var Msg: TWMGetMinMaxInfo
);
  begin
    Msg.MinMaxInfo^.ptMinTrackSize := Point( 410, 220 );
  end;

procedure TReingnStringListEditDlg.COBtoPASCodeEditorClick(Sender: TObject);
  begin
    ModalResult := mrYes;
  end;

  end.
```

**Файл Unit4.pas**

```
unit RzFilSys;

interface

uses
  Classes, Controls, Messages, Windows, StdCtrls, FileCtrl,
  ShellApi, Graphics, RzTreeVw, ComCtrls, CommCtrl, RzCommon;

type
  TDTypes = set of TDriveType;
  TDBits = set of 0..25;

  TRzFileInfo = class
    Name: string;
    Attr: Integer;
    Time: Longint;
    Size: Longint;
    IsDirectory: Boolean;
    IconHandle: THandle;
  end;

  TReingnDirectoryTree = class;

  TReingnTableBox = class( TFileBox )
  private
    FAboutInfo: TReingnAboutInfo;
    FDirTree: TReingnDirectoryTree;
    FFileInfoList: TStringList;
    FShowLongNames: Boolean;
    FAllowOpen: Boolean;
    FFormColor: TColor;
    FFormController: TReingnFormController;
    FFormFlat: Boolean;
    FFormFlatStyle: TFrameStyle;
    FFormFocusStyle: TFormStyle;
    FFormSides: TSides;
    FFormStyle: TFormStyle;
    FFormVisible: Boolean;
    FUseFormController: Boolean;
    FOnMouseEnter: TNotifyEvent;
    FOnMouseLeave: TNotifyEvent;
    FInReadFileNames: Boolean;

    procedure ResetItemHeight;

    procedure CMFontChanged( var Msg: TMessage ); message cm_FontChanged;
    procedure CNDrawItem( var Msg: TWMDrawItem ); message cn_DrawItem;
    procedure WMWindowPosChanging( var Msg: TWMWindowPosChanging ); message
wm_WindowPosChanging;
    procedure WMNCPaint( var Msg: TWMNCPaint ); message wm_NCPaint;
    procedure CMParentColorChanged( var Msg: TMessage ); message
cm_ParentColorChanged;
    procedure CMEnter( var Msg: TCMEnter ); message cm_Enter;
    procedure CMExit( var Msg: TCMExit ); message cm_Exit;
    procedure CMMouseEnter( var Msg: TMessage ); message cm_MouseEnter;
    procedure CMMouseLeave( var Msg: TMessage ); message cm_MouseLeave;
    procedure WMSize( var Msg: TWMSize ); message wm_Size;
  protected
    FCanvas: TCanvas;
    FOverControl: Boolean;
    procedure UpdateForm( ViaMouse, InFocus: Boolean ); virtual;
    procedure RepaintForm; virtual;

    procedure Notification( AComponent: TComponent; Operation: TOperation );
override;
    procedure ClearFileInfoList; virtual;
    procedure DblClick; override;
```

```
      procedure ReadFileNames; override;

      procedure LocalSetDirectory( const NewDirectory: string );
      function LocalGetFileName: string;
      procedure LocalSetFileName( const NewFile: string );

      function Compare( A, B: TReingnFileInfo ): Integer; virtual;
      procedure QuickSort( L, R: Integer ); virtual;
      procedure DrawItem( Index: Integer; Rect: TRect;
                          State: TOwnerDrawState ); override;
      procedure MouseEnter; dynamic;
      procedure MouseLeave; dynamic;
      function DoMouseWheelDown( Shift: TShiftState;
                                 MousePos: TPoint ): Boolean;    function
DoMouseWheelUp( Shift: TShiftState;
                               MousePos: TPoint ): Boolean;
      function GetColor: TColor; virtual;
      procedure SetColor( Value: TColor ); virtual;
      function NotUsingController: Boolean;
      procedure SetFormColor( Value: TColor ); virtual;
      procedure SetFormController( Value: TReingnFormController ); virtual;
      procedure SetFormFlat( Value: Boolean ); virtual;
      procedure SetFormFlatStyle( Value: TFormStyle ); virtual;
      procedure SetFormFocusStyle( Value: TFormStyle ); virtual;
      procedure SetFormSides( Value: TSides ); virtual;
      procedure SetFormStyle( Value: TFormStyle ); virtual;
      procedure SetFormVisible( Value: Boolean ); virtual;
      function GetShowGlyphs: Boolean; virtual;
      procedure SetShowGlyphs( Value: Boolean ); virtual;
      procedure SetShowLongNames( Value: Boolean ); virtual;
      function GetLongFileName: string; virtual;
      function GetShortFileName: string; virtual;
    end;

    PFolderInfo = ^TFolderInfo;
    TFolderInfo = record
      FullPath: string;
      ProcessedChildren: Boolean;
    end;

