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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем”

Виконав здобувач вищої освіти
IV курсу, групи КІ-19
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Коптєв Р.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Кафедра_КбПЗ_2023рік

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Коптєву Руслану Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 7-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

Кафедра_КібПЗ_2023рік

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Коптєв Р.В.
(прізвище та ініціали)

Кафедра_КбПЗ_2023рік

АНОТАЦІЯ

Коптєв Р.В. Програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

Метою розробки є програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

Результат роботи – програмна реалізація системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

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

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

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

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: комп'ютерна інженерія, реінжиринг коду, модернізація інформаційних та комп'ютерних систем

Кафедра_КОПЗ_2023рік

ABSTRACT

Koptiev R.V. Code reengineering system software for modernization of information and computer systems. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the code re-engineering system in the modernization of information and computer systems.

The goal of the development is the code reengineering system software for the modernization of information and computer systems.

The result of the work is the software implementation of the code reengineering system for the modernization of information and computer systems.

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

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

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

The program was developed in the Delphi 10.4 environment.

Keywords: computer engineering, code reengineering, modernization of information and computer systems

КафеДра_КОПЗ_2023рік

ЗМІСТ

Кафедра_КОПЗ_2023рік

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AES – Advanced Encryption Standard – алгоритм шифрування

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)

ЕЦП – електронний цифровий підпис

КД – ключова дискета

НСД – несанкціонований доступ

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

Кафедра_КОПЗ_2023рік

ВСТУП

Актуальність теми. Існує певний клас великих програмних комплексів, написаних на старих мовах програмування, таких як Кобол, ПЛ/1, Фортран і ін., які функціонують десятиріччями на стрімко застаріваючому встаткуванні (по-англійському такі системи називають legacy systems). Більшу частину їх представляють, так звані, бізнес-додатки, зайняті у фінансовій сфері промисловості. По дослідницькій оцінці 80% з них написано мовою Кобол, що являє собою близько 180 мільярдів рядків, а вкладені кошти, включаючи вартість устаткування й витрати на навчання обслуговуючого персоналу, оцінюються сумою більше 3 трильйонів доларів [1-10]. Головна проблема таких додатків – дуже висока вартість супроводу. Це відбувається в силу наступних причин: наявність дуже громіздкого програмного коду, часто погано документованого; необхідності перевірки всього додатка при зміні структури даних [21]; ерозії структури, що робить кожна наступна зміна дорожче [16]; старіння встаткування; недоліку кваліфікованого персоналу.

Дані програмні комплекси не здатні швидко адаптуватися до нових технологій, що появились у результаті науково-технічного прогресу, таким як Internet-технології, об’єктно-орієнтоване програмування, розподілена об'єктна архітектура, графічний інтерфейс користувача й ін. Крім того, устаткування цих комплексів має високе відношення ціни до продуктивності в порівнянні із сучасними рішеннями на основі розподілених систем. У теперішній час істотний інтерес викликає такий спосіб рятування старих додатків від вищезгаданих недоліків, як автоматизований переклад на нові мови програмування й апаратні платформи. Одним із завдань, що виникають при реалізації даного рішення, є побудова динамічної підтримки (runtime support). Оскільки при перекладі необхідно зберегти структуру додатка, використовуючи високорівневі мови

Кафедра_КОПЗ_2023рік

програмування, коло розв'язуваних питань для динамічної підтримки істотно відрізняється в порівнянні із традиційним компіляторним підходом [14, 18].

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

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

– Огляд існуючих систем реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

– Дослідження системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

– Програмна реалізація системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КПЗ_2023рік

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Визначення поняття реінжинірингу і його складових

Для відповіді на питання про необхідність динамічної підтримки приведемо визначення предметної області на основі огляду літератури. У роботі [23] дане коротке визначення реінжинірингу як процесу трансформації додатку для його відповідності до нових вимог. Більш повно він описаний як діяльність, спрямована на збільшення розуміння й/або поліпшення додатку для кращого супроводу, можливості повторного використання й здатності до розвитку [27]. Реінжиніринг – дослідження й зміна існуючої системи для відтворення її в новій формі. Даний процес являє собою комбінацію етапів, таких як:

– зворотне проектування;
– реструктуризація (нормалізація);
– редокументування;
– пряме проектування;
– переорієнтація системи [25].

Для цілей подальшого викладу буде використано саме це розгорнуте визначення, оскільки воно розкриває функціональні частини предметної області. Приведемо також визначення використаних термінів.

Зворотне проектування – процес розуміння, аналізу й абстракції системи в новій формі на більш ранньому етапі її життєвого циклу. Пряме проектування – діяльність, що використовує застарілий додаток, результати зворотного проектування й нові вимоги для одержання нової системи. Редокументування – аналіз системи для виробництва супровідної документації в різному виді, включаючи посібник користувача й оформлення вихідних кодів системи. Реструктуризація – процес трансформації системи з однієї форми подання в іншу

Кафедра_КОПЗ_2023рік

зі збереженням рівня абстракції й функціонального поводження. Переорієнтація – трансформація й переклад існуючої системи в нову конфігурацію, наприклад такий як нова ОС або нове обладнання [25]. Приведемо визначення деяких термінів, використаних у роботі й взаємозалежних з поняттям реінжинірингу. Реінжиніринг даних – процес застосування методів реінжинірингу для даних. Прикладом такої діяльності може служити переклад файлів даних старих систем у реляційні бази даних. Реінжиніринг бізнес-процесу – фундаментальне переосмислення й радикальна зміна роботи додатка для досягнення більших змін у продуктивності. Трансляція вихідних текстів – трансформація програми з однієї мови програмування або з одного діалекту в іншій [25].

Відзначимо, що на початку процесу реінжинірингу старої системи представляється необхідним застосовувати глобальний аналіз і оцінку додатка: інвентаризацію, планування й оцінка ресурсів, аналіз частин (перевірка на коректність компонентів, наявність недозволених зовнішніх посилань і ін.). Як показано в роботі [29], у процесі проведення таких дій ставиться діагноз системи, визначаються частини, що вимагають заміни, і виробляється загальна стратегія трансформації. При проведенні реінжинірингу складові етапи проводяться в певному порядку: зворотне проектування, потім пряме (якщо використовуються) і ін., при цьому можливо одночасне виконання деяких процесів, наприклад, реструктуризації й переорієнтації. Детальний розгляд всіх аспектів і сценаріїв реінжинірингу виходить за рамки даної роботи.

1.2 Область застосування

Необхідність динамічної підтримки для завдань реінжинірингу

Далі складові процеси реінжинірингу будуть розглянуті менш формально для з'ясування необхідності побудови для них динамічної підтримки.

Ціль зворотного проектування – добування з вихідних текстів додатка знання про предметну область у вигляді структурної інформації або

Кафедра_КПЗ_2023рік

специфікацій. Тому його використовують як відправну точку для інших етапів, у першу чергу для прямого проектування. Цей процес, у свою чергу, представляє ітерацію життєвого циклу додатка – побудова специфікацій, проектування й розробка з урахуванням нових вимог і використанням інформації попереднього етапу. Реструктуризація й редокументування можуть застосовуватися, як окремо у вигляді, наприклад, створення легко читаємого й структурованого коду , так і комплексно з вищезгаданими етапами. Приклад останнього способу реінжинірингу описаний у роботі [23]. Реінжиніринг бізнес-процесу являє собою серйозну зміну логіки роботи додатка, але він, також як і попередні етапи, не має потреби в побудові якої-небудь підтримки часу виконання при використанні, тому що не робить зміни середовища існування або виконання системи. Проведення переорієнтації в процесі реінжинірингу дозволяє використовувати в трансформованому додатку нові технології. Це досягається за рахунок зміни середовища існування системи. У випадку істотної трансформації додатку необхідна побудова динамічної підтримки, як для збереження функціональності, так і для використання нових можливостей. Більш докладно дане питання розглянуте в наступних параграфах даної роботи.

Відзначимо можливість застосування нових технологій без використання переорієнтації за рахунок заміни інструментальних засобів. Це досягається за допомогою розширення вихідної мови програмування додатка й може бути реалізовано низькорівневим перекладом в іншу мову. В останнього випадку також необхідна побудова динамічної підтримки.

Основними перевагами способу відновлення інструментальних засобів є максимальна дешевина й простота використання, а недоліками – ще більш ускладнений синтаксис і збереження «особливостей» вихідних систем [11].

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КОПЗ_2023рік

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Найпоширеніші способи переорієнтації

Як було показано, з п’яти компонентів процесу реінжинірингу тільки процес переорієнтації потребує динамічної підтримки, тому розглянемо його найпоширеніші варіанти. Далі будемо вважати, що додаток є сукупністю трьох компонентів: користувальницького інтерфейсу, даних (включаючи доступ до даних) і програмної логіки. Переорієнтацію різних частин будемо вважати незалежним. Відразу відзначимо, що огляд реінжинірингу даних виходить за рамки даної роботи. Про деякі аспекти даної проблеми можна прочитати в роботі [12]. Крім того, для старих додатків є характерним порівняльна простота користувальницького інтерфейсу, тому акцент при розгляді переорієнтації робиться на застосування до програмної логіки, як до більше важкого завдання.

Виділимо наступні методи переорієнтації:

1. Створення інтерфейсної оболонки [21]. Цей метод складається з наступних частин: аналіз виконуваних додатком або його частиною функцій, створення об'єктної оболонки, що приховує стару систему й надає виявлений інтерфейс для її клієнтів.

2. Переклад частини або всього додатка на інші мови [11]. Даний підхід складається із трансляції вихідних текстів, при цьому попередньо можуть виконуватися інші процеси реінжинірингу, наприклад, пряме й зворотне проектування. При цьому для використання нових технологій немає необхідності розширення мови програмування.Це досягається в силу зміни середовища існування системи засобами динамічної підтримки.

Кафедра_КПЗ_2023рік

Необхідність застосування якого-небудь із вищевказаних методів з'ясовується після проведення глобального аналізу системи, при цьому найбільш ефективним може є використання комбінованого підходу, з можливою заміною частин додатка [29]. Далі розглянемо ці методи більш детально у світлі динамічної підтримки.

Динамічна підтримка для методу інтерфейсної оболонки

Даний метод застосовується до програмної логіки. Він являє собою різновид способу трансформації додатка методом чорного ящика, тобто не проводячи глибокого аналізу додатка. Тому система, що піддається подібній операції, повинна добре функціонувати й виконувати точно певні дії із чітким зовнішнім інтерфейсом. По цій же причині вартість реалізації такого рішення досить низька [29]. Як відзначено в роботі [21], його використання дозволяє співіснувати старим і новим частинам системи в процесі перекладу.

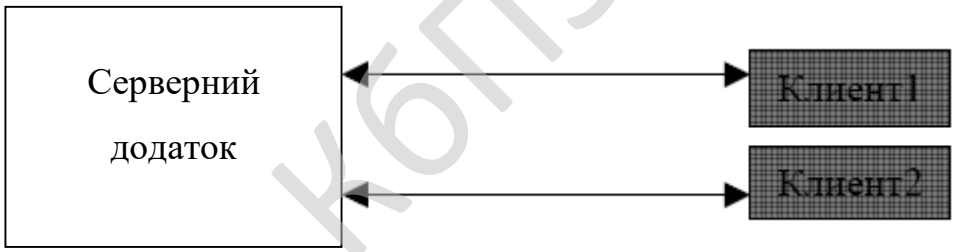


Рисунок 2.1 – Стара модель системи

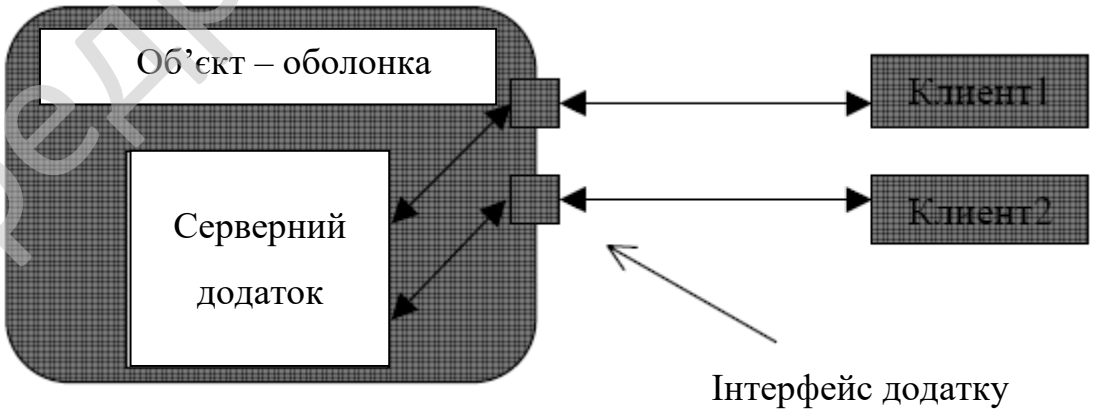


Рисунок 2.2 – Нова модель системи

Кафедра_КбПЗ_2023рік

Для з'ясування інтерфейсу системи із клієнтами проводиться аналіз системи на наявність баз даних і експортованих функцій. Після виконання даного етапу відбувається створення об'єктної інтерфейсної оболонки. Сутність трансформації наведена на рисунку 2.1 і 2.2.

При такому способі переорієнтації використовуються наступні рішення в дусі нових технологій:

1. Приховання даних і реалізації (у вигляді старої системи) усередині створеного об'єкта-оболонки як об’єктно-орієнтованому підході.

2. Наявність проміжного шару між клієнтом і сервером як у трьохрівневій архітектурі.

3. Здатність взаємодії за допомогою сучасних мережних протоколів (вихід на Internet/Intranet – технології)

Таким чином, отриманий додаток являє собою типову розподілену об'єктну систему. При цьому можливо залишити функціональний інтерфейс у старому середовищі (це може бути необхідним для співіснування старих і нових частин у великій системі). Крім того, можна змінити його для нового середовища, що дозволить різко розширити область застосування трансформованого додатка, наприклад для інших розподілених систем і/або Internet. В останньому випадку необхідно також змінити або переписати його клієнтів.

З узагальненої точки зору, доцільно винести в динамічну підтримку тільки примітиви взаємодії старої системи з оболонкою й оболонки із зовнішнім миром. При цьому обсяг останніх буде невеликий, а для перших він пропорційний розміру інтерфейсу.

Прикладом реалізації даного способу може служити інтерфейс ECI фірми IBM, що дозволяє виконувати програми на віддалених ЕОМ типу mainframe.

Динамічна підтримка для методу перекладу на інші мови

Цей метод може застосовуватися як до програмної логіки, так і до користувальницьких компонентів. Як ми вже відзначали, трансформація останніх не представляє серйозної проблеми. Стосовно до програмної логіки, по

Кафедра_КОПЗ_2023рік

виконуваних етапах до проведення переорієнтації системи, можна розглянути два підходи: «чорного» ящика й «білого» ящика [29].

Сутність першого варіанта: використовуючи такі ж обмеження на додаток, що трансформується, як і для методу оболонки, зробити переклад його вихідних текстів на інші мови. У цьому випадку також немає необхідності в глибокому аналізі додатка, тому даний процес реінжинірингу може бути значною мірою автоматизований [11].