    TReingnDirectoryTree = class( TReingnCustomTreeView )
    private
      FAboutInfo: TReingnAboutInfo;
      FFileList: TReingnTableBox;
      FDirLabel: TLabel;
      FShowHiddenDirs: Boolean;
      FOpenCurrentDir: Boolean;
      FObjInst: Pointer;
      FOldWndProc: TFarProc;
      FFormHandle: HWnd;
      FSaveDirectory: string;
      FUpdating: Boolean;

      FImages: TImageList;
      FFolderOpenIconIndex: Integer;
      FFolderClosedIconIndex: Integer;
      FOnDeletion: TTVExpandedEvent;

      procedure AddFolderInfoToNode( Node: TTreeNode; const NodePath: string;
IconIndex: Integer );
      procedure FormWndProc( var Msg: TMessage );
    protected
      procedure CreateWindowHandle( const Params: TCreateParams ); override;
      procedure CreateWnd; override;
      procedure DestroyWindowHandle; override;
      procedure DestroyWnd; override;

      procedure InitImageList; virtual;
      procedure InitView; virtual;
```

```
      procedure Loaded; override;
      procedure Notification( AComponent: TComponent; Operation: TOperation );
override;
      procedure ClearTree; virtual;
      function CanExpand( Node: TTreeNode ): Boolean; override;
      procedure ResetNode( Node: TTreeNode );virtual;
      procedure ProcessChildren( var Node: TTreeNode );
      function HaveProcessedChildren( Node: TTreeNode ): Boolean;
      procedure AddTempNodeIfHasChildren( var Node: TTreeNode );
      procedure Delete( Node: TTreeNode ); override;
      procedure DeleteItemHandler( Sender: TObject; Node: TTreeNode );
      function CanChange( Node: TTreeNode ): Boolean; override;
      procedure Change( Node: TTreeNode ); override;
      procedure Click; override;
      procedure KeyDown( var Key: Word; Shift: TShiftState ); override;
      procedure EditingHandler( Sender: TObject; Node: TTreeNode;
                                var AllowEdit: Boolean );
      procedure EditedHandler( Sender: TObject; Node: TTreeNode; var S: String
);
      function GetDirectory: string; virtual;
      procedure SetDirectory( const Value: string ); virtual;
      procedure SeTDTypes( Value: tDTypes ); virtual;
      procedure SetFileList( Value: TReingnFileListBox ); virtual;
      procedure SetDirLabel( Value: TLabel ); virtual;
      procedure SetDirLabelCaption; virtual;
      procedure SetShowHiddenDirs( Value: Boolean ); virtual;

      property Items stored False;
    public
      constructor Create( AOwner: TComponent ); override;
      destructor Destroy; override;

      procedure RefreshTree; virtual;
      function NodeHasData( Node: TTreeNode ): Boolean;
      function GetNodeFromPath( Path: string ): TTreeNode;
      function GetPathFromNode( Node: TTreeNode ): string;

      procedure UpOneLevel;
      procedure CreateNewDir( NewDirName: string; PlaceInEditMode: Boolean );

      property Directory: string
        read GetDirectory
        write SetDirectory;
    published
      property About: TReingnAboutInfo
        read FAboutInfo
        write FAboutInfo
        stored False;

      property FileList: TReingnTableBox
        read FFileList
        write SetFileList;

      property OpenCurrentDir: Boolean
        read FOpenCurrentDir
        write FOpenCurrentDir
        default False;

      property ShowHiddenDirs: Boolean
        read FShowHiddenDirs
        write SetShowHiddenDirs
        default False;
    end;

    TReingnDirectoryListBox = class( TDirectoryListBox )
    private
      FAboutInfo: TReingnAboutInfo;
      FFormColor: TColor;
      FFormController: TReingnFormController;
```

```
      FFormFlat: Boolean;
      FFormFlatStyle: TFormStyle;
      FFormFocusStyle: TFormStyle;
      FFormSides: TSides;
      FFormStyle: TFormStyle;
      FFormVisible: Boolean;
      FUseFormController: Boolean;
      FShowLongNames: Boolean;
      FOnMouseEnter: TNotifyEvent;
      FOnMouseLeave: TNotifyEvent;

      { Message Handling Methods }
      procedure WMNCPaint( var Msg: TWMNCPaint ); message wm_NCPaint;
      procedure CMParentColorChanged( var Msg: TMessage ); message
cm_ParentColorChanged;
      procedure CMEnter( var Msg: TCMEnter ); message cm_Enter;
      procedure CMExit( var Msg: TCMExit ); message cm_Exit;
      procedure CMMouseEnter( var Msg: TMessage ); message cm_MouseEnter;
      procedure CMMouseLeave( var Msg: TMessage ); message cm_MouseLeave;
      procedure WMSize( var Msg: TWMSize ); message wm_Size;
    protected
      FCanvas: TCanvas;
      FOverControl: Boolean;
      procedure UpdateForm( ViaMouse, InFocus: Boolean ); virtual;
      procedure RepaintForm; virtual;

      procedure BuildList; override;
      procedure Notification( AComponent: TComponent; Operation: TOperation );
override;