Другий підхід проводиться після проведення зворотного проектування як заключна частина прямого проектування. Опишемо попередні етапи. Ключовою деталлю в даному підході є розуміння програми за допомогою аналізу вихідних текстів [24]:

– виділення компонентів системи разом з їхніми залежностями, використовуючи відповідні механізми виділення;

– пошук проектної інформації, створення абстракцій системи, а також методи об'єктно-орієнтованого моделювання [21]: побудова концептуальної моделі предметної області й ін.

Відзначимо що в порівнянні з попереднім підходом, для даного необхідно набагато більша участь персоналу, що займається рішенням даної проблеми, хоча й тут використання автоматизації для частин процесу може бути дуже продуктивно [28].

Розглянемо питання про динамічну підтримку в цих випадках. При використанні першого підходу необхідно побудувати динамічну підтримку, що відбиває функціональне поводження старого середовища в новій. Це може представляти істотний обсяг роботи при:

– сильному розходженні середовищ;

– необхідності точної відповідності поводження додатка до й після трансформації;

Кафедра_КбПЗ_2023рік

– відсутність вимоги мінімізації супровідного коду. Архітектура й зовнішній вигляд нової системи перебувають під впливом останніх двох факторів.

Приклад побудови системи динамічної підтримки для даного випадку наданий у практичній частині даної роботи.

У другому випадку все залежить від ступеня абстракції системи, наприклад, можна переробити додаток, використовуючи відповідні засоби об'єктного моделювання, так, що воно навіть на рівні реалізації буде незалежно від вихідного додатка. Використання після цього засобів занурення в нове середовище дозволить одержати нову систему, що виконує ті ж функції. Подібна трансформація може відбуватися при реінжинірингу бізнесу-процесу. При такому сценарії існує необхідність у динамічній підтримці тільки для абстракцій, використаних при моделюванні. Як правило, її обсяг невеликий при зануренні в об’єктно-орієнтоване середовище, тому що середовище моделювання буде родинне з останім. Зазначений випадок є виключенням із практики для великих систем у силу їхнього розміру. У переважній більшості випадків практичного перекладу виходить синтез між даною ситуацією й попередньої (у першому підході) з більшим ухилом до останньої. Із цієї причини розумно вважати, що обсяг і функції динамічної підтримки в цьому випадку залишаються такими ж.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Кафедра_КОПЗ_2023рік

чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

Кафедра_КбПЗ_2023рік

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

Кафедра_КОПЗ_2023рік

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

Кафедра_КБПЗ_2023рік

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

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

Кафедра_КОПЗ_2023рік

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

Кафедра_КОПЗ_2023рік

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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.

Кафедра_КОПЗ_2023рік

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.

Кафедра_КбПЗ_2023рік

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Завдання динамічної підтримки у світлі особливостей вихідних додатків мовою Кобол

Мова програмування Кобол (Common Business Oriented Language) увійла у промислове використання в 1961р. Її створення переслідувало кілька цілей:

– полегшити процес перепрограмування при переносі програм з одного типу ЕОМ на іншій;

– спростити внесення в програму дрібних змін для обробки комерційної інформації;

– зменшити витрати часу на написання й налагодження нових програм [7].

За своє довге життя вона піддалася численним змінам, тому на сьогоднішній день існує більше 16 її різновидів.

Як показано в роботі [19], існує ряд особливо важливих можливостей мови Кобол, що роблять її особливо привабливою для використання в бізнес додатках:

– Робота з різнорідними структурними даними, тоді як для наукових додатків характерна робота із плаваючою точкою, а для системних із цілими числами.

– Десяткова арифметика заданої точності.

– Побудова зручних звітів.

– Обробка великих обсягів даних, що складаються з різнорідних структурних записів.

Програма може мати у своєму тексті включення на інших мовах, таких як IMS, CICS, SQL і ін., які заміщаються при препроцесуванні на зовнішні виклики.

Кафедра_КБПЗ_2023рік

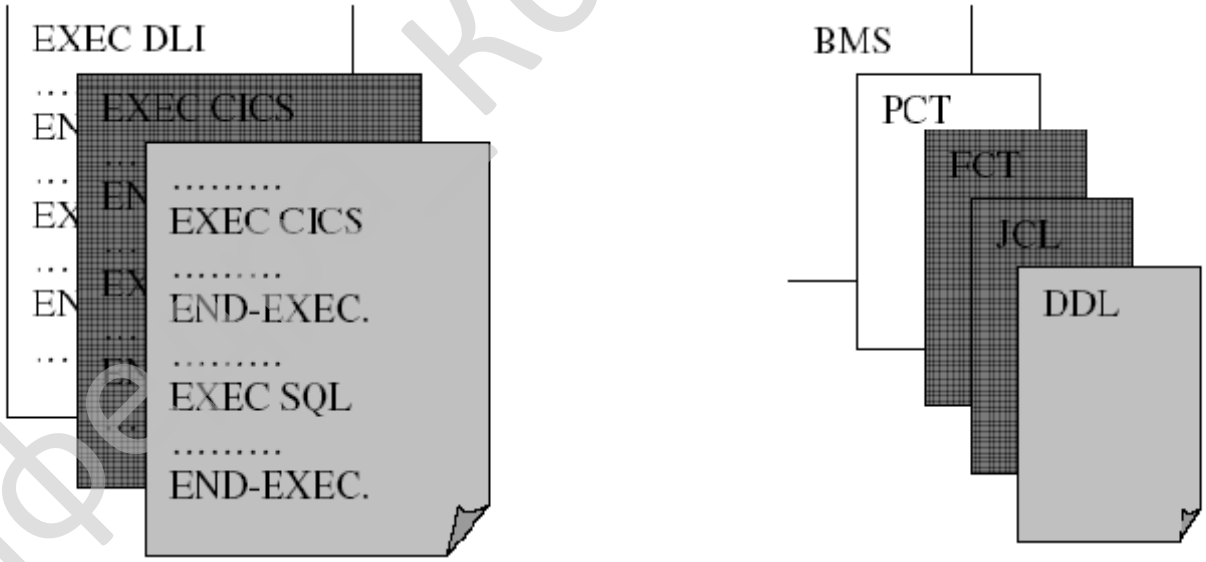
Крім того, типовий додаток (див. рисунок 3.1) складається не тільки із сотень таких програм, але набору зовнішніх сервісних модулів на JCL, DDL, BMS і ін. Опишемо коротко цілі використання різних модулів і включень:

– DDL (Data Definition Language) – сценарії опису бази дані додатків.

– SQL (Structured Query Language) і IMS (Information Management System) – робота з базами даних.

– Модулі на JCL (Job Control Language) використовуються для керування програмами, що підтримують працездатність додатка.

– Вставки з використанням CICS (Customer Information Control System) заслуговують окремого розгляду. Вони дозволяють програмі порівняно легко працювати з файлами (для їхнього опису використовуються модулі, називані FCT – File Control Table), спілкуватися з користувачем (для цього використовуються файли опису екранів, називаємі BMS – Basic Mapping Support), викликати інші програми (оголошені у файлах, називаних PCT – Program Control Table) і використовувати різні сервісні функції.



Програми на мові Кобол Зовнішні модулі

Рисунок 3.1 – Типовий додаток мовою Кобол.

Кафедра КбПЗ 2023рік

Опишемо завдання динамічної підтримки, ґрунтуючись на вищеописаних особливостях вихідних додатків:

1. Необхідність реалізації десяткової арифметики із заданою точністю.

2. Підтримка типів і операцій над ними. Основними труднощами тут є громіздкість типу елементарної змінної й невідповідність вихідних і цільових машинних типів. Так будь-яка елементарна змінна має два атрибути: маску редагування й клас зберігання. Оскільки існує близько 20 різних символів, що редагують, і 6 класів зберігання, необхідно розділити даний тип на трохи менші по функціональному принципу.

3. Програмне визначення типу кодування даних (EBCDIC або ASCII). Подання в пам'яті окремих типів змінних залежить від поточного кодування.

4. Можливість перевизначення фізичної пам'яті змінних, тобто наявності різних даних з однаковим зсувом.

5. Реалізація певної кількості убудованих функцій.

6. Передача параметрів при міжпрограмних викликах.

7. Підтримка для включень на інших мовах (SQL і CICS). Слід зазначити, що система CICS має більші можливості, тому повна її реалізація вимагає великого обсягу роботи.

Додаткові вимоги від динамічної підтримки

Приведемо ряд додаткових вимог, пропонованих до побудови переведеної програми і її динамічної підтримки часу. До них відносяться:

1. Для програмної логіки: реєнтерабельність, тобто здатність безконфліктного виконання коду декількома потоками керування одночасно в рамках одного адресного простору. Це дозволяє використовувати неї в серверних системах.

2. Підтримка нових можливостей, що з'явилися, для створення інтерфейсу й програмної логіки.

3. Прийнятна швидкість виконання.

4. Забезпечення комплексного перекладу програмної логіки й

Кафедра_КПЗ_2023рік

користувальницького інтерфейсу.

Далі реалізація цих вимог розглянута більш докладно.

Інтеграція динамічної підтримки із системою автоматизованого перекладу RescueXP

Конвертор RescueXP 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 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 (див. рисунок 3.2). Спочатку програма подається на вхід синтаксичному аналізатору мови Кобол. Коли зустрічається вставка на іншій мові, то він робить перемикання на

Кафедра_КОПЗ_2023рік

відповідний аналізатор. Результатом роботи цього етапу служить дерево, що складається зі специфічних для вхідних мов вузлів.

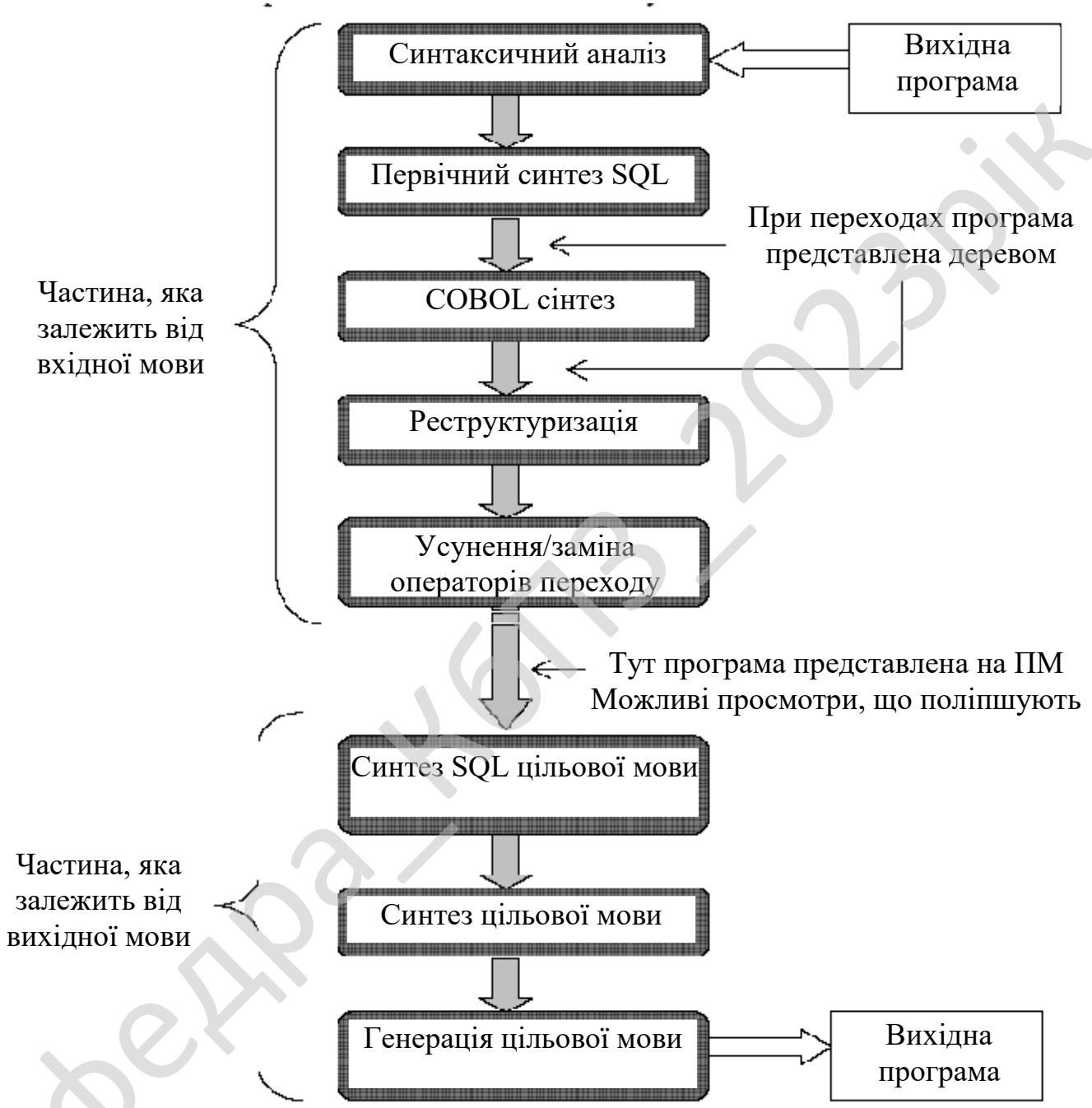


Рисунок 3.2 – Схема конвертора RescueXP для програмної логіки

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 Кобола роблять проміжне подання [2], у якому

Кафедра_КБПЗ_2023рік

залишилися деякі конструкції мови Кобол. 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.

Кафедра_КОПЗ_2023рік

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.

Динамічна підтримка інтерфейсних компонент

Опишемо підтримку часу виконання, побудовану для інтерфейсних компонент. Як уже було відзначено, тільки для деяких цільових мов довелося написати супровідний код.

Розглянемо пророблену роботу для інтерфейсних компонент мовою HTML

Коротко опишемо використовувані технологічні рішення. HTML (HyperText Markup Language) – мова для публікації гіпертекстової інформації, докладний опис можна знайти в специфікації [26]. Головною її особливістю є її структуроване подання за допомогою ярликів (tags). CSS (Cascading Style Sheet, специфікація в роботі [30]) – мова опису елементів подання (колір тексту, тип шрифту й ін.) для гіпертекстових документів. Javascript – об'єктно-орієнтована скриптова мова для програмування при відображенні сторінок HTML. DHTML – розширення цієї мови, що дозволяє динамічно змінювати подання вищевказаних сторінок.

Опишемо структуру інтерфейсних компонент. Каркас кожного компонента представлений мовою HTML, подійно-керований код написаний з використанням мови JavaScript, а атрибути подання зроблені із застосуванням технології CSS. У результаті отримані интерфейсні компоненти забезпечують чіткий поділ змісту й подання, що полегшує їхній супровід і модифікацію. Проте, використовуваних засобів не вистачає для побудови системи меню (щоз'являються під час процесу конвертації) за умови незалежності від типу використовуваного браузера Internet. Вихід був знайдений за допомогою написання мовою Javascript об'єктної бібліотеки, що використовує нестандартне розширення DHTML. Відмітна риса такого рішення в тому, що усередині цього модуля сховані істотні різниці

Кафедра_КОПЗ_2023рік

реалізації розширення. Таким чином, така интерфейсна компонента успішно працює на переважній більшості браузерів.