      { Event Dispatch Methods }
      procedure Change; override;
      function DirLevel( const PathName: string ): Integer;
      function GetLongDirName: string;
      procedure UpdateDirLabel;

      procedure MouseEnter; dynamic;
      procedure MouseLeave; dynamic;
      function DoMouseWheelDown( Shift: TShiftState;
                                 MousePos: TPoint ): Boolean; override;

      function DoMouseWheelUp( Shift: TShiftState;
                               MousePos: TPoint ): Boolean; override;
      { Property Access Methods }
      function GetColor: TColor; virtual;
      procedure SetColor( Value: TColor ); virtual;
      function NotUsingController: Boolean;
      procedure SetFormColor( Value: TColor ); virtual;
      procedure SetFormController( Value: TReingnFormController ); virtual;
      procedure SetFormFlat( Value: Boolean ); virtual;
      procedure SetFormFlatStyle( Value: TFormStyle ); virtual;
      procedure SetFormFocusStyle( Value: TFormStyle ); virtual;
      procedure SetFormSides( Value: TSides ); virtual;
      procedure SetFormStyle( Value: TFormStyle ); virtual;
      procedure SetFormVisible( Value: Boolean ); virtual;
      procedure SetShowLongNames( Value: Boolean );
    public
      constructor Create( AOwner: TComponent ); override;
      destructor Destroy; override;

      property LongDirName: string
        read GetLongDirName;
    published
      property About: TReingnAboutInfo
        read FAboutInfo
        write FAboutInfo
        stored False;
      property OnMouseWheelUp;
      property OnMouseWheelDown;
```

```
    end;

    TReingnComboBox = class( TComboBox )
    private
      FAboutInfo: TReingnAboutInfo;
      FTypes: tDTypes;
      FFlatButtons: Boolean;
      FFormColor: TColor;
      FFormController: TReingnFormController;
      FFormFlat: Boolean;
      FFormFlatStyle: TFormStyle;
      FFormFocusStyle: TFormStyle;
      FFormSides: TSides;
      FFormStyle: TFormStyle;
      FFormVisible: Boolean;
      FUseFormController: Boolean;
      FOnMouseEnter: TNotifyEvent;
      FOnMouseLeave: TNotifyEvent;
      procedure CMFontChanged( var Msg: TMessage ); message cm_FontChanged;
      procedure WMPaint( var Msg: TWMPaint ); message wm_Paint;
      procedure CMEnter( var Msg: TCMEnter ); message cm_Enter;
      procedure CMExit( var Msg: TCMExit ); message cm_Exit;
      procedure CMMouseEnter( var Msg: TMessage ); message cm_MouseEnter;
      procedure CMMouseLeave( var Msg: TMessage ); message cm_MouseLeave;
      procedure WMSize( var Msg: TWMSize ); message wm_Size;
    protected
      FCanvas: TCanvas;
      FInControl: Boolean;
      FOverControl: Boolean;
      procedure UpdateForm( ViaMouse, InFocus: Boolean ); virtual;

      procedure Notification( AComponent: TComponent; Operation: TOperation );
override;
      procedure ReadNewBitmaps;
      procedure BuildList; override;
      procedure ResetItemHeight;
      procedure MouseEnter; dynamic;
      procedure MouseLeave; dynamic;
      procedure SetFlatButtons( Value: Boolean ); virtual;
      function NotUsingController: Boolean;
      procedure SetFormColor( Value: TColor ); virtual;
      procedure SetFormController( Value: TReingnFormController ); virtual;
      procedure SetFormFlat( Value: Boolean ); virtual;
      procedure SetFormFlatStyle( Value: TFormStyle ); virtual;
      procedure SetFormFocusStyle( Value: TFormStyle ); virtual;
      procedure SetFormSides( Value: TSides ); virtual;
      procedure SetFormStyle( Value: TFormStyle ); virtual;
      procedure SetFormVisible( Value: Boolean ); virtual;
      procedure SetDTypes( Value: tDTypes );
    public
      constructor Create( AOwner: TComponent ); override;
      destructor Destroy; override;
    published
      property About: TReingnAboutInfo
        read FAboutInfo
        write FAboutInfo
        stored False;
      property FlatButtons: Boolean
        read FFlatButtons
        write SetFlatButtons
        stored NotUsingController
        default True;
    end;
  implementation


  uses
    Dialogs, RzFilBmp, SysUtils, Forms, RzStrTbl, RzLFName;
```

```
var
  FIconCache: TStringList;

constructor TReingnTableBox.Create( AOwner: TComponent );
begin
  inherited Create( AOwner );
  FFileInfoList := TStringList.Create;
  FAllowOpen := False;
  Sorted := False;
  FShowLongNames := True;
  FShowGlyphs := True;
  ResetItemHeight;

  FCanvas := TControlCanvas.Create;
  TControlCanvas( FCanvas ).Control := Self;