Відзначимо неможливість її використання без модифікації в серверах протоколу HTTP, тому що в них відсутня серверна програмна логіка, що залежить від типу використовуваного сервера. Рішення даної проблеми для конкретного сервера наведено нижче.

Розглянемо пророблену роботу для інтерфейсних компонент мовою Java

Підтримка часу виконання тут потрібна з кількох причин. По-перше, для забезпечення мінімального сервісу при роботі, наприклад, закриття створюваних вікон при відповідній дії. По-друге, через додаткові можливості, що з'являються в процесі конвертації: створення нестандартних елементів меню й ін. Крім того, у динамічній підтримці два варіанти класів: для стандартного набору віконних класів (AWT – Abstract Window Toolkit) і платформонезалежного (JFC – Java Foundation Classes). Побудовані компоненти успішно використовувалися для сценарію реінжинірингу з інтерфейсної оболонкою.

Заслуговує окремого розгляду питання підтримки часу виконання для компонентів серверної логіки (servlets). Технологія являє собою набір абстрактних класів і функцій програмного інтерфейсу для побудови систем працюючих за принципом запит/відповідь, прикладом можуть служити сервери протоколу HTTP. Компонента серверної логіки – клас мовою Java, написаний з використанням вищезгаданої технології, що представляє собою абстракцію рівня додатка. Запит до неї змушує її робити відповідь, де запит і відповідь – об'єкти, що відбивають терміни предметної області. На перший погляд, така система виглядає як об’єктно-орієнтована, так званий серверний CGI – додаток, але є й істотні відмінності:

– для виконання запиту використовується окремий потік керування, а не окремий процес;

– якщо відбувається кілька одночасних запитів до одного об'єкта, то

Кафедра_КПЗ_2023рік

(залежно від типу об'єкта) використовується або той же екземпляр, або будь-який інший даного типу з пула вільних;

– також питання ідентифікації сеансу (роботи одного клієнта із сервером) не лежить на компоненті.

Ці й інші переваги роблять використання таких серверних компонентів настільки популярним. У цьому випадку, кожний серверний компонент по запиту клієнта повинен повернути відповідний HTML файл. У первісному рішенні HTML-зміст перебував усередині серверного компонента. Із цієї причини вироблялися зупинки сервера при необхідності поміняти интерфейсне подання. Наступне рішення виявилося більше гнучким: був написаний клас динамічної підтримки для звертання до зовнішнього файлу змісту. При відповіді на запит він перевіряє дату зовнішнього файлу, якщо вона не збігається з датою збереженої в пам'яті копії, або вона ще не завантажена, то відбувається завантаження, після цього дані передаються клієнтові. Крім того, була додана можливість вставки у файл HTML – подання параметрів часу виконання. Це досягається за рахунок модифікованої структури файлу подання.

Рисунок 3.3 – Модифікована структура HTML файлу

Кафедра КБПЗ 2023рік

Під час завантаження HTML файлу виробляється розбір його на фрагменти по наявності коментарів спеціального виду. При обробці запиту клієнта, компонента серверної логіки витягає з універсальної адреси (URL), переданого їй, частину з параметрами, потім робить дії в такий спосіб. У циклі, поки не оброблений файл цілком, ділянка тексту без коментарів передається клієнтові, потім, якщо параметр із таким ім'ям присутній, передається його значення, а інакше використовується значення за замовчуванням.

Побудовані компоненти успішно використовувалися в практичних проектах реінжинірингу з перекладом програмної логіки.

Динамічна підтримка програмної логіки

Способи побудови динамічної підтримки для мови Java

Розглянемо питання реалізації динамічної підтримки для програмної логіки на прикладі мови Java. Java є об’єктно-орієнтованою, платформонезалежною мовою програмування зі збором сміття, призначена для виконання усередині віртуальної машини Java. Перелічимо головні його особливості:

– наявність вбудованої підтримки для потоків керування;

– безліч перевірок часу виконання (вихід за кордон масиву й ін.);

– відсутність можливості роботи з пам'яттю (у тому числі, немає посилань на елементарні типи);

– наявність механізму виключень;

– великої кількості убудованих функцій у вигляді набору класів;

– спосіб виконання – інтерпретація;

– відкомпільоване подання у вигляді байт-коду.

Для подолання проблеми повільного виконання використовуються компілятори часу виконання в машинний код [13].

Існує, принаймні, три способи побудови підтримки часу виконання, що відрізняються поданням елементарних даних вихідної щодо типів цільової мови:

1. Переклад елементарних типів в елементарні типи.
2. Переклад елементарних типів в об'єкти.

Кафедра_КОПЗ_2023рік

3. Змішаний переклад.

Оскільки першочерговою вимогою є збереження функціональності, то обраний другий варіант. Цьому сприяють наступні особливості вихідної мови в порівнянні з вихідним: невідповідність типів, наявність перевизначень, арифметики фіксованої точності й ін. **Цей підхід реалізований у даній роботі, далі він розглянутий у деталях**.

Торкнемося трохи інших варіантів: вони є більш вигідним вибором (наприклад, з погляду продуктивності або обсягу текстів підтримки часу виконання) за умови ігнорування або емуляції певних особливостей вихідної мови. При такому підході конвертація програми наближається до компіляції в цільову мову, як до більш низькорівневого процесу. При цьому вона обмежена необхідністю подальшого супроводу отриманої програми. Наприклад, якщо робити переклад, ґрунтуючись на першому варіанті, то, для досягнення функціональної еквівалентності отриманої вихідної програми, необхідно додавати форматуючі виклики для подання значень у старому виді. Це може знижувати виразність зовнішнього вигляду такої програми. Більш докладно це питання розглянуте в роботі [9].

Проте, обсяг підтримки часу виконання тим менше, чим більше використовуються елементарні типи. Але іноді це може приводити до серйозних втрат у продуктивності, наприклад, у мові Java, більшість змінюваних класів робить синхронізацію методів для здійснення реєнтерабельніоті, що не завжди виправдано. А велике використання незмінних об'єктів приводить до частих припинень виконання внаслідок «складань сміття». Із цієї причини представляється найбільш доцільним використовувати змішану стратегію.

У цей момент зроблена розробка підтримки часу виконання з використанням елементарних типів для обмеженого набору тестів. Домінуючим фактором тут є зовнішній вигляд програми. У даних умовах поки не знайдено прийнятного рішення для підтримки арифметики фіксованої точності й інших особливостей вхідної мови. Із цієї причини цей варіант далі не розглядається.

Кафедра_КбПЗ_2023рік

Рішення динамічною підтримкою завдань 1-7 вихідного додатка.

Далі розглядаються рішення завдань вихідного додатка мовою Кобол у даній системі динамічної підтримки для мови Java. Її структура наведена на рисунку. 3.4

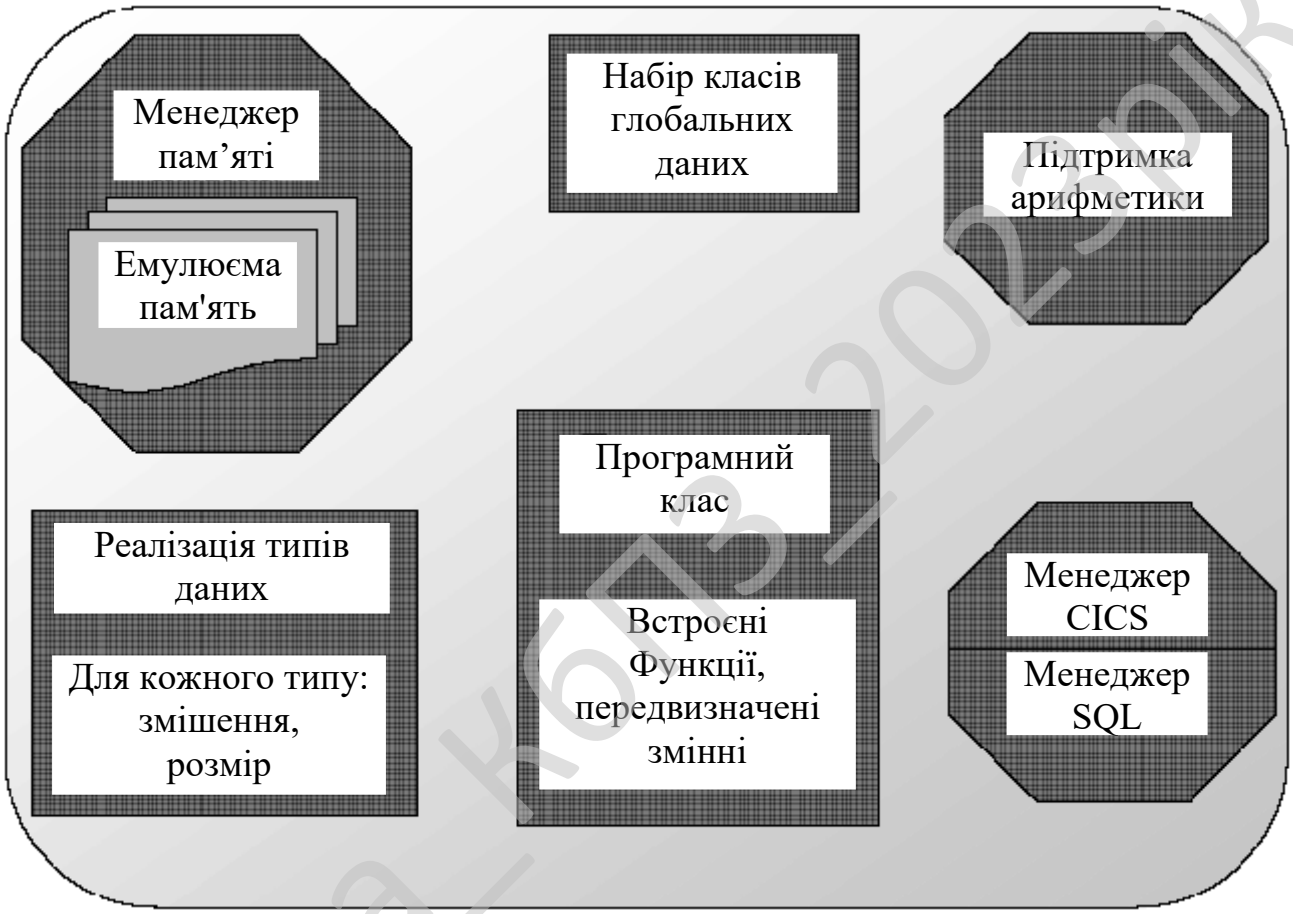


Рисунок 3.4 – Структура динамічної підтримки програмної логіки

Оскільки в мові Java ідеологічно відсутня можливість перевизначення пам'яті, було вирішено її емулювати за допомогою масиву з байтів, схованого в *менеджері пам'яті.* Тоді значення об'єктів даних представляються у вигляді зсувів і розмірів у цьому масиві. Таким чином, підтримка перевизначень (завдання №4) виходить автоматично, тому що в цьому випадку два різних об'єкти будуть мати однаковий зсув у пам'яті. Також значення будь-якої структури є сукупністю значень її полів за рахунок того, що вони мають зсув

Кафедра_КПЗ_2023рік

відносно її початку (якби значення змінних зберігалися усередині об'єктів даних, то досягти цього було б не так просто).

Реалізація убудованих функцій, частина яких обумовлена вхідною мовою, а інша процесом конвертації, і опис визначених змінних перебувають усередині *програмного класу.* Він використовується, як предок для переведеної програми, забезпечуючи рішення завдання №6. Також він містить набір сервісних методів і даних, необхідних для роботи самої підтримки часу виконання.

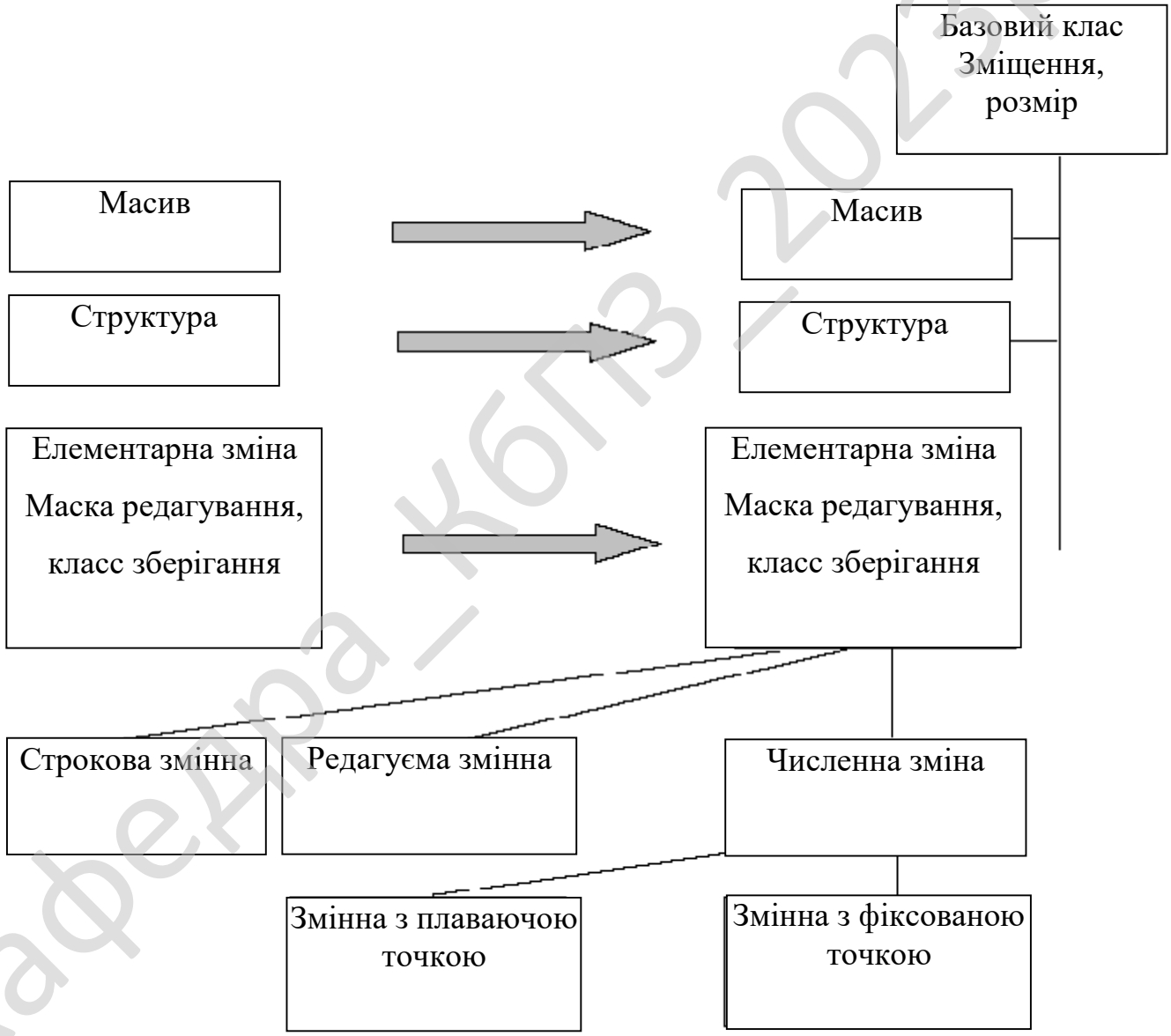


Рисунок 3.5 – Подання типів даних

КафеДра_КбПЗ_2023рік

Опишемо організацію підтримки математичних операцій (завдання №1). Базова підтримка арифметики з фіксованою точкою виконана у вигляді *окремого класу* (Decimal). При необхідності здійснення математичної операції для будь-який чисельної змінної здійснюється створення необхідної кількості його екземплярів, ініціалізованих значенням аргументів, з наступним викликом відповідного методу об'єкта. Результат виконання операції (проміжне значення) теж представлене екземпляром класу Decimal, у якому він зберігається або до наступної операції, або до присвоєння змінної.

Розглянемо реалізацію підтримки типів даних і операцій (рішення завдання №2). На рисунку 3.5 представлена побудована ієрархія типів даних і відповідність із вихідними типами для вхідної мови. Відзначимо, що в програмі можуть бути присутнім тільки листові типи з даного дерева спадкування. Кожний побудований тип є класом, що забезпечує необхідну функціональність. Таким чином, зроблений поділ сутності елементарної змінної по типу маски редагування й класу зберігання на кілька класів. Для всіх класів є методи строкового порівняння, присвоювання, одержання подання значення в пам'яті за значенням. Для числових змінних додані методи різних математичних операцій і порівнянь. Для забезпечення швидкості роботи відбувається зберігання часто використовуваної інформації усередині класів елементарної змінної в спеціальному полі. Воно заповнюється при ініціалізації даних.

Головна проблема передачі параметрів при міжпрограмнмх викликах (завдання №7) – передача по посиланню. Параметри, сприймані програмою, розташовуються в змінних спеціальної секції компонування. При цьому тип використання з'ясовується на момент виклику. Якщо в секції компонування присутня структура, і вона використовується у виклику по посиланню, то необхідно зробити модифікацію зсувів її полів. Це вимагає використання механізму рефлексії або додаткової роботи під час синтезу вихідної програми. Із цієї причини використовується спосіб копіювання й відновлення [14]. Відзначимо у зв'язку із цим додаткову складність при роботі з мовою Java: необхідність

Кафедра_КОПЗ_2023рік

використання класів-оболонок або масивів при бажанні передавати змінні елементарних типів по посиланню разом зі способом копіювання-відновлення.

Рішення завдання програмно обумовлених кодувань і зміна внутрішнього подання залежно від поточного кодування (№3) зроблено в такий спосіб. Передбачається, що весь додаток має однакову таблицю кодів. Використовується дворівневий підхід. Частина методів підтримки даних (форматування значення по масці редагування й ін.) працюють на робочому поданні, що використовує тільки одне кодування (ASCII). Інша частина робить перетворення робочого подання в емулюєме й назад, використовуючи таблиці прямої й зворотної кодувань. Настроювання таблиць виробляється під час виконання методу настроювання середовища виконання (додаються на момент синтезу в цільову мову) до створення даних. Його параметрами є тип кодування, розмір пам'яті, що відводиться для дані програми й ін.

Система CICS має більші можливості, тому було вирішено реалізувати лише найпоширеніші її частини. На даний момент підтримано операторів міжпрограмного переходу, взаємодії з користувачем, робота із чергою, деякі сервісні функції й глобальні змінні середовища. Вся підтримка CICS (№8) винесена в окремий модуль.

Опишемо найцікавіші питання, вирішені при її побудові. Опишемо оператори міжпрограмного взаємодії. Це може бути перехід на іншу програму зі збереженням і без поточної на стеці, переривання програми, повернення із програми. Реалізація зроблена в такий спосіб: визначені два типи виняткових ситуацій, використовуваних динамічною підтримкою. Один з них позначає повернення або перехід на іншу програму із заміщенням поточного образа, інший – закінчення програми. Кожний перехід зі збереженням образа на стеці відбувається в охоронному блоці, у якому ловляться всі виняткові ситуації у виконуваній програмі й фільтруються службові. Для них проводяться наступні дії: для того, що закінчує – викид (відбувається передача попередньому охоронному блоку), для іншого виду – перехід із заміщенням або повернення.

Кафедра_КОПЗ_2023рік

Помітимо, що для коректної роботи першого модуля й співіснування операторів виклику й закінчення програми мови Кобол з даною реалізацією CICS, при старті першої або виклику іншої програми необхідно також зробити перехід зі збереженням поточного кадру обробки виключень на стеці.

Динамічна підтримка для серверних додатків

Для використання переведених програм як серверні додатки здійснене виконання деяких додаткових вимог для динамічної підтримки. Деталі їхньої реалізації розглянуті далі.

В CICS менеджері додана можливість запуску програм окремим потоком керування. Це зроблено класом динамічної підтримки, що робить виконання даного йому програмного класу окремим потоком за допомогою убудованих засобів мови Java. Головне переваго написаного класу – можливість його повторного використання, що досягається наявністю семафора очікування. Після закінчення роботи програмного класу, об'єкт «засипає» на ньому доти, поки йому не дадуть новий клас для виконання. Оскільки такі об'єкти являють більшу цінність, менеджер CICS містить пул уже розподілених. При необхідності запуску він шукає «сплячий» об'єкт у пулі, і якщо знаходить, то використовує його, інакше створює новий із занесенням у пул.

Для забезпечення швидкої роботи динамічної підтримки реалізований пул тимчасових об'єктів. Опишемо механізм його роботи (ще один приклад наданий у роботі [15], схожі ідеї використовуються в системах керування пам'яті для дрібних об'єктів змінного розміру). На певному етапі розвитку даної системи автоматизованого перекладу було зроблене тестування часу виконання конкретного додатка. Результати показували сильне уповільнення роботи переведеного додатка, і застосування компіляторів часу виконання з байт-коду в машинний код не допомогло сильно зменшити розрив у швидкості. Після детального дослідження проблеми виявилося, що через неявне звільнення розподіленої пам'яті й створення безлічі тимчасових об'єктів, часті "підчищення" підсистеми складання сміття віртуальної машини Java роблять припинення всіх

Кафедра_КБПЗ_2023рік

потоків керування. Такими об'єктами в тестуємій архітектурі були об'єкти математичної підтримки й два різних класи, що використовувалися для зберігання імітуємого й робочого подань значень. Крім того, було відзначено, що велике створення екземплярів незмінних об'єктів (наприклад, класу String) приводить до того ж ефекту. Не чекаючи реалізації більш ефективних алгоритмів для звільнення пам'яті у віртуальній машині Java, скорочено до мінімуму використання незмінних об'єктів, і створена реалізація пулу тимчасових об'єктів. Для можливості повторного використання останні модифіковані додаванням ініціалізуючих методів.

Головне припущення в побудованій моделі: тимчасові об'єкти не використовуються всі одночасно. В об'єкті глобальних даних для кожного типу тимчасових об'єктів створюється певне число екземплярів (звичайно 8 або 16) і лічильник, що визначає номер наступного використовуваного об'єкта. Після використання він інкременується на циклічній основі (наприклад, що випливає після восьмого буде перший).

Для об'єктів, що зберігають значення, є певна верхня границя, при запиті об'єкта меншого розміру використовуються тимчасовий, а якщо більшого, то виробляється виділення нового об'єкта. Подібна границя існує й у систем виділення фізичної пам'яті. Для зменшення можливості використання нового об'єкта, виконання операції присвоєння між нечисловими змінними, які можуть мати розмір, що перевищує границю, виробляється копіюванням ділянок пам'яті. Використання даних методів дозволило прискорити роботу додатка в більш ніж у десять разів.

Реентерабельність є базовою вимогою для серверних додатків. Опишемо її реалізацію для трансформованої програми. Оскільки програма є класом, то поки ми використовуємо один екземпляр програмного класу для кожного потоку керування, проблем при виконанні в його межах не відбувається. При створенні нового потоку керування йому привласнюється номер, так званий *контекст виконання,* унікально визначаємий його й використовувані їм ресурси. Це число

Кафедра_КОПЗ_2023рік

являє собою номер сегмента використовуваної пам'яті. Воно зберігається в кожному програмному класі й у кожній змінній, в останніх це оформлено у вигляді частини поля зсуву (8 старших біт з 4-х байтового знакові цілого). Об'єкт глобальних даних, включаючи пул тимчасових об'єктів, тепер є елементом масиву глобальних даних і залишається глобальним у межах одного потоку керування. Таким чином, у будь-якому місці, де повинне виробляється звертання до нього, необхідно мати в наявності значення контексту виконання. При виклику однієї програми з іншої, викликувана успадковує контекст виконання програми, що викликає.

Зупинимося на питанні забезпечення унікальними номерами контексту виконання. Зсуви в змінних у текстах програми присутні у відносній (без номера сегмента) формі, тому при створенні нового програмного класу захоплюється глобальний монітор, що приводить будь-який інший потік керування (thread), що намагається змінити ці глобальні дані до закінчення роботи першого, у стан очікування. Далі програма ініціалізується в монопольному режимі, при цьому, якщо створюється новий потік керування, то відбувається одержання номера контексту керування із забезпеченням створення нового пула тимчасових об'єктів. Після цього глобальне статичне поле в програмного класу виставляється в значення зсуву, що містить номер сегмента, а при виконанні конструкторів змінних відбувається корекція зсуву з використанням даного поля. Після закінчення створення екземпляра класу програми монітор звільняється. Така ж послідовність початкових і кінцевих дій відбувається при видаленні контексту виконання, але використані ресурси лише позначаються як вільні, що дозволяє при створенні наступного контексту їх повторно використовувати.

Динамічна підтримка нових технологій

Розглянемо, що було зроблено для створення компонентів програмної логіки з використанням нових технологій: java applet (додаток) і java beans. Перша являє собою об'єкт, призначений для використання усередині браузера Internet на віддаленій машині. Він має істотні обмеження виконуваних дій (тому

Кафедра_КОПЗ_2023рік

середовище його виконання іноді називають «пісочницею») для запобігання небезпечних дій. У нашім варіанті, додаток створюється як контейнер для програмного класу. Він містить код для створення середовища виконання, наприклад, зміни вхідних і вихідних потоків даних (тому що воно не має прав використовувати їх як звичайний додаток) на віконний інтерфейс із користувачем, створення пускової кнопки й ін. Оскільки дані дії є легко параметризуємими ім'ям виконуваної програми, то вони винесені в малий програмний клас динамічної підтримки. У результаті створений додаток має малий розмір.

Java beans це компоненти програмної логіки, призначені до використання за допомогою візуальних засобів для швидкої розробки додатків. На рівні реалізації вони являють собою класи, оформлені певним чином. Коротко опишемо їхню архітектуру. Для кожного компонента визначається набір їх властивостей (properties), вироблені події, експортовані методи й інші властивості. Якщо цей набір атрибутів відсутній у вигляді класу опису, дана інформація будується на основі автоматичного рефлексивного аналізу, використовуючи сигнатури елементів класу. Оскільки останнє не є цілком придатним у цьому випадку, наприклад, може бути викликаний внутрішній метод, створюється клас опису під час синтезу цільової мови. Він має батьком клас підтримки, що містить код заповнення вищезгаданих атрибутів. Важливою особливістю компонентів є здатність до збереження або відновлення своїх властивостей, використовуючи файлові структури. Оскільки така можливість убудована в мову Java для звичайних об'єктів, то необхідно подбати тільки про зсуви в зміних (записувати їх у відносному виді), емулюємої пам'яті й ресурсах не підлягаючих збереженню. При цьому, для подолання можливих конфліктів між різними потоками керування при операціях збереження або відновлення виробляється захват глобального монітора, також як і при створенні нового програмного класу. Специфічна для компонентів функціональність (наявність видимого методу виконання й ін.) оформлена у вигляді окремого класу, що успадковується від

Кафедра_КОПЗ_2023рік

програмного класу підтримки. При цьому трансформована програма для її реалізації відповідно міняє ім'я батьківського класу.

У двох останніх випадках використання підтримки часу виконання не є обов'язковим, оскільки вона використана, в основному, для наочності або зменшення розміру текстів. При необхідності створення більше простого програмного коду, її тексти можна злегка змінити для автоматичного включення під час переглядів синтезу або генерації у вихідну мову.

Динамічна підтримка для комплексного перекладу

Із самого початку для спрощення процесу перекладу додатка, була встановлена посилка про незалежність перекладу інтерфейсної частини від програмної логіки. Але у випадку наявності в програмній логіці роботи з користувачем, у такій постановці потрібні ручні подальші виправлення отриманих компонентів для відновлення даної можливості. Тому що це представляється не завжди зручним, у конверторі RescueXP була додана можливість одержання інтегрованих компонентів. Частина цієї роботи виконана в підтримці часу виконання.

Спілкування з користувачем у старих додатках виконується асинхронним образом (у той час, коли користувач уводить які-небудь дані, програма продовжує виконуватися). Із цієї причини необхідно синхронізувати виконання інтерфейсних компонент із програмною логікою. Це можливо здійснити сильною трансформацією останньої (спосіб описаний у роботі [6]), але даний підхід на сучасний момент не реалізований. Інший можливий спосіб – рознести програмну логіку й інтерфейс у два різних потоки керування, а синхронізацію виконувати використанням відповідних примітивів. Варіант останнього був реалізований в [9], опис методу приводиться нижче. Відзначимо, що частина рішення даного завдання, наприклад, додавання операторів обміну інформації, здійснюється на етапі синтезу в цільову мову й генерації інтерфейсних компонент.

Архітектура даного методу синхронізації припускає, що програмна логіка й інтерфейсні компоненти різнорідні по середовищу виконання, (наприклад,

Кафедра_КбПЗ_2023рік

здійсненний модуль для першої й HTML сторінки + код мовою сервера для другої). Це виражається в наявності об'єкта посередника, що і приховує деталі сполучення, а його існування можливо й у першому середовищі, і в другому. Даний метод синхронізації припускає, що час виконання частини коду від запису до читання даних менше, ніж час виконання интерфейсної ділянки від читання до запису. Оскільки на даному етапі відбувається взаємодія з користувачем і передача даних в оба напрямки (якщо він перебуває на іншій машині, то воно може бути значним), то дане обмеження представляється не сильно істотним.