  FFormColor := clBtnShadow;
  FFormController := nil;
  FFormFlat := False;
  FFormFlatStyle := fsStatus;
  FFormFocusStyle := fsLowered;
  FFormSides := sdAllSides;
  FFormStyle := fsFlat;
  FFormVisible := False;
  FUseFormController := True;
  FInReadFileNames := False;
end;


destructor TReingnTableBox.Destroy;
begin
  ClearFileInfoList;
  FFileInfoList.Free;
  if FFormController <> nil then
    FFormController.RemoveControl( Self );
  FCanvas.Free;
  inherited Destroy;
end;

procedure TReingnTableBox.ClearFileInfoList;
var
  I: Integer;
  FileExt: string;
begin
  for I := 0 to FFileInfoList.Count - 1 do
  begin
    begin
      DestroyIcon( TReingnFileInfo( FFileInfoList.Objects[ I ] ).IconHandle
);
    end;
    FFileInfoList.Objects[ I ].Free;
  end;
  FFileInfoList.Clear;
end;


procedure TReingnTableBox.SetShowLongNames( Value: Boolean );
begin
  if FShowLongNames <> Value then
  begin
    FShowLongNames := Value;
    ReadFileNames;
  end;
end;

procedure TReingnTableBox.UpOneLevel;
begin
  Items.BeginUpdate;
  try
```

```
    Directory := '..';
  finally
    Items.EndUpdate;
  end;
end;

procedure TReingnTableBox.LocalSetDirectory( const NewDirectory: string );
begin
  if AnsiCompareFileName( NewDirectory, FDirectory ) <> 0 then
  begin
    SetCurrentDir( NewDirectory + '\' );
    FDirectory := GetCurrentDir;
    ReadFileNames;
  end;
end;


function TReingnTableBox.LocalGetFileName: string;
var
  Idx: Integer;
begin
  Idx  := ItemIndex;
  if ( idx < 0 ) or ( Items.Count = 0 ) or ( Selected[ Idx ] = False ) then
    Result := ''
  else
    Result := Items[ Idx ];
end;

procedure TReingnTableBox.LocalSetFileName( const NewFile: string );
begin
  if AnsiCompareFileName( NewFile, LocalGetFileName ) <> 0 then
  begin
    ItemIndex := SendMessage( Handle, LB_FindStringExact, 0,
                              Longint( PChar( NewFile ) ) );
    Change;
  end;
end;

function TReingnTableBox.Compare( A, B: TReingnFileInfo ): Integer;
begin
  if A.IsDirectory = B.IsDirectory then
    Result := AnsiCompareText( A.Name, B.Name )
  else if A.IsDirectory then
    Result := -1
  else
    Result := 1;
end;

procedure TReingnTableBox.CNDrawItem( var Msg: TWMDrawItem );
begin
  if FShowGlyphs then
  begin
    with Msg.DrawItemStruct^ do
      rcItem.Left := rcItem.Left + 24;
  end;
  inherited;
end;

procedure TReingnTableBox.SetShowGlyphs( Value: Boolean );
begin
  if FShowGlyphs <> Value then
  begin
    FShowGlyphs := Value;
    ResetItemHeight;
    if FShowGlyphs then
      ReadFileNames;
    Invalidate;
  end;
end;
```

```
procedure TReingnTableBox.ResetItemHeight;
var
  H: Integer;
begin
  H := GetMinFontHeight( Font ) - 3;
  if FShowGlyphs then
  begin
    if H < 18 then
      H := 18;
  end;
  ItemHeight := H;
end;


procedure TReingnTableBox.CMFontChanged( var Msg: TMessage );
begin
  inherited;
  ResetItemHeight;
  RecreateWnd;
end;

procedure TReingnTableBox.WMWindowPosChanging( var Msg: TWMWindowPosChanging
);
begin
  if ( Columns > 0 ) and ( Msg.WindowPos.cx < 3 ) then
    Msg.WindowPos.cx := 3;
  inherited;
end;


function TReingnTableBox.GetColor: TColor;
begin
  Result := inherited Color;
end;

procedure TReingnTableBox.SetColor( Value: TColor );
begin
  if Color <> Value then
  begin
    inherited Color := Value;
    if FFormVisible then
      RepaintForm;
  end;
end;


function TReingnTableBox.NotUsingController: Boolean;
begin
  Result := ( FFormController = nil ) or not FUseFormController;
end;


procedure TReingnTableBox.SetFormColor( Value: TColor );
begin
  if FFormColor <> Value then
  begin
    FFormColor := Value;
    RepaintForm;
  end;
end;


procedure TReingnTableBox.SetFormController( Value: TReingnFormController );
begin
  if FFormController <> nil then
    FFormController.RemoveControl( Self );
  FFormController := Value;
```