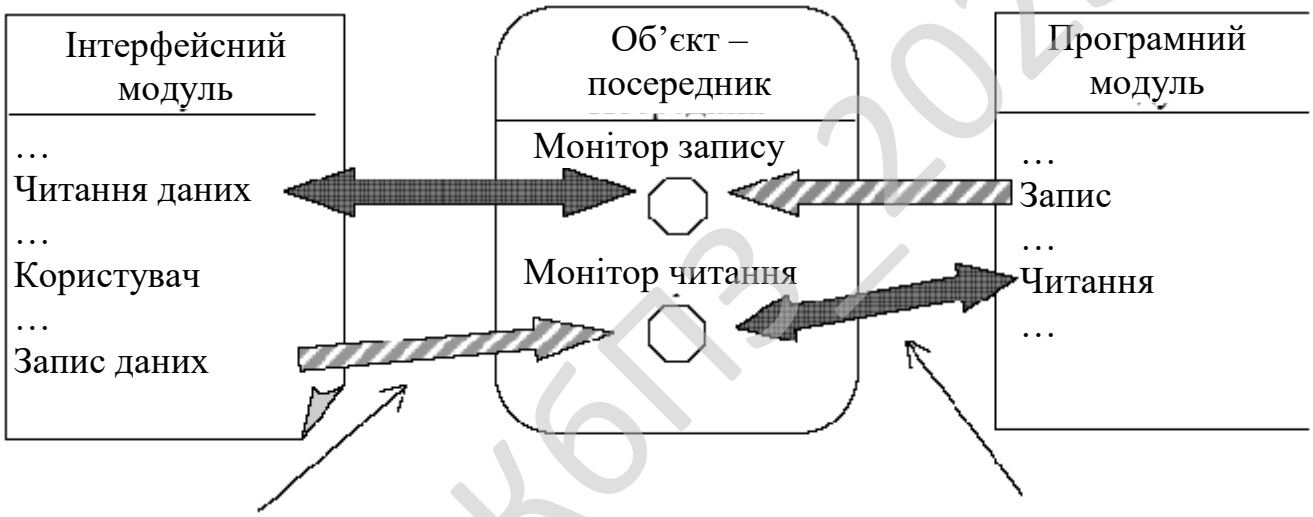


Рисунок 3.6 – Схема взаємодії частин системи.

Опишемо алгоритм синхронізації. Ініціатива йде від програмної логіки, коли вона хоче передати дані в інтерфейсну компоненту, остання повинна вже чекати цієї події, використовуючи монітор запису. Передавши інформацію, програма відкриває цей монітор і продовжує працювати далі до моменту читання, де вона засипає на моніторі читання. Тим часом, звільнений від оковів потік керування інтерфейсного модуля виводить отриману інформацію й чекає реакції користувача, після чого інформація, дана останнім, передається сплячій програмі. Вона будиться, а логіка інтерфейсного модуля, зажадавши даних, засипає на

Кафедра_КБПЗ_2023рік

семафорі запису, після цього цикл повторюється необхідне число раз. Помітимо, що інформація, передана в останню, може містити директиву переходу на інший інтерфейсний модуль. Передача даних і робота із семафорами схована в об'єкті посереднику, при цьому очікування сигналу може припинитися через протікання певного інтервалу часу. В останньому випадку відбувається завершення роботи тої частини, яку вчасно не розбудили. Це може відбутися з кількох причин: помилка програми, довга або неправильна реакція користувача (остання викликає збій ролей у даній схемі), і ін.

До числа функцій підтримки часу виконання при використанні даної схеми варто також віднести обробку переданих даних (побічно виробляються присвоювання значень визначеним CICS змінним) і здійснення неявного читання. Дана дія виробляється, у випадку, якщо поточна програма зробила запис інформації для користувача, а потім виконала оператор переходу на іншу програму без збереження поточної.

3.2 Розробка структурної схеми

Всім відомо що зараз є велика кількість вихідного коду на старих мовах програмування, а саме таких як Cobol, 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.

Кафедра_КбПЗ_2023рік

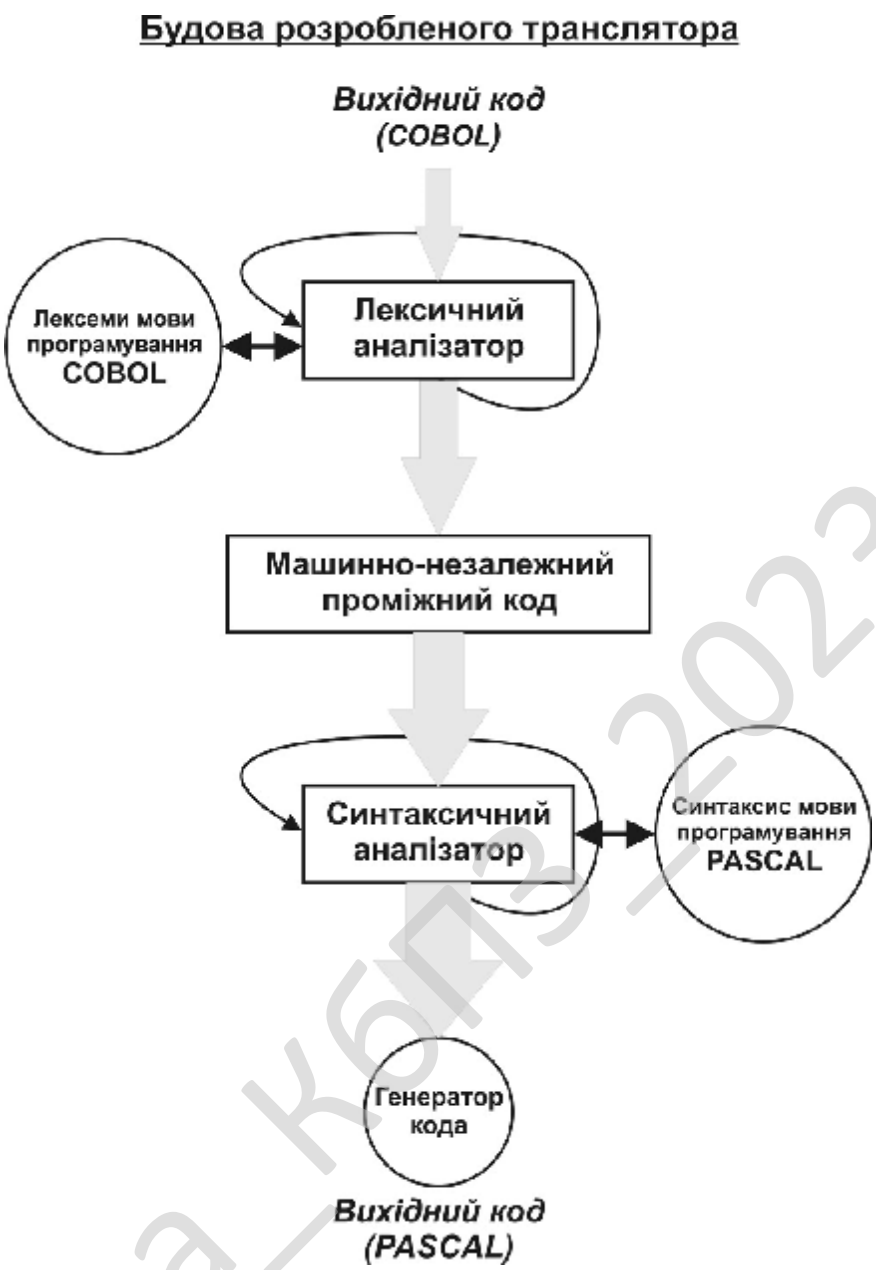


Рисунок 3.7 – Структурна схема системи

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 даже таке що код після перетворення не буде у повній мірі робити але сама структура алгоритму залишиться.

У розробленій дипломній програмі використається переклад частин вихідного коду з мови програмування Cobol у Pascal.

Кафедра_КБПЗ_2023рік

На рисунку 3.7 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 COBOL, проміжного машино-незалежного коду, синтаксичного аналізатора з синтаксисом мови Pascal та генератора коду.

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

3.3 Розробка функціональної схеми

Розглянемо детально роботу транслятора котра зображується на рисунку 3.8. Після початку обробки коду Cobol проходить перевірка кода на відповідність (докладніше розглянуто у дипломних 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

Кафедра_КОПЗ_2023рік

пропозицій мови відповідно до граматики мови. Тільки за відсутності помилок можлива робота наступних модулів компілятора.

Рисунок 3.8 – Функціональна схема системи

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.

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 –

Кафедра_КбПЗ_2023рік

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.

Розробка функціональної схеми програмної частини схеми

Розглянемо структуру розробленої програми та її дії з вихідним кодом Cobol безпосередньо. На рисунку 3.9 зображено функціональну схему програмної частини.

Інтерфейс windows взаємодіє з інтерфейсом розробленого дипломного програмного забезпечення котрий в свою чергу розгалуджується на налагодження параметрів ПЗ, авторське право, лексичний аналізатор котрий взаємодіє з синтаксичним аналізатором та генератор кода.

Розроблена програма містить в собі дві основні таблиці – таблиця термінальних символів Cobol та синтаксис Pascal.

Ці таблиці взаємодіють між. друг другом, ця взаємодіє розглядається на рисунку 3.10. Кожному термінальному символу чи послідовності символів на Cobol є своя заміна у Pascal. В залежності від наповнення цих таблиць буде робити реінженірінг (перетворення коду).

Кафедра_КПЗ_2023рік

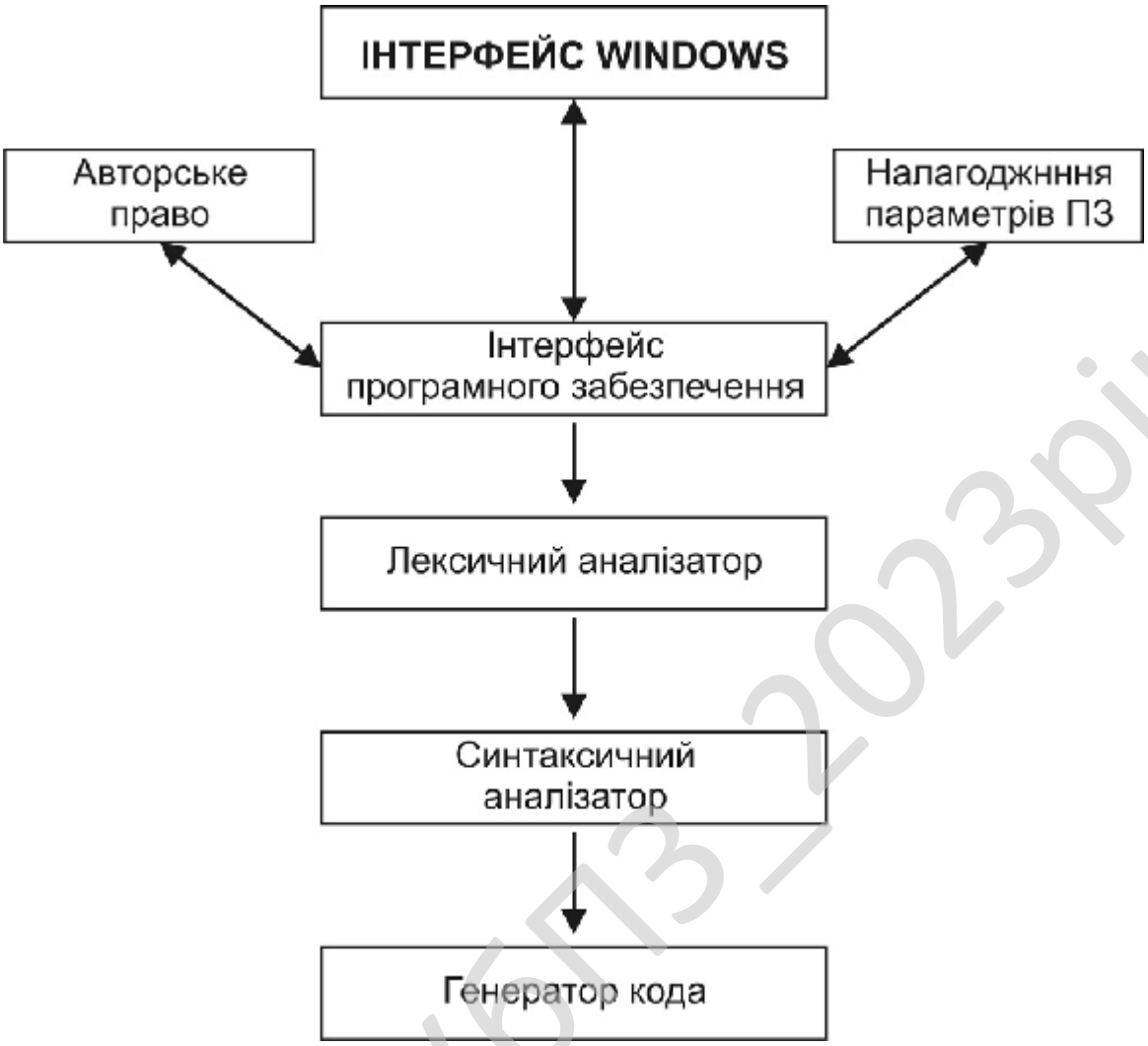


Рисунок 3.9 – Функціональна схема програмної частини

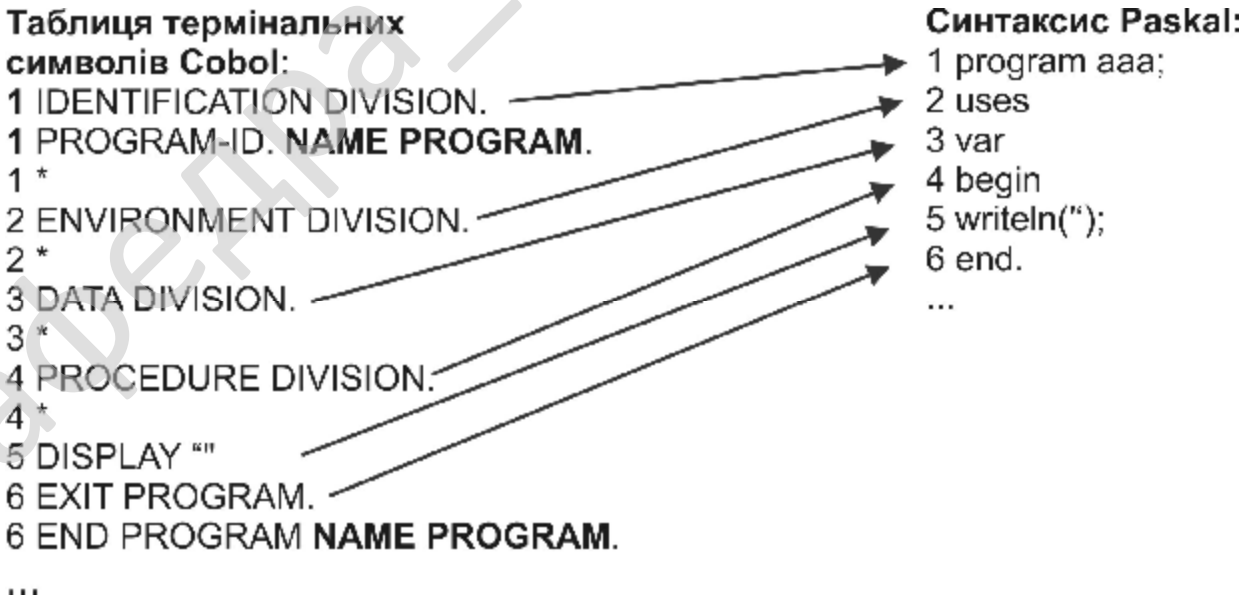


Рисунок 3.10 – Таблиці термінальних символів Cobol та синтаксис Pascal

Кафедра_КБПЗ_2023рік

3.4 Розробка діаграми процесів

Важливим критерієм при розробці програмного забезпечення це є грамотна розробка структури роботи системи. На діаграмі процесів зображеній на рисунку 3.11 можна точно зрозуміти як працює і взаємодіє розроблене програмне забезпечення. в цілому.

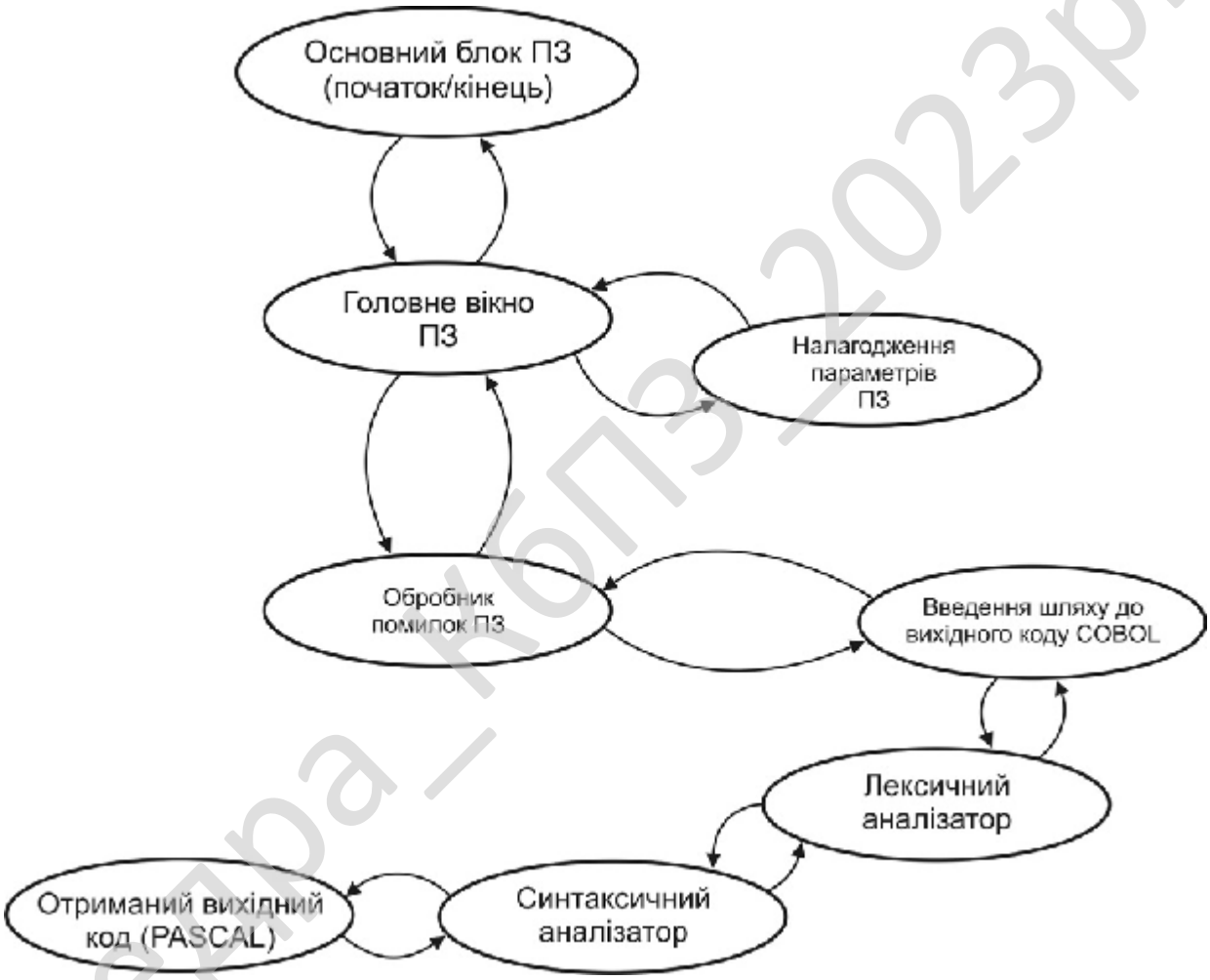


Рисунок 3.11 – Діаграма процесів

Кафедра_КбПЗ_2023рік

Починається і закінчується програма в першому блоці це є основною крапкою відрахунку діаграми. При переміщенні по стрілках можна побачити загальну схему взаємодії блоків і їх входження один в одного, а саме як з основного блоку програми управляння переходить спочатку до головного вікна програми та потім до вікна налагодження параметрів ПЗ та обробника помилок .

Далі через послідовність введення шляху до вихідного коду COBOL, лексичного аналізатора, синтаксичного аналізатора отримується отриманий вихідний код (PASCAL).

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.

Кафедра_КбПЗ_2023рік

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 розглянут основний алгоритм роботи дипломної програми. З основного алгоритму програми винесені дві основних підпрограми це підпрограма лексичного аналізатора (рисунок 4.2) та підпрограма синтаксисичного аналізатора та генерації коду (рисунок 4.3).

Першою основною задачею алгоритмові 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 мові PASCAL 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.

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.

Кафедра_КОПЗ_2023рік

```
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.
24 .. інші дані.
```