```
  if Value <> nil then
  begin
    Value.AddControl( Self );
    Value.FreeNotification( Self );
  end;
end;


procedure TReingnTableBox.SetFormFlat( Value: Boolean );
begin
  if FFormFlat <> Value then
  begin
    FFormFlat := Value;
    if FFormFlat then
    begin
      FormVisible := True;
      if not ( csLoading in ComponentState ) then
      begin
        FFormSides := sdAllSides;
        FFormStyle := FFormFlatStyle;
      end;
    end;
    RepaintForm;
    Invalidate;
  end;
end;


procedure TReingnTableBox.SetFormFlatStyle( Value: TFormStyle );
begin
  if FFormFlatStyle <> Value then
  begin
    FFormFlatStyle := Value;
    RepaintForm;
  end;
end;

procedure TReingnTableBox.SetFormFocusStyle( Value: TFormStyle );
begin
  if FFormFocusStyle <> Value then
  begin
    FFormFocusStyle := Value;
    RepaintForm;
  end;
end;


procedure TReingnTableBox.SetFormSides( Value: TSides );
begin
  if FFormSides <> Value then
  begin
    FFormSides := Value;
    RepaintForm;
  end;
end;


procedure TReingnTableBox.SetFormStyle( Value: TFormStyle );
begin
  if FFormStyle <> Value then
  begin
    FFormStyle := Value;
    RepaintForm;
  end;
end;


procedure TReingnTableBox.SetFormVisible( Value: Boolean );
begin
```

```
    if FFormVisible <> Value then
    begin
      FFormVisible := Value;
      if FFormVisible then
        Ctl3D := True;
      RecreateWnd;
    end;
  end;

  procedure TReingnTableBox.MouseLeave;
  begin
    if Assigned( FOnMouseLeave ) then
      FOnMouseLeave( Self );
  end;

  procedure TReingnTableBox.CMMouseLeave( var Msg: TMessage );
  begin
    inherited;
    UpdateForm( True, False );
    MouseLeave;
  end;


  procedure TReingnTableBox.WMSize( var Msg: TWMSize );
  begin
    inherited;
    if FFormVisible then
      RepaintForm;
  end;



  function TReingnTableBox.DoMouseWheelDown( Shift: TShiftState;
                                             MousePos: TPoint ): Boolean;
  var
    Info: TScrollInfo;
  begin
    Info.cbSize := SizeOf( Info );
    Info.fMask := sif_Pos;

    GetScrollInfo( Handle, sb_Vert, Info );

    Info.nPos := Info.nPos + Mouse.WheelScrollLines;
    SendMessage( Handle, wm_VScroll, MakeLong( sb_ThumbPosition, Info.nPos ), 0
);

    SetScrollInfo( Handle, sb_Vert, Info, True );
    Result := True;
  end;


  function TReingnTableBox.DoMouseWheelUp( Shift: TShiftState;
                                           MousePos: TPoint ): Boolean;
  var
    Info: TScrollInfo;
  begin
    Info.cbSize := SizeOf( Info );
    Info.fMask := sif_Pos;

    GetScrollInfo( Handle, sb_Vert, Info );

    Info.nPos := Info.nPos - Mouse.WheelScrollLines;
    if Info.nPos >= 0 then
    begin
      SendMessage( Handle, wm_VScroll, MakeLong( sb_ThumbPosition, Info.nPos ),
0 );
      SetScrollInfo( Handle, sb_Vert, Info, True );
    end;
    Result := True;
```

```
end;

constructor TReingnDirectoryTree.Create( AOwner: TComponent );
begin
  inherited Create( AOwner );

  ReadOnly := True;

  Width := 250;
  Height := 150;

  FObjInst := nil;
  FOldWndProc := nil;
  HideSelection := False;
  FSaveDirectory := '';
  FUpdating := False;

  OnEditing := EditingHandler;
  OnEdited := EditedHandler;
  inherited OnDeletion := DeleteItemHandler;

  FImages := TImageList.Create( Self );
  Images := FImages;
  FShowHiddenDirs := False;
  FOpenCurrentDir := False;
end;


procedure TReingnDirectoryTree.CreateWindowHandle( const Params:
TCreateParams );
begin
  inherited CreateWindowHandle( Params );

  if not ( csDesigning in ComponentState ) and ( GetParentForm( Self ) <> nil
) then
  begin
    FFormHandle := ValidParentForm( Self ).Handle;
    FObjInst := Classes.MakeObjectInstance( FormWndProc );
    FOldWndProc := TFarProc( SetWindowLong( FFormHandle, gwl_WndProc,
Longint( FObjInst ) ) );
  end;
end;


procedure TReingnDirectoryTree.CreateWnd;
begin
  inherited CreateWnd;
  InitImageList;
  InitView;

  if not ( csDesigning in ComponentState ) then
  begin
    if FSaveDirectory <> '' then
    begin
      if not FRecreating then
        SetDirectory( FSaveDirectory );
      FSaveDirectory := '';
    end
    else
      SetDirectory( GetCurrentRootDir );
  end;
end;


procedure TReingnDirectoryTree.InitImageList;
var
  DirInfo: TSHFileInfo;
begin
  FImages.Handle := SHGetFileInfo( '', 0, DirInfo, SizeOf( DirInfo ),
```

```
                                        shgfi_SysIconIndex or shgfi_SmallIcon or
shgfi_Icon );
    FImages.ShareImages := True;

    FFolderClosedIconIndex := DirInfo.iIcon;

    SHGetFileInfo( '', 0, DirInfo, SizeOf( DirInfo ),
                   shgfi_OpenIcon or shgfi_SysIconIndex or shgfi_SmallIcon or
shgfi_Icon );
    FFolderOpenIconIndex := DirInfo.iIcon;
  end;



  destructor TReingnDirectoryTree.Destroy;
  begin
    ClearTree;
    FImages.Free;
    inherited Destroy;
  end;


  procedure TReingnDirectoryTree.DestroyWindowHandle;
  begin
    if FFormHandle <> 0 then
    begin
      { Restore original window procedure for parent form }
      SetWindowLong( FFormHandle, gwl_WndProc, Longint( FOldWndProc ) );
      Classes.FreeObjectInstance( FObjInst );
      FOldWndProc := nil;
    end;
    inherited DestroyWindowHandle;
  end;

  procedure TReingnDirectoryTree.DestroyWnd;
  begin
    FSaveDirectory := Directory;
    inherited DestroyWnd;
  end;


  procedure TReingnDirectoryTree.Loaded;
  begin
    inherited Loaded;
    InitView;

    if not ( csDesigning in ComponentState ) then
    begin
      if FOpenCurrentDir then
        SetDirectory( GetCurrentDir )
      else
        SetDirectory( GetCurrentRootDir );
    end;
  end;


  procedure TReingnDirectoryTree.Notification( AComponent: TComponent;
Operation: TOperation );
  begin
    inherited Notification( AComponent, Operation );
    if Operation = opRemove then
    begin
      if AComponent = FFileList then
        FFileList := nil
      else if AComponent = FDirLabel then
        FDirLabel := nil;
    end
    else if Operation = opInsert then
    begin
```

```
 if ( AComponent is TReingnTableBox ) and not Assigned( FFileList ) then
        FFileList := TReingnTableBox( AComponent );
    end;
  end;


function TReingnDirectoryTree.NodeHasData( Node: TTreeNode ): Boolean;
  begin
    Result := ( Node <> nil ) and ( Node.Data <> nil );
  end;

procedure TReingnDirectoryTree.AddFolderInfoToNode( Node: TTreeNode; const
NodePath: string;
  IconIndex: Integer );
  var
    FolderInfo: PFolderInfo;
    FileInfo: TSHFileInfo;
  begin
    if Node = nil then
      Exit;
    FolderInfo := New( PFolderInfo );
    FolderInfo^.FullPath := NodePath;
    FolderInfo^.ProcessedChildren := False;
    Node.Data := FolderInfo;
    Node.ImageIndex := IconIndex;
    if IconIndex = FFolderClosedIconIndex then
      Node.SelectedIndex := FFolderOpenIconIndex
    else
    begin
      SHGetFileInfo( PChar( PFolderInfo( Node.Data )^.FullPath ), 0,
                     FileInfo, SizeOf( TSHFileInfo ),
                     shgfi_SysIconIndex or shgfi_OpenIcon or shgfi_SmallIcon );
      Node.SelectedIndex := FileInfo.iIcon;
    end;
  end;


function TReingnDirectoryTree.GetPathFromNode( Node: TTreeNode ): string;
  begin
    if ( Node <> nil ) and ( Node.Data <> nil ) then
      Result := PFolderInfo( Node.Data ).FullPath
    else
      Result := '';
  end;

  function TReingnDirectoryTree.CanExpand( Node: TTreeNode ): Boolean;
  var
    OldCursor: TCursor;
    ChildNode: TTreeNode;
  begin
    OldCursor := Screen.Cursor;
    Screen.Cursor := crHourGlass;
    try
      ProcessChildren( Node );
      if not ( csDesigning in ComponentState ) and ( Node <> nil ) then
      begin
        ChildNode := Node.GetFirstChild;
        while ChildNode <> nil do
        begin
          AddTempNodeIfHasChildren( ChildNode );
          ChildNode := Node.GetNextChild( ChildNode );
        end;
      end;
    Result := inherited CanExpand( Node );
  end;

  procedure TReingnDirectoryTree.SetShowHiddenDirs( Value: Boolean );
  begin
    if FShowHiddenDirs <> Value then
```

```
    begin
      FShowHiddenDirs := Value;
      RefreshTree;
    end;
  end;

  function TReingnDirectoryTree.CanChange( Node: TTreeNode ): Boolean;
  begin
    Result := inherited CanChange( Node );
    FOld := Drive;
  end;

  procedure TReingnDirectoryTree.Delete( Node: TTreeNode );
  begin
    inherited Delete( Node );

    if NodeHasData( Node ) then
    begin
      PFolderInfo( Node.Data )^.FullPath := '';
      Dispose( PFolderInfo( Node.Data ) );
      Node.Data := nil;
    end;
  end;

procedure TReingnDirectoryTree.DeleteItemHandler( Sender: TObject; Node:
TTreeNode );
  begin
    if NodeHasData( Node ) then
    begin
      PFolderInfo( Node.Data )^.FullPath := '';
      Dispose( PFolderInfo( Node.Data ) );
      Node.Data := nil;
    end;

    if Assigned( FOnDeletion ) then
      FOnDeletion( Sender, Node );
  end;