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 ідентифікатораі номер рядка стані специфікатором у коді лексеми. Якщо є ,те 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.
```

Кафедра_КбПЗ_2023рік

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.

38 ... інші оператори.

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.

Виділимо у розробленій блок-схемі основні блоки (рисунок 4.1).

Початок роботи та перевірки:

Початкова ініціалізація змінних, та підключення бібліотек;

Підключення бібліотеки лексичного аналізатора;

Підключення бібліотеки синтаксичного аналізатора;

Підключення лексем мови програмування COBOL;

Підключення синтаксису мови програмування PASCAL;

Робота програми:

Виведення головного вікна ПЗ на екран;

Очікування дій користувача;

Виведення вікна вибору вихідного коду та зберігання генерованого коду;

Вибір вихідного коду COBOL;

Приховання вікна вибору вихідного коду та зберігання генерованого коду;

Робота підпрограми – лексичного аналізатора

Виведення повідомлення початку лексичного аналізу;

Підключення таблиці термінальних символів та таблиці лібералів;

Кафедра_КПЗ_2023рік

Читання файлу з кодом COBOL;

Перевірка на наявність коду COBOL;

Читання лексеми;

Розпізнавання лексеми мови;

Пошук та підстановка коду з таблиці термінальних символів;

Створення послідовності із всіх лексем коду;

Утворення машинно-незалежного проміжного коду;

Виведення повідомлення кінця лексичного аналізу;

Робота підпрограми – підпрограма синтаксисичного аналізатора та генерації коду

Виведення повідомлення початку синтаксисичного аналізу;

Підключення синтаксису мови програмування PASCAL;

Виділення фрагменту машинно-незалежного проміжного коду;

Аналіз фрагменту машинно-незалежного проміжного коду;

Обробка помилки, формування тексту синтактичного повідомлення;

Виведення тексту синтактичного повідомлення;

Створення послідовності із всіх фрагментів об'єктної моделі;

Виведення повідомлення кінця синтаксисичного аналізу;

Приховання головного вікна програми;

Виведення вікна генерації вихідного коду;

Запуск генератора коду;

Створення структури файлів вихідного коду;

Заповнення файлів вихідного коду;

Закінчення генерації та отримання коду Pascal;

Приховання вікна генерації вихідного коду;

Виведення головного вікна програми;

Завершення роботи:

Приховання головного вікна ПЗ на екран;

Відключення додаткових модулів;

Звільнення ресурсів програми.

Кафедра_КОПЗ_2023рік

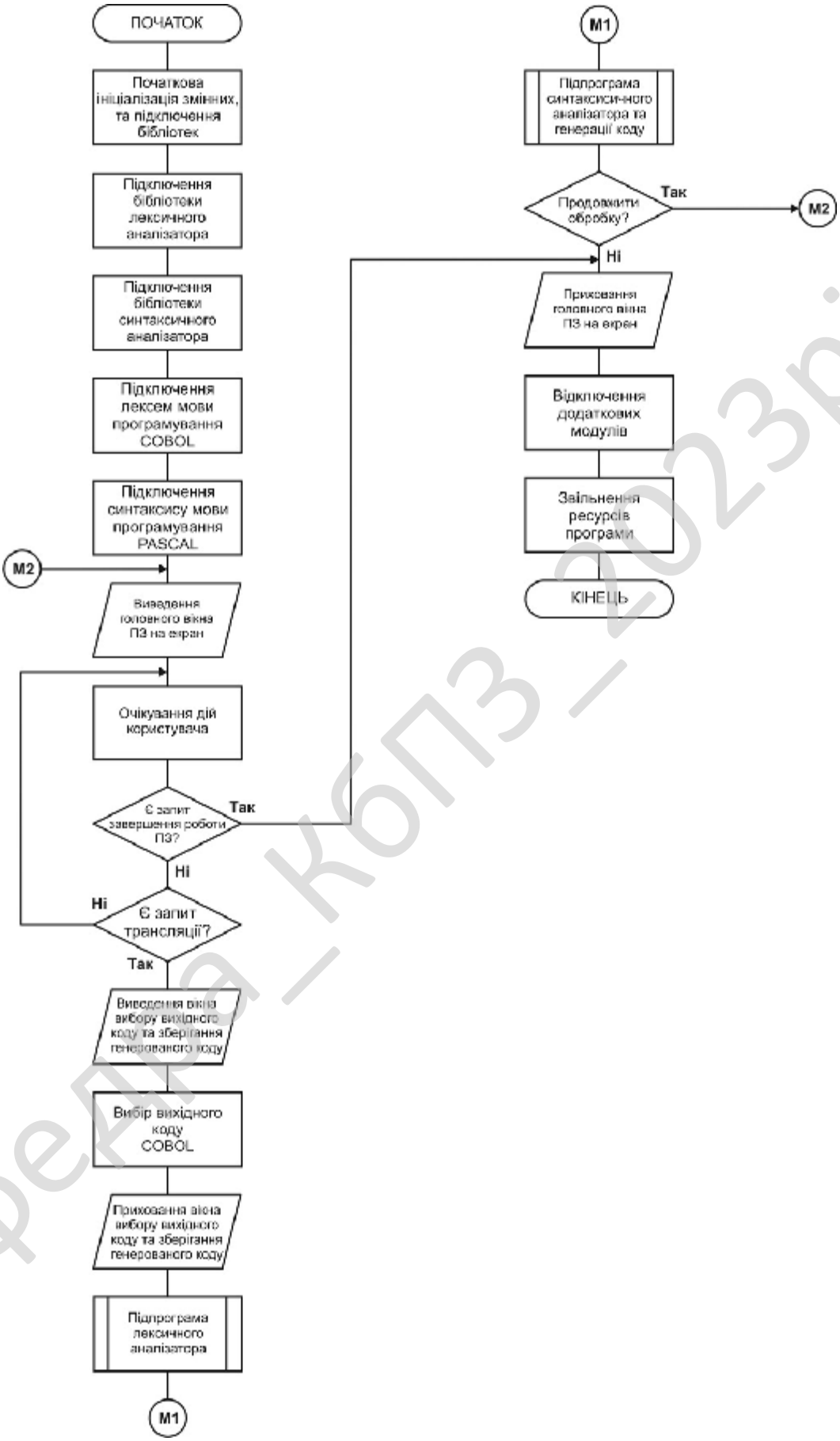


Рисунок 4.1 – Блок схема основної частини ПЗ

Кафедра_КбПЗ_2023рік

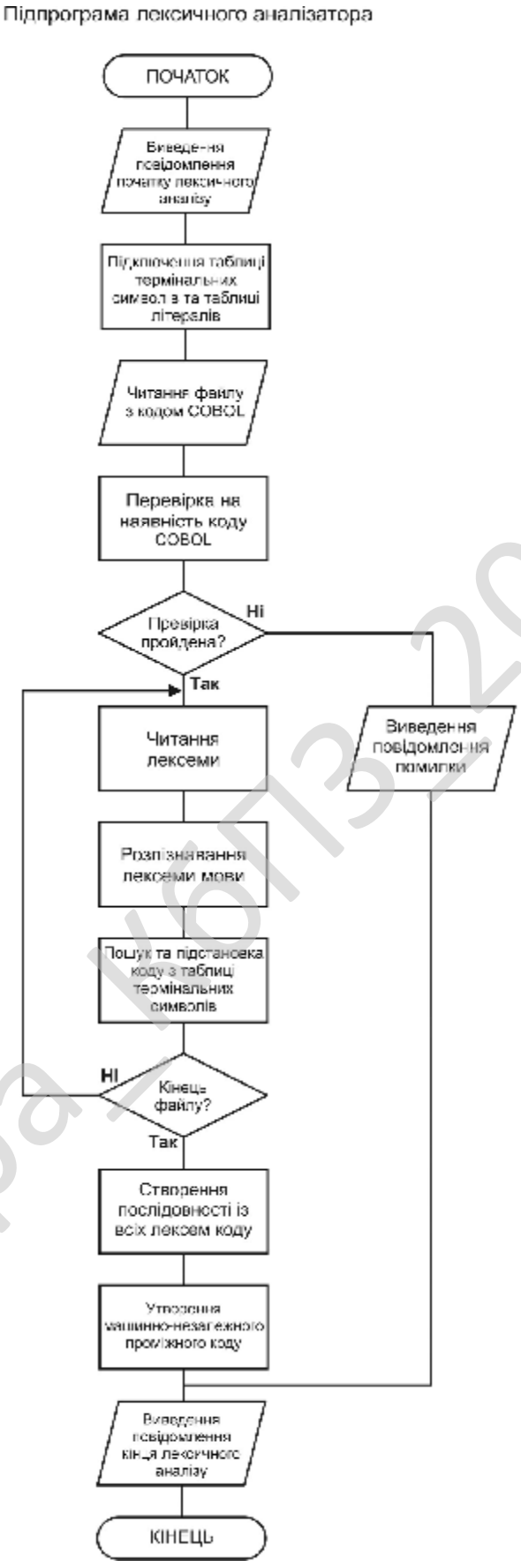


Рисунок 4.2 – Блок-схема підпрограми лексичного аналізатора

Кафедра КБПЗ 2023рік

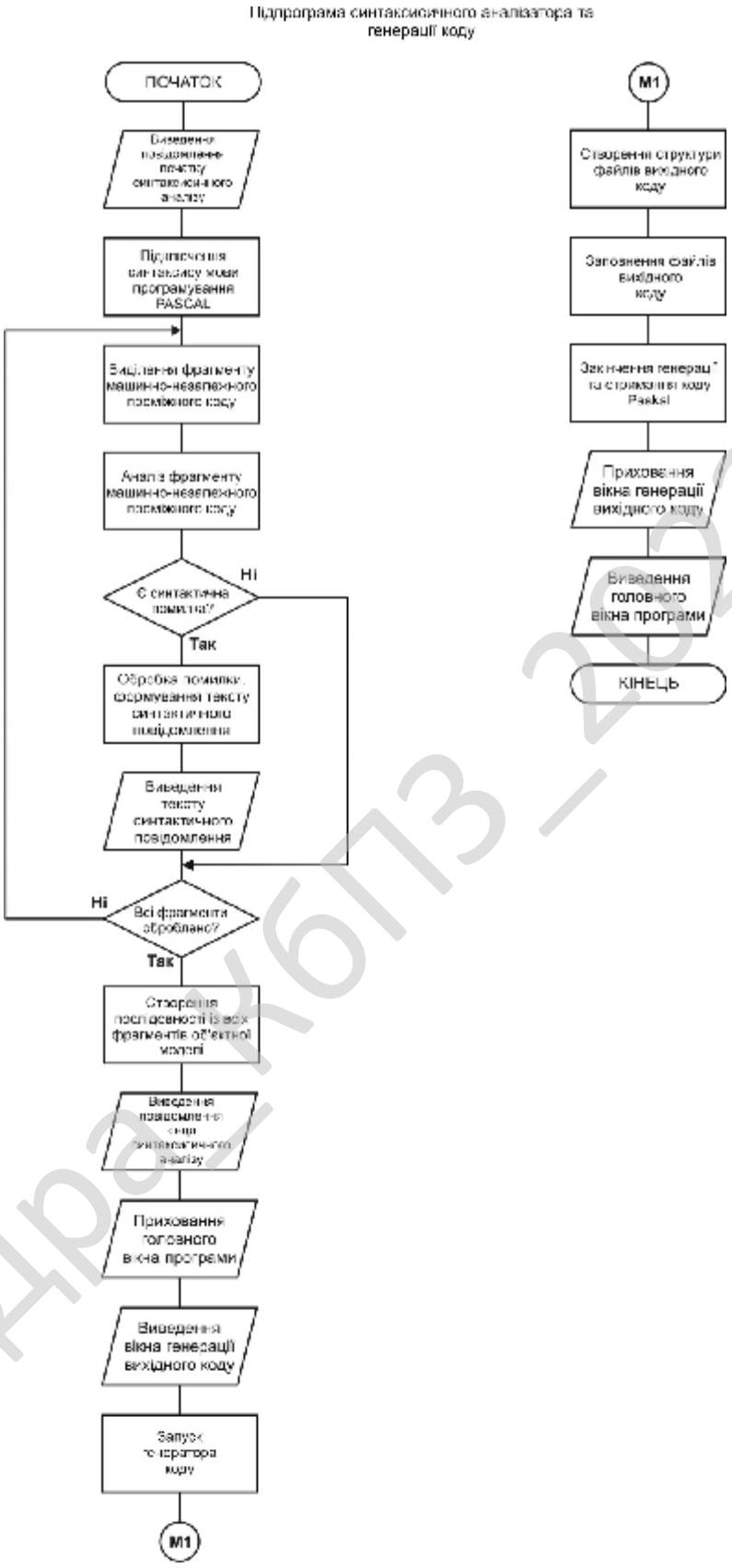


Рисунок 4.3 – Блок-схема підпрограми синтаксисичного аналізатора та генерації коду

Кафедра КБПЗ – 2023рік

Розрахункові структури даних транслятора що підтверджують вірність конструкторських рішень

Розглянемо структури даних котрі використовувалися у трансляторі опишемо їх в вигляді роз'яснення.

Рядок – послідовність символів , що належати кінцевому безлічі, або алфавітові. Наприклад , якщо А – безліч{0,1} , те послідовність 01011101101110 є рядок над А. Для відділення рядків друг від друга застосовуються або спец символи, або змінюється алфавіт.

Властивості:

1) довжина рядків перемінна;

2) звертання до елементів рядка відбувається з якого-небудь кінця(тобто важлива упорядкованість послідовності , а не її індексація);

3) якщо рядок представляє кінцеву пропозицію деякої мови, ті існують обмеження, які символи можуть зустрічатися разом , що задається синтаксисом мови.

Орієнтований граф – формально визначається як G=(X,U) ,де X – безліч елементів, називаних вершинами; U – безліч дуг або ребер графа: U X*X. Якщо a і b вершини , a (a,b) – ребро.

Неорієнтований граф – у якому відсутні орієнтовані ребра: (a,b)U , ті і (b,a) U. Якщо на додаток до графові задана функція w:U R , ті це зважений граф . Функція w визначає довжину або ваги шкірного ребра в графі.

Дерево – Кореневе дерево(орієнтоване дерево, дерево) – орієнтований граф , у якому- усі вершини, крім однієї (кореня), знаходяться в голові тільки одний дуги; корінь дерева не знаходиться в голові якої-небудь дуги; корінь зв'язаний з кожною вершиною дерева.

Обхід дерева – це упорядкована послідовність вершин дерева, у якому кожна вершина зустрічається тільки один раз. У кожну вершину попадаємо принаймні один раз, а взагалі говорячи 3 рази. При обході дерева можуть виконуватися різні дії,

Кафедра_КОПЗ_2023рік

Стеки і черги – динамічні структури, приспособлені для того, щоб додавати елементи в безлічі і видаляти їх. Стекова пам'ять, організована за принципом LIFO ("Last In First Out"). Стек – обмежен з однієї сторони послідовністю перемінної довжини.

Список-послідовність елементів, де кожен елемент містить інформацію двох типів: внутрішню-властиву даному елементові, і зовнішню-про зв'язок даного вузла з іншими елементами спискові. Зовнішня інформація про зв'язок елемента має вигляд покажчиків, тобто адреса вузлів, зв'язаних з даними.

Порівняння побудованої підтримки із запропонованою в системі PerCobol

Середовище розробки додатків PerCobol фірми Sychronix [11] являють собою інструментальний засіб, що дозволяє програмам мовою Кобол використовувати сучасні технології.

Це досягається за рахунок розширення синтаксису вихідної мови в припущенні, що вони будуть оформлені у вигляді додатків мовою Java. Так, наприклад, додана можливість роботи з мовою HTML, виклику зовнішніх програм, використання віконного інтерфейсу й ін. Хоча дана функціональність реалізується за рахунок перекладу програм у мову Java, автори продукту не рекомендують робити переорієнтацію таким способом, тому що даний процес близький до компіляції. Це виражається в наступних деталях.

1. Текст результуючої програми сильно насичений деталями реалізації. Наприклад, 10 строковий Hello, World стає 500 строковою програмою.
2. Орієнтованість на один потік керування.
3. Відсутність структурованості при перекладі даних.

Зробимо порівняння динамічної підтримки в даній системі з описаною вище підтримкою. Помітимо, що перша не є доступною у вихідних текстах, використовувана інформація – результат зворотного проектування.

Відзначимо загальні архітектурні рішення. У даній динамічній підтримці елементарні типи даних представлені об'єктами, а значення зберігаються в масиві

Кафедра_КОПЗ_2023рік

емулюємої пам'яті. Також є присутнім підтримка арифметики з фіксованою точністю, виконана на більш низькому рівні, чим дана реалізація.

Відзначимо розходження з побудованою системою. Типи даних представлені у вигляді іншої ієрархії, так структура й масив представлені одним класом, що є предком для числового типу, сини якого діляться по класу зберігання.

Кожний клас даних містить велику кількість полів, наприклад, кешоване значення, посилання для подання у вигляді ієрархічного бінарного дерева зі зворотним зв'язком по відношенню змісту. Дана організація дозволяє відрізняти об'єкт, що представляє структуру, від об'єкта елементарного даного й робити кешування значень в об'єктах даних. Зсуви в пам'яті представлені окремим класом, через який виробляється доступ до значення, причому використовувана пам'ять не обов'язково програмна. Дана можливість передбачена для створення тимчасових об'єктів. При побудові даної системи не приділялася увага мінімізації їхнього числа.

Підтримка часу виконання цього продукту містить супровідний код для файлових операцій (чого немає в побудованій системі), але позбавлена такого для мови CICS. Крім того, у ній реалізовані оператори, що рідко зустрічаються, мови Кобол (масиви змінної довжини й ін.). Серед нових можливостей, що є присутніми у даній системі, слід зазначити можливість створення компонент серверної логіки (виведений текст і запити на введення передаються клієнтові), які можуть працювати без наявності сервера, доступ до програмних змінних за допомогою графічного інтерфейсу й ін.

Кафедра_КбПЗ_2023рік

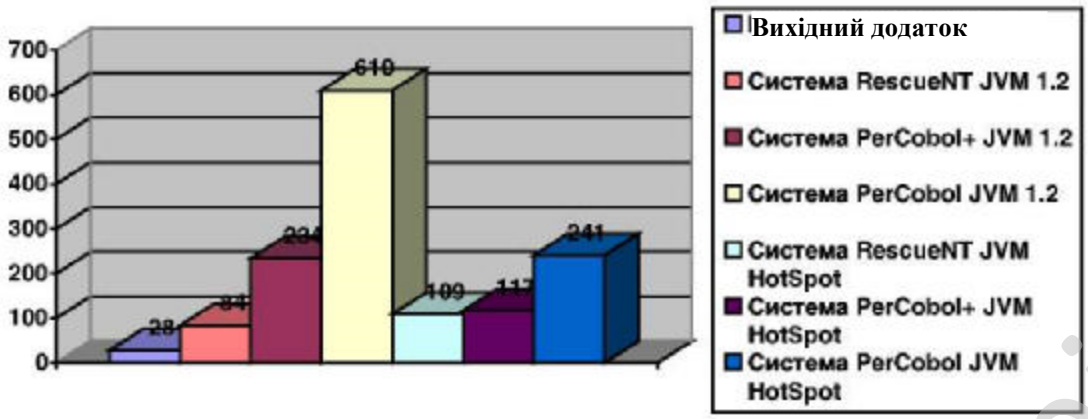


Рисунок 4.4 – Час виконання обчислювального тесту

Для порівняння продуктивності двох систем був зроблений вимір часу виконання обчислювального тесту при використанні різних віртуальних машин. На рисунку 4.xxxxxx показані його результати. Для порівняння наведений час виконання додатка до конвертації. Система PerCobol+ відрізняється від PerCobol включеним режимом кешування пам'яті в змінні. Прокоментуємо отримані результати для різних віртуальних машин. Особливістю JVM 1.2 є наявність високоефективного генератора машинного коду з байт-коду програми (це видно зі значення для системи RescueXP), але дуже неефективне «складання сміття». У системі JVM HotSpot використана техніка динамічної оптимізації 20% коду, виконуваного 80% часу, і інша реалізація неявного звільнення пам'яті. Саме остання особливість привела до сильного поліпшення показників системи PercCobol, тому що по якості виробленого машинного коду вона за попередньої реалізації (з показників системи RescueXP).

Підіб'ємо підсумок порівнянню. Недоліки системи підтримки часу виконання PerCobol:

1. Падіння продуктивності внаслідок неконтрольованого створення тимчасових об'єктів.

2. Громіздкість через змішування особливостей вихідної й цільової мов.

3. Відсутність підтримки CICS, виконання перекладу тільки програмної логіки.

Кафедра_КПЗ_2023рік

Перевага цієї системи:

1. Повнота охвату вхідної мови.
2. Більші можливості.
3. Вдале рішення перетворення вихідних типів в об'єкти, що містять схожі типи цільової мови.

При використанні системи PerCobol як поліпшений інструментальний засіб для певного виду старих додатків, перевага динамічної підтримки переважують недоліки.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може ще підняти швидкість шифрування в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 4.5).

Кафедра_КОПЗ_2023рік

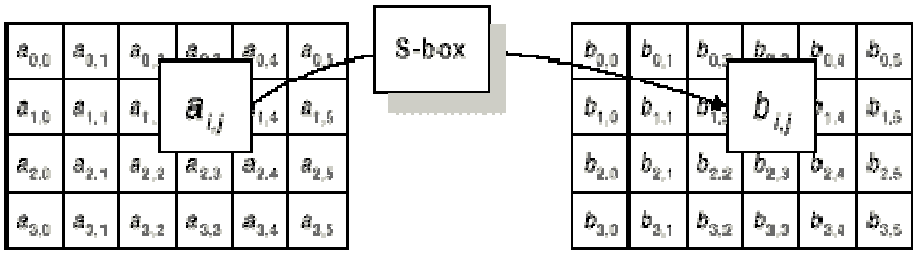


Рисунок 4.5 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 4.6).

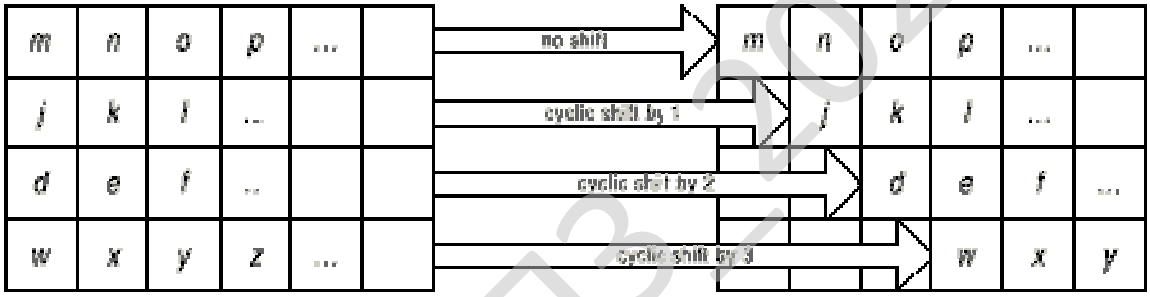


Рисунок 4.6 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 4.7).

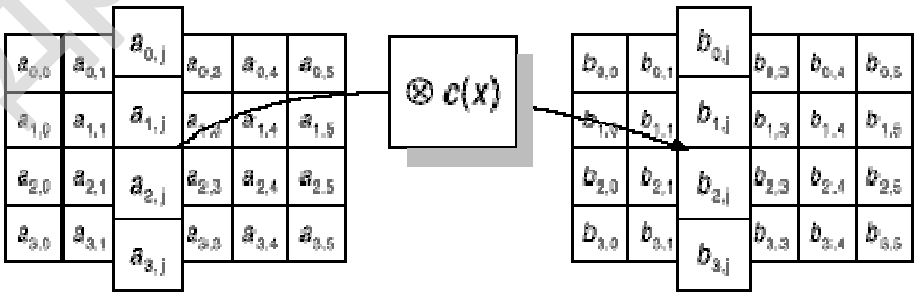


Рисунок 4.7 – Математичне перетворення, що перемішує дані усередині стовпця

Кафедра_КбПЗ_2023рік

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 4.8).

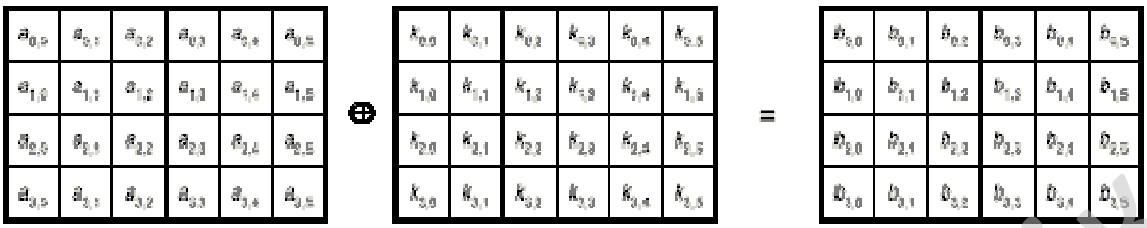


Рисунок 4.8 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

Кафедра_КбПЗ_2023рік

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Після написання програмного продукту його потрібно впровадити в промислову експлуатацію. При цьому виправляються помилки, які були помічені, система настроюється на відповідний режим роботи.

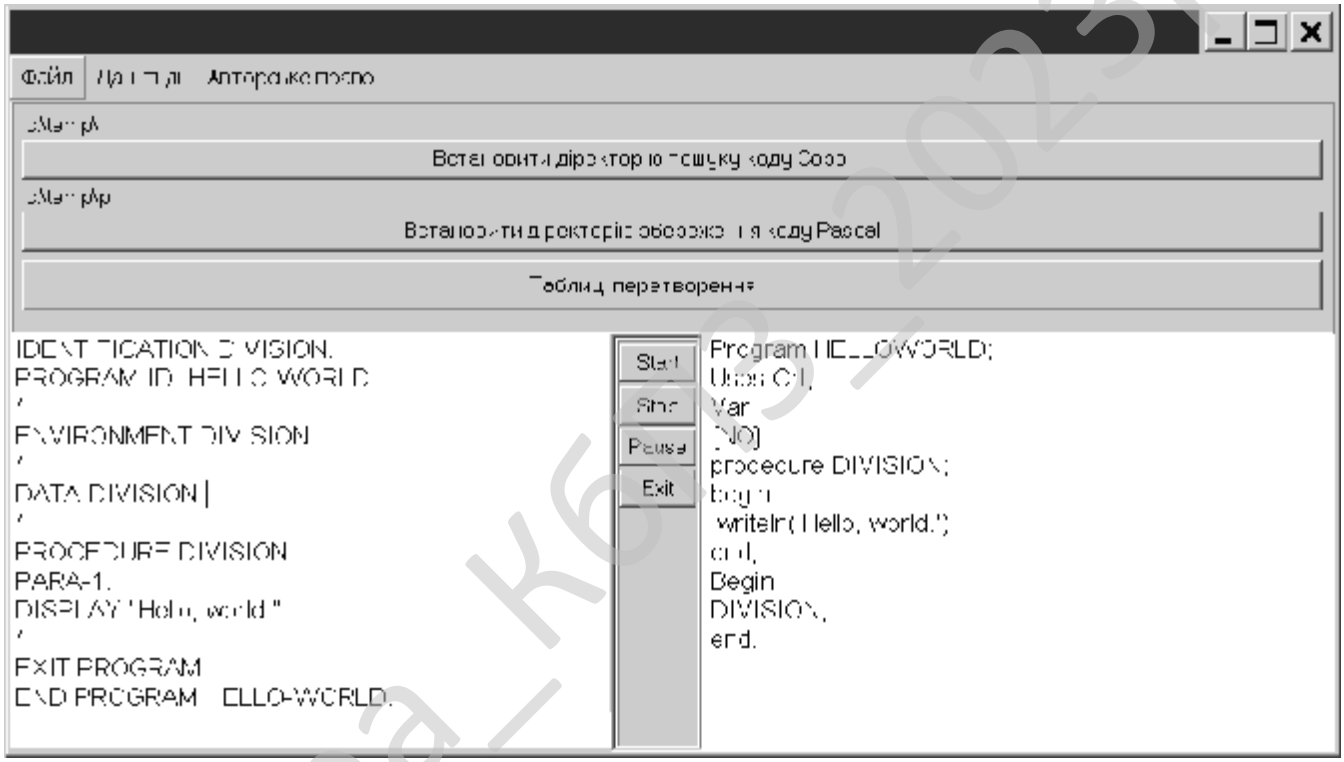


Рисунок 5.1 – Головне вікно програми

На рисунку 5.1 зображена головна форма програми. З неї ми бачимо, що процес реінжинірінга відбувається наступним чином:

У вікно яке розташовано по ліву сторону екрану заноситься код написаний на мові Кобол. У праве вікно відображається код, на мові Паскаль, який отриманий у результаті виконання програми.

На рисунку 5.2 зображено вікно у якому у режимі реального часу можлива правка, коду, який отримано у результаті виконання роботи програмного забезпечення розробленого у результаті виконання дипломного проекту.

Кафедра_КБПО_2023рік

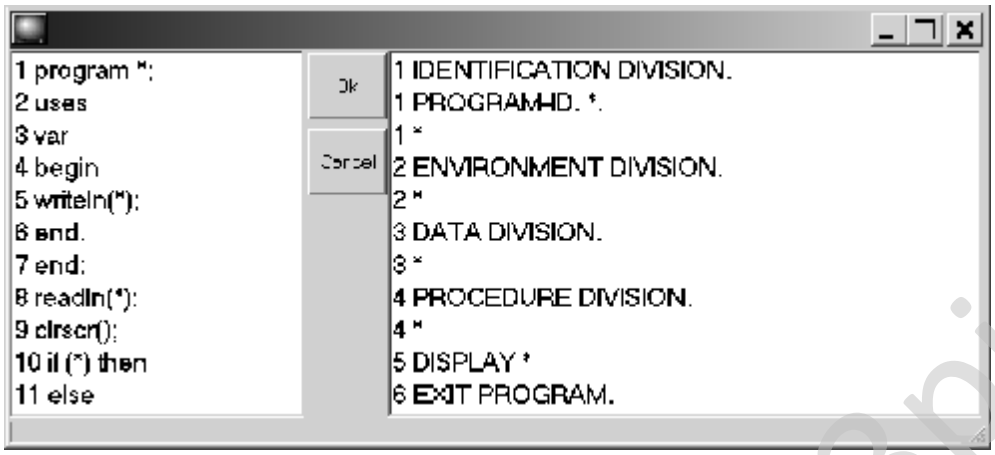


Рисунок 5.2 – Вікно редагування результатів реінжинірінгу

Рисунок 5.3 – Авторське право

Кафедра_КБПЗ_2023рік

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

– Досліджена система реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

– На основі отриманих результатів досліджень створена програмна реалізація системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання реінжирингу коду при модернізації інформаційних та комп'ютерних систем.

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

Кафедра_КбПЗ_2023рік

системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем. 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 AES.

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.

Кафедра_КБПЗ_2023рік

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Апрелев А. Единое внутреннее представление программы в процессе перевода ее с одного языка на другой. // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
2. Богданов В., Гордеев В. Практический опыт написания синтаксического анализатора языка программирования КОБОЛ // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
3. Богданов В. Синтезирующая часть системы RescueWare // Трансляция вызовов и переходов из COBOL в C++ // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
4. Волошин К., Плисс О. Трансляция вызовов и переходов из COBOL в C++ // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
5. Волошин К., Плисс О. Устранение локальных GOTO // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
6. Гордеев В. Структура управления диалогом // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
7. Джермейн К. Программирование на IBM/360 – М.: Мир, 1971. – 870 с.
8. Друнин А., Штукенберг Д. Построение срезов программ // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”
9. Казанский Б. Генерирующая часть системы автоматизированного реинжиниринга RescueWare // Дипломная работа

Кафедра_КПЗ_2023рік

10. Плисс О. Представление программы в виде косвенного дерева // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”

11. Терехов А.Н., Терехов А.А. Перенос приложений и проблема 2000 года //"Компьютер-Пресс", N8, 1998, стр.92-96

12. Трошин С.Л. Перевод встроенных SQL приложений в системе Rescueware // “Записки семинара кафедры системного программирования. Автоматизированное средство реинжиниринга RescueWare.”.

13. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**.

14. Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**.

15. Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018.218.230

16. Kovalenko Oleksandr, The mathematical model of the testing technology for DOM XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-

Кафедра_КОПЗ_2023рік

o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

17. Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

18. Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монографія / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 305 с.

19. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

20. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

21. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

22. Коваленко А.В. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

23. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко,

Кафедра_КБПЗ_2023рік

А.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

24. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

25. Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

26. Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

27. Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

28. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

29. Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

30. Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

31. Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації

Кафедра_КБПЗ_2023рік

та зв’язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

32. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

33. Коваленко О.В. Управління ризиками розроблення програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. – 2018. – С. 128-140.

34. Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

35. Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

36. Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141

37. Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

38. Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

39. Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

Кафедра_КБПЗ_2023рік

40. Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2(3). – Харків. – 2018. – С. 41-48.

41. Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

42. Коваленко О.В. Математична модель технології тестування вразливості до SQL ін'єкцій / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

43. Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

44. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

45. Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

46. Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.В. Коваленко, А.А. Смирнов // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. –

Кафедра_КПЗ_2023рік

С. 264-266.

47. Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

48. Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

49. Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

50. Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

51. Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез VIII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

52. Коваленко А.В. Исследование дерева рисков разработки

Кафедра_КБПЗ_2023рік

программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез ІІІ міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

53. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

54. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

55. Коваленко А.В. Метод управления рисками разработки программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.В. Коваленко, А.А. Смирнов // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

56. Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць ІІІ міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

57. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез ІІ

Кафедра_КБПЗ_2023рік

науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

58. Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

59. Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез дев’ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

60. Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

61. Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. Кіровоград: КНТУ. – 2017. – С. 92.

62. Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез IX

Кафедра_КОПЗ_2023рік

міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

63. Kovalenko O.V. Method of testing the dom xss vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // 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. – P. 7.

64. Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // 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.

65. Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

66. Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

67. Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез «Securitea informationala 2018». Conferenta

Кафедра_КБПЗ_2023рік

internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

68. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

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

70. Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

71. Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада

72. 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.

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

Кафедра_КБПЗ_2023рік

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

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

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

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

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

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

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

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

82. 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.

83. 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

84. 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

Кафедра_КБПЗ_2023рік

Додаток А
(обов'язковий)

Технічне завдання

Зміст

Кафедра_КОПЗ_2023рік

					ВКРБ-123.23.0003.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Коптєв Р.В.			Програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп'ютерних систем	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Буравченко К.О.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-19	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 7-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

Кафедра_КБПЗ_2023рік

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем;

– 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.

Кафедра_КБПЗ_2023рік

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

Кафедра_КбПЗ_2023рік

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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 78 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

Кафедра_КБПЗ_2023рік

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 1.06.2023 р.

Кафедра_КБПЗ_2023рік

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

Програмне забезпечення системи реінжирингу коду при модернізації інформаційних та комп’ютерних систем

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 40

Літера: РП

Кропивницький – 2023 року

Кафедра_КбПЗ_2023рік

Складальний файл DPR

```
program 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.
```

Кафедра_КбПЗ_2023рік

Файл Unit1.pas

```
unit Unit1;

interface

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

type
PI_Cobol_HEADER = ^I_Cobol_HEADER;
I_Cobol_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)
```

Кафедра КОПЗ 2023 рік

```
      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 MapCobolFileAndCheckSum(Filename: PChar;
  // Cobol File to get checksum
    var HeaderSum,
    CheckSum: DWORD // Calculated checksum
    ): DWORD;       // 0 if success
stdcall; external 'reingeniring.dll' name 'MapCobolFileAndCheckSumA';

  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 MapCobolFileAndCheckSum(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_Cobol_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 Cobol program file!');
     exit;
     end;
  Blockread(f,c,1);
  if c<>'E' then begin
```

Кафедра КБПЗ 2023рік

```
   Showmessage('Not a Cobol program file!');
   exit;
   end;
Blockread(f,c,1);
if c<>#0 then begin
   Showmessage('Not a Cobol program file!');
   exit;
   end;
Blockread(f,c,1);
if c<>#0 then begin
   Showmessage('Not a Cobol program file!');
   exit;
   end;
Seek(f,Dos.e_lfanew+24); // Go to the optional header
BlockRead(f,NT,sizeof(NT));
Showmessage('Cobol program file!');
if NT.CheckSum>0 then // Do not change if file has checksum
 Showmessage('Already has checksum '+inttohex(NT.CheckSum,8))
 else
 if MapCobolFileAndCheckSum(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.
```

Кафедра_КбПЗ_2023рік

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 '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;
```

Кафедра_КбПЗ_2023рік

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

 end.
```

Кафедра_КбПЗ_2023рік

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 );
```

Кафедра КбПЗ_2023рік

```
      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;
```

Кафедра_КБПЗ_2023рік

```
    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:
```

Кафедра_КбПЗ_2023рік

```
            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 Cobol';
    SEditorLines = 'Lines Pascal';

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

Кафедра_КбПЗ_2023рік

```
    with EdtStrings.Font do
    begin
  Name := Cobol.ReadString( Section, 'FontName', 'MS Sans Serif' );
  Size := Cobol.ReadInteger( Section, 'FontSize', 8 );
  Color := Cobol.ReadInteger( Section, 'FontColor', clWindowText );
  StyleBits := Cobol.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 := Cobol.ReadInteger( Section, 'TabSize', 8 );
    Left := Cobol.ReadInteger( Section, 'Left', ( Screen.Width - Width ) div 2
);
    Top := Cobol.ReadInteger( Section, 'Top', ( Screen.Height - Height ) div 2
);
    Width := Cobol.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
    Cobol.Free;
  end;


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

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

Кафедра_КОПЗ_2023рік

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

Кафедра_КбПЗ_2023рік

```
  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
```

Кафедра_КОПЗ_2023рік

```
    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;
```

Кафедра КБПЗ_2023рік

```
    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
```

Кафедра_КбПЗ_2023рік

```
    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 );
```

Кафедра_КбПЗ_2023рік

```
  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;
```

Кафедра КБПЗ_2023рік

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

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

  end.
```

Кафедра_КбПЗ_2023рік

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;
```

Кафедра КбПЗ_2023рік

```
      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;
```

Кафедра_КБПВ_2023рік

```
      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;
```

Кафедра_КОПЗ_2023рік

```
      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;
```

Кафедра_КбПЗ_2023рік

```
    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;
```

Кафедра КБПЗ – 2023рік

```
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
```

Кафедра_КОПЗ_2023рік

```
    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;
```

Кафедра_КбПЗ_2023рік

```
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;
```

Кафедра_КбПЗ_2023рік

```
  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
```

Кафедра_КбПЗ_2023рік

```
    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;
```

Кафедра_КбПЗ_2023рік

```
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 ),
```

Кафедра_КБПЗ_2023рік

```
                                        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
```

Кафедра_КОПЗ_2023рік

```
 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
```

Кафедра_КбПр_2023рік

```
    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;
```

Кафедра_КОПЗ_2023рік

```
  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;
```

Кафедра_КбПЗ_2023рік

```
      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
```

Кафедра_КбПЗ_2023рік

```
  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;
```

Кафедра_КОПЗ_2023рік

```
    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;
```

Кафедра_КОПЗ_2023рік

```
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
```

Кафедра_КбПЗ_2023рік

```
  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.
```

Кафедра_КбПЗ_2023рік

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.
```

Кафедра_КбПЗ_2023рік