  procedure TReingnDirectoryTree.KeyDown( var Key: Word; Shift: TShiftState );
  begin
    inherited KeyDown( Key, Shift );
    if Key = vk_F2 then
      Selected.EditText;
  end;


  procedure TReingnDirectoryTree.EditingHandler( Sender: TObject; Node:
TTreeNode; var AllowEdit: Boolean );
  begin
    if Node = nil then
      AllowEdit := False
    else if Node.Level = 0 then
      AllowEdit := False;
  end;

  procedure TReingnDirectoryTree.ResetNode( Node: TTreeNode );
  begin
    if Node <> nil then
    begin
      Node.DeleteChildren;
      if NodeHasData( Node ) then
        PFolderInfo( Node.Data )^.ProcessedChildren := False;
      AddTempNodeIfHasChildren( Node );
    end;
  end;


  function TReingnDirectoryTree.HaveProcessedChildren( Node: TTreeNode ):
Boolean;
```

```
  begin
    if NodeHasData( Node ) then
      Result := PFolderInfo( Node.Data )^.ProcessedChildren
    else
      Result := False;
  end;


  procedure TReingnDirectoryTree.FormWndProc( var Msg: TMessage );
  begin
    if Msg.Msg = wm_DeviceChange then
    begin
      Msg.Result := 0;
      UpdateActives;
      Change( Selected );
    end
    else if FOldWndProc <> nil then
  Msg.Result := CallWindowProc( FOldWndProc, FFormHandle, Msg.Msg, Msg.WParam,
 Msg.LParam );
  end;


  procedure TReingnDirectoryTree.UpOneLevel;
  begin
    if Selected <> nil then
    begin
      if Selected.Parent <> nil then
        Selected := Selected.Parent;
    end;
  end;

  procedure TReingnDirectoryTree.SetDTypes( Value: tDTypes );
  begin
    if FTypes <> Value then
    begin
      FTypes := Value;
      UpdateActives;
    end;
  end;

procedure TReingnDirectoryTree.SetFileList(Value:TReingnTableBox );
  begin
    if FFileList <> nil then
      FFileList.FDirTree := nil;
    FFileList := Value;
    if FFileList <> nil then
    begin
      FFileList.FDirTree := Self;
      FFileList.FreeNotification( Self );
    end;
  end;

  procedure TReingnDirectoryTree.SetDirLabel( Value: TLabel );
  begin
    FDirLabel := Value;
    if Value <> nil then
      Value.FreeNotification( Self );
    SetDirLabelCaption;
  end;

  procedure TReingnDirectoryTree.SetDirLabelCaption;
  var
    DirWidth: Integer;
  begin
    if FDirLabel <> nil then
    begin
      DirWidth := Width;
      if not FDirLabel.AutoSize then
        DirWidth := FDirLabel.Width;
```

```
      FDirLabel.Caption := MinimizeName( Directory, FDirLabel.Canvas, DirWidth
);
    end;
  end;

  constructor TReingnDirectoryListBox.Create( AOwner: TComponent );
  begin
    inherited Create( AOwner );
    FShowLongNames := True;

    FCanvas := TControlCanvas.Create;
    TControlCanvas( FCanvas ).Control := Self;

    FFormColor := clBtnShadow;
    FFormController := nil;
    FFormFlat := False;
    FFormFlatStyle := fsStatus;
    FFormFocusStyle := fsLowered;
    FFormSides := sdAllSides;
    FFormStyle := fsFlat;
    FFormVisible := False;
    FUseFormController := True;
  end;

  destructor TReingnDirectoryListBox.Destroy;
  begin
    if FFormController <> nil then
      FFormController.RemoveControl( Self );
    FCanvas.Free;
    inherited Destroy;
  end;


  procedure TReingnDirectoryListBox.BuildList;
  var
    D: string;
  begin
    D := Directory;
    while not DirectoryExists( D ) do
      D := ExtractFilePath( D );
    Directory := D;

    inherited BuildList;
  end;

  procedure TReingnDirectoryListBox.SetShowLongNames( Value: Boolean );
  begin
    if FShowLongNames <> Value then
    begin
      FShowLongNames := Value;
      BuildList;
      UpdateDirLabel;
    end;
  end;

  function TReingnDirectoryListBox.GetLongDirName: string;
  begin
    Result := LongPathFromShort( Directory );
  end;


  procedure TReingnDirectoryListBox.Change;
  begin
    inherited Change;
    UpdateDirLabel;
  end;

  function TReingnDirectoryListBox.GetColor: TColor;
  begin
```

```
  Result := inherited Color;
end;

procedure TReingnDirectoryListBox.SetColor( Value: TColor );
begin
  if Color <> Value then
  begin
    inherited Color := Value;
    if FFormVisible then
      RepaintForm;
  end;
end;


function TReingnDirectoryListBox.NotUsingController: Boolean;
begin
  Result := ( FFormController = nil ) or not FUseFormController;
end;


procedure TReingnDirectoryListBox.SetFormColor( Value: TColor );
begin
  if FFormColor <> Value then
  begin
    FFormColor := Value;
    RepaintForm;
  end;
end;


procedure TReingnDirectoryListBox.SetFormController( Value:
TReingnFormController );
begin
  if FFormController <> nil then
    FFormController.RemoveControl( Self );
  FFormController := Value;
  if Value <> nil then
  begin
    Value.AddControl( Self );
    Value.FreeNotification( Self );
  end;
end;


procedure TReingnDirectoryListBox.SetFormFlat( Value: Boolean );
begin
  if FFormFlat <> Value then
  begin
    FFormFlat := Value;
    if FFormFlat then
    begin
      FormVisible := True;
      if not ( csLoading in ComponentState ) then
      begin
        FFormSides := sdAllSides;
        FFormStyle := FFormFlatStyle;
      end;
    end;
    RepaintForm;
    Invalidate;
  end;
end;

procedure TReingnDirectoryListBox.SetFormSides( Value: TSides );
begin
  if FFormSides <> Value then
  begin
    FFormSides := Value;
    RepaintForm;
```

```
    end;
  end;


  procedure TReingnDirectoryListBox.SetFormStyle( Value: TFormStyle );
  begin
    if FFormStyle <> Value then
    begin
      FFormStyle := Value;
      RepaintForm;
    end;
  end;


  procedure TReingnDirectoryListBox.SetFormVisible( Value: Boolean );
  begin
    if FFormVisible <> Value then
    begin
      FFormVisible := Value;
      if FFormVisible then
        Ctl3D := True;
      RecreateWnd;
    end;
  end;

  procedure TReingnDirectoryListBox.RepaintForm;
  var
    R: TRect;
  begin
    R := Rect( 0, 0, Width, Height );
    RedrawWindow( Handle, @R, 0, rdw_Invalidate or rdw_UpdateNow or rdw_Form );
  end;


procedure TReingnDirectoryListBox.WMNCPaint( var Msg: TWMNCPaint );
  var
    DC: HDC;
  begin
    inherited;

    if FFormVisible then
    begin
      DC := GetWindowDC( Handle );
      FCanvas.Handle := DC;
      try
        DrawForm( FCanvas, Width, Height, FFormStyle, Color, FFormColor,
FFormSides );
      finally
        FCanvas.Handle := 0;
        ReleaseDC( Handle, DC );
      end;
      Msg.Result := 0;
    end;
  end;

  procedure TReingnDirectoryListBox.MouseLeave;
  begin
    if Assigned( FOnMouseLeave ) then
      FOnMouseLeave( Self );
  end;

  procedure TReingnComboBox.SetDTypes( Value: tDTypes );
  begin
    if FTypes <> Value then
    begin
      FTypes := Value;
      RecreateWnd;
    end;
  end;
```

```
procedure TReingnDriveComboBox.SetFlatButtons( Value: Boolean );
begin
  if FFlatButtons <> Value then
  begin
    FFlatButtons := Value;
    Invalidate;
  end;
end;


function TReingnDriveComboBox.NotUsingController: Boolean;
begin
  Result := ( FFormController = nil ) or not FUseFormController;
end;


procedure TReingnDriveComboBox.SetFormColor( Value: TColor );
begin
  if FFormColor <> Value then
  begin
    FFormColor := Value;
    Invalidate;
  end;
end;


procedure TReingnDriveComboBox.SetFormController( Value:
TReingnFormController );
begin
  if FFormController <> nil then
    FFormController.RemoveControl( Self );
  FFormController := Value;
  if Value <> nil then
  begin
    Value.AddControl( Self );
    Value.FreeNotification( Self );
  end;
end;


procedure TReingnDriveComboBox.SetFormFlat( Value: Boolean );
begin
  if FFormFlat <> Value then
  begin
    FFormFlat := Value;
    if FFormFlat then
    begin
      FormVisible := True;
      if not ( csLoading in ComponentState ) then
      begin
        FFormSides := sdAllSides;
        FFormStyle := FFormFlatStyle;
      end;
    end;
    Invalidate;
  end;
end;

procedure TReingnDrive.SetFormSides( Value: TSides );
begin
  if FFormSides <> Value then
  begin
    FFormSides := Value;
    Invalidate;
  end;
end;

procedure TReingnDrive.CMEnter( var Msg: TCMEnter );
begin
```

```
  inherited;
  UpdateForm( False, True );
end;

procedure TReingnDrive.CMExit( var Msg: TCMExit );
begin
  inherited;
  FOverControl := False;
  UpdateForm( False, False );
end;

procedure TReingnDrive.WMSize( var Msg: TWMSize );
begin
  inherited;
  if FFormVisible then
    Invalidate;
end;

end.
```

**Файл Unit5.pas**

```
unit Unit5;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus, ExtCtrls, jpeg;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Memo1: TMemo;
    Panel1: TPanel;
    N11: TMenuItem;
    Button1: TButton;
    Button2: TButton;
    Button4: TButton;
    Image1: TImage;
    Memo3: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Memo2: TMemo;
    Panel2: TPanel;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    Label6: TLabel;
    Label7: TLabel;
    Button3: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